Low-Latency Network-Adaptive Error Control for Interactive Streaming

Silas L. Fong
University of Toronto
Toronto, Ontario, Canada
silas.fong@utoronto.ca

Salma Emara
University of Toronto
Toronto, Ontario, Canada
salma@ece.utoronto.ca

Baochun Li
University of Toronto
Toronto, Ontario, Canada
bli@ece.toronto.edu

Ashish Khisti
University of Toronto
Toronto, Ontario, Canada

Wai-Tian Tan
Cisco Systems
San José, California

Xiaoqing Zhu
Cisco Systems
San José, California

John Apostolopoulos
Cisco Systems
San José, California

ABSTRACT

We introduce a novel network-adaptive algorithm that is suitable for alleviating network packet losses for low-latency interactive communications between a source and a destination. Network packet losses happen in a bursty manner as well as an arbitrary manner, where the former is usually due to network congestion and the latter can be caused by unreliable wireless links. Our network-adaptive algorithm estimates in real time the best parameters of a recently proposed streaming code that corrects both arbitrary losses (which cause crackling noise in audio) and burst losses (which cause undesirable jitters and pauses in audio) using forward error correction (FEC). The network-adaptive algorithm updates the coding parameters in real time as follows: The destination estimates appropriate coding parameters based on its observed packet loss pattern and then the parameters are fed back to the source for updating the underlying code. In addition, a new explicit construction of practical low-latency streaming codes that achieve the optimal tradeoff between the capability of correcting arbitrary losses and the capability of correcting burst losses is provided. Simulation evaluations based on real-world packet loss traces reveal that our proposed network-adaptive algorithm combined with our optimal streaming codes achieves significantly higher reliability compared to uncoded and non-adaptive FEC schemes over UDP (User Datagram Protocol).

CCS CONCEPTS

• **Networks** → **Transport protocols**; • **Theory of computation** → *Online learning algorithms.*

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3350942

KEYWORDS

network-adaptive algorithm, forward error correction (FEC), interactive streaming, low-latency

ACM Reference Format:
Silas L. Fong, Salma Emara, Baochun Li, Ashish Khisti, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. 2019. Low-Latency Network-Adaptive Error Control for Interactive Streaming. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3343031.3350942

1 INTRODUCTION

Real-time interactive streaming is an essential component for many low-latency applications over the Internet including high-definition video conferencing, augmented/virtual reality, and online gaming. In particular, low-latency video conferencing has been a cornerstone for communication and collaboration for individuals and enterprises. At the core of these important applications is the need to reliably deliver packets with low latency.

Packet erasure (loss) at the network layer for an end-to-end communication over the Internet is inevitable. Two main approaches have been implemented at the transport layer to control end-to-end error introduced by the network layer: Automatic repeat request (ARQ) and forward error correction (FEC). Both ARQ and FEC can alleviate the damages of packet losses that may be caused by unreliable wireless links or congestion at network bottlenecks. However, ARQ schemes are not suitable for real-time streaming applications that involve arbitrary global users because each retransmission will incur an extra round-trip delay which may be intolerable. Specifically, correcting an erasure using ARQ results in a 3-way delay (forward + backward + forward), and this aggregate (3-way) delay including transmission, propagation and processing delays is required to be lower than 150 ms for interactive applications such as voice and video according to the International Telecommunication Union (ITU) [9, 15]. This aggregate delay makes ARQ impractical for communication between two distant global users with aggregate delay larger than 150 ms (even if the signals travel at the speed

of light, the minimum possible aggregate delay between two diametrically opposite points on the earth's circumference is at least 200 ms [1]).

On the contrary, FEC schemes are amenable to low-latency communications among global users because no retransmission is required. Instead of using retransmissions to achieve high reliability, FEC schemes increase the correlation among the transmitted symbols by adding redundant information. Low-density parity-check (LDPC) and digital fountain codes are two traditional FEC schemes that are currently used in the DVB-S2 [2] and DVB-IPTV [3] standards for non-interactive streaming applications. However, they are typically operated over a few thousand symbols and are not suitable for interactive streaming applications where short block lengths (e.g., a few hundred symbols) are required due to the stringent delay constraints. On the other hand, low-latency FEC schemes that operate over short block lengths have been proposed to improve interactive communication [6, 10, 14, 17]. Indeed, the use of FEC schemes for protecting voice streams against packet erasures largely attributed to the success of Skype [7], and an adaptive hybrid NACK/FEC has been used in WebRTC to obtain a better trade-off between temporal quality, spatial video quality and end-to-end delay [6].

Recently, several systematic studies have been carried out to characterize the fundamental limits of streaming codes (i.e., low-latency FEC schemes) that correct burst and arbitrary (isolated) erasures [4, 11, 13], where the former is usually due to network congestion and the latter can be caused by unreliable wireless links. In particular, the authors in [4, 11] have provided a high-complexity construction of a class of FEC streaming codes that possess the following two properties:

(a) Correct both arbitrary and burst erasures, which cause crackling noise and undesirable jitters/pauses respectively for audio.

(b) Achieve the optimal tradeoff between the capability of correcting arbitrary erasures and the capability of correcting burst erasures.

Therefore, we are motivated to design real-time error control based on low-complexity FEC streaming codes that satisfy Properties (a) and (b) and implement the design in real-world networks. Our real-time error control consists of the following:

(i) A new explicit construction of low-latency streaming codes over GF(256) that satisfy Properties (a) and (b).

(ii) In order to take varying network conditions into account, we also design a network-adaptive algorithm that updates the parameters of our constructed low-latency streaming codes in real time as follows: The destination estimates appropriate coding parameters based on its observed erasure pattern and then the estimated parameters are fed back to the source for updating the code.

In addition, we conduct real-world experiments to demonstrate the performance of our network-adaptive FEC streaming scheme. Our experimental results reveal that our network-adaptive scheme achieves significantly higher reliability compared to uncoded and non-adaptive FEC schemes over UDP (User Datagram Protocol).

2 CONCEPT OF FEC STREAMING CODES

We first present a general framework of an FEC streaming code as illustrated in Figure 1. The source periodically generates a sequence of multimedia frames. Each multimedia frame together with a parity

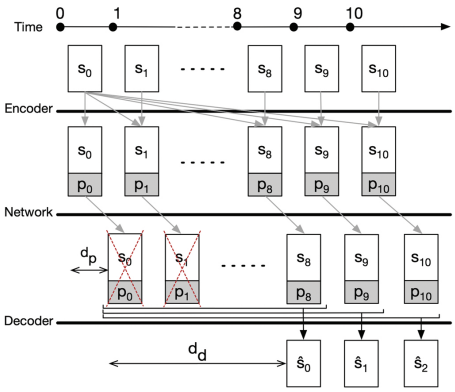


Figure 1: A general FEC framework

Figure 2: Packet drops marked by dark squares

frame is encapsulated in a network packet, which travels to the destination with a propagation delay d_{p}. The network packets can be dropped by the network in an arbitrary manner due to unreliable (wireless) links or in a bursty manner due to network congestion, which are illustrated in Figure 2 respectively. The destination aims to recover the multimedia frames sequentially subject to a decoding delay constraint d_{d}, where lost multimedia frames can be recovered with the help of subsequent parity frames. For example, if packets 0 and 1 are dropped as illustrated in Figure 1, then the parity frames in packets 2 to 7 may help recover frame 0 with decoding delay of 8 frames, and the parity frames in packets 2 to 8 may help recover frame 1 with the same decoding delay.

If we follow existing FEC technologies (e.g., WebRTC [6] and Skype [7]) and choose the parity frames based on coding over the past multimedia frames using maximum-distance separable (MDS) codes, the resulting FEC streaming code is optimal for correcting arbitrary erasures subject to the decoding delay d_{d}. For instance, if each parity frame equals the XOR between the past two frames (e.g., $\mathbf{p}_2 = \mathbf{s}_0 + \mathbf{s}_1$), then the resultant code can correct two arbitrary erasures with decoding delay of 3 frames (e.g., if packets 1 and 2 are erased, then $\mathbf{s}_1$ can be recovered from packets 3 and 4; if packets 1 and 3 are erased, then $\mathbf{s}_1$ can be recovered from packets 0 and 2). However, the above construction is not optimal for correcting burst erasures. In general, simple FEC streaming strategies based on MDS codes are not optimal.

In order to achieve the optimal tradeoff between the capability of correcting arbitrary losses and the capability of correcting burst losses subject to a decoding delay constraint, we have to carefully choose the parity frames. The existence of such optimal parity frames has been recently proved in [4, 11].

3 FORMULATION OF STREAMING CODES

This section formally states the optimality of streaming codes.

Notation

The sets of natural numbers and non-negative integers are denoted

by $\mathbb{N}$ and $\mathbb{Z}_+$ respectively. A finite field is denoted by $\mathbb{F}$. The set of k-dimensional row vectors over $\mathbb{F}$ is denoted by $\mathbb{F}^k$. A row vector in $\mathbb{F}^k$ is denoted by $\mathbf{a} \triangleq [a_0\ a_1\ \ldots\ a_{k-1}]$. The k-dimensional identity matrix is denoted by $\mathbf{I}_k$ and the $L \times B$ all-zero matrix is denoted by $\mathbf{0}^{L\times B}$. An $L \times B$ parity matrix of a systematic MDS $(L+B, L)$-code is denoted by $\mathbf{V}^{L\times B}$, where $L+B$ and L denote the blocklength and the number of data symbols respectively. For this MDS code, any L columns of $[\mathbf{I}_L\ \mathbf{V}^{L\times B}] \in \mathbb{F}^{L\times(L+B)}$ are independent. It is well known that a systematic MDS $(L+B, L)$-code always exists as long as $|\mathbb{F}| \geq L+B$ [12].

Definitions and Optimality of Streaming Codes

All the definitions below follow the conventions as in [4, 11, 13].

Definition 1. *An $(n,k,T)_{\mathbb{F}}$-streaming code consists of:*
(1) A sequence of messages $\{\mathbf{s}_i\}_{i=0}^{\infty}$ where each message $\mathbf{s}_i$ consists of k symbols in $\mathbb{F}$.
(2) An encoder f_i for each $i \in \mathbb{Z}_+$. Each f_i is used by the source at channel use i to encode the source messages according to $\mathbf{x}_i = f_i(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_i)$, where each packet $\mathbf{x}_i$ consists of n symbols in $\mathbb{F}$.
(3) A decoder φ_{i+T} for each $i \in \mathbb{Z}_+$. Each φ_{i+T} is used by the destination at channel use $i+T$ to produce $\hat{\mathbf{s}}_i \in \mathbb{F}^k$, an estimate of $\mathbf{s}_i$, based on the subset of packets $\{\mathbf{x}_j\}_{j=0}^{i+T}$ that are non-erased.

Definition 2. *An $(n,k,T)_{\mathbb{F}}$-streaming code is said to* correct any (B,N)-erasure *if $\hat{\mathbf{s}}_i = \mathbf{s}_i$ holds for all $i \in \mathbb{Z}_+$ for the following sliding-window channel: In any sliding window $\{i, i+1, \ldots, i+T\}$ of size $T+1$ starting at channel use $i \in \mathbb{Z}_+$, either a burst erasure of length at most B or multiple erasures of count at most N are introduced.*

Definition 3. *The (T,B,N)-capacity is the maximum rate k/n achievable by $(n,k,T)_{\mathbb{F}}$-streaming codes that correct all (B,N)-erasures.*

Theorem 1 ([4, 11]). *Define*

$$\mathrm{C}(T,B,N) \triangleq \frac{T-N+1}{T-N+B+1}. \tag{1}$$

For any $T \geq B \geq N \geq 1$, the (T,B,N)-capacity equals $\mathrm{C}(T,B,N)$.

Theorem 1 motivates the following definition of optimal codes.

Definition 4. *For any $T \geq B \geq N \geq 1$, an $(n,k,T)_{\mathbb{F}}$-code that corrects any (B,N)-erasure is said to be* optimal *if $\frac{k}{n} = \mathrm{C}(T,B,N)$.*

4 EXPLICIT CONSTRUCTION OF OPTIMAL STREAMING CODES OVER GF(256)

Although the existence of optimal streaming codes has been proved in [4, 11], no explicit construction over practical field size was given. Motivated by the fact that finite fields with characteristic 2 allow very efficient computation, we provide the first explicit construction of optimal streaming codes over GF(256) when $T \leq 11$. Readers who are uninterested in the explicit construction may skip the remaining part of this section and take the following result as granted:

> *"Let $\mathbb{F}$ = GF(256). For any $T \geq B \geq N \geq 1$, an $(n,k,T)_{\mathbb{F}}$-code that corrects any (B,N)-erasure can be efficiently generated."*

The explicit construction leverages a standard periodic interleaving approach which constructs streaming codes based on block codes as defined below.

Definition 5. *An $(n,k,T)_{\mathbb{F}}$-block code consists of:*
(1) A sequence of k symbols $\{s[i]\}_{i=0}^{k-1}$ where $s[i] \in \mathbb{F}$.
(2) A generator matrix $\mathbf{G} \in \mathbb{F}^{k\times n}$. The n codeword symbols are generated as $\big[x[0]\ \ldots\ x[n-1]\big] = \big[s[0]\ \ldots\ s[k-1]\big]\,\mathbf{G}$.
(3) A decoder φ_{i+T} for each $i \in \{0,1,\ldots,k-1\}$. Each φ_{i+T} is used by the destination at channel use $\min\{i+T, n-1\}$ to produce $\hat{s}[i] \in \mathbb{F}$, an estimate of $s[i]$, based on the subset of codeword symbols $\big[x[0]\ x[1]\ \ldots\ x[\min\{i+T, n-1\}]\big]$ that are non-erased.

Definition 6. *An $(n,k,T)_{\mathbb{F}}$-block code is said to* correct any (B,N)-erasure *if $\hat{\mathbf{s}}_i = \mathbf{s}_i$ holds for all $i \in \mathbb{Z}_+$ as long as either a burst erasure of length at most B or multiple erasures of count at most N occur in every sliding window of size $T+1$. The block code is said to be* optimal *if $\frac{k}{n} = \mathrm{C}(T,B,N)$.*

We will leverage the following lemma to construct optimal streaming codes based on optimal block codes. The lemma is a direct consequence of [4, Lemma 1] with the identification $W = T+1$. The proof is standard and is based on periodic interleaving (cf. [5] and [13, Sec. IV-A])).

Lemma 1. *Given an $(n,k,T)_{\mathbb{F}}$-block code which corrects any (B,N)-erasure, we can periodically interleave the block code and construct an $(n,k,T)_{\mathbb{F}}$-streaming code which corrects any (B,N)-erasure.*

Due to Lemma 1, Definition 4 and Definition 6, the search for explicit construction of optimal streaming codes over GF(256) reduces to the search for explicit construction of optimal block codes over GF(256). The following lemma states specific structures of the parity matrices of optimal codes. The lemmas are expressed with the help of the following definition of an *m-row N-diagonal matrix*:

$$\mathbf{D}_N^{m\times(N+m)} \triangleq \begin{bmatrix} d_0^{(0)} & \cdots & d_{N-1}^{(0)} & 0 & \cdots & \cdots & 0 \\ 0 & d_0^{(1)} & \cdots & d_{N-1}^{(1)} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & d_0^{(m-1)} & \cdots & d_{N-1}^{(m-1)} & 0 \end{bmatrix}$$

where $\{d_\ell^{(i)} \mid 0 \leq i \leq m-1, 0 \leq \ell \leq N-1\}$ assume arbitrary values.

Lemma 2 ([4, Lemmas 2, 3 and 4]). *Fix any $T \geq B \geq N \geq 1$ and let $k \triangleq T-N+1$ and $n \triangleq k+B$. If $k \geq B$ (i.e., $k/n \geq 1/2$), there exists a $\mathbf{P}$ having the form*

$$\left[\begin{array}{c} \mathbf{D}_N^{(B-N)\times B} \\ \hline \mathbf{0}^{N\times(B-N)} \;\vdots\; \mathbf{P}_{right} \\ \hline \mathbf{V}^{(k-B)\times B} \end{array}\right] \tag{2}$$

such that $\mathbf{G} = [\mathbf{I}_k\ \mathbf{P}]$ is the generator matrix of an (n,k,T)-code that corrects any (B,N)-erasures, where $\mathbf{D}_N^{m\times(N+m)}$ is an m-row N-diagonal matrix, $\mathbf{P}_{right}$ is an $N \times N$ matrix, and $\mathbf{V}^{(k-B)\times B}$ is a $(k-B)\times B$ parity matrix of a systematic MDS code. On the other hand, if $k < B$ (i.e., $k/n < 1/2$), there exists a $\mathbf{P}$ having the form

$$\left[\begin{array}{c:c} \mathbf{P}_{left} & \mathbf{D}_{k-B+N}^{(B-N)\times k} \\ \mathbf{V}_{left}^{(k-B+N)\times(B-k)} & \mathbf{0}\ \mathbf{V}_{right}^{(k-B+N)\times(k-B+N)} \end{array}\right] \tag{3}$$

such that $\mathbf{G} = [\mathbf{I}_k\ \mathbf{P}]$ is the generator matrix of an (n,k,T)-code that corrects any (B,N)-erasure, where $\mathbf{D}_{k-B+N}^{(B-N)\times k}$ is a $(B-N)$-row $(k-B+N)$-diagonal matrix, $\left[\mathbf{V}_{left}^{(k-B+N)\times(B-k)}\ \mathbf{V}_{right}^{(k-B+N)\times(k-B+N)}\right]$ constitutes the $(k-B+N)\times N$ parity matrix of a systematic MDS code, $\mathbf{P}_{left}$ is a $(B-N)\times(B-k)$ matrix, and $\mathbf{0}$ is the $(k-B+N)\times(B-N)$ zero matrix.

In the rest of the paper, we assume $\mathbb{F}$ = GF(256). Suppose we are given a $k \times B$ matrix $\mathbf{V}^{k\times B}$ where $k \geq 1$ and $B \geq 1$, and let $n \triangleq k + B$. Then, we construct a parity matrix $\mathbf{P} \in \mathbb{F}^{k\times B}$ as follows.

- If $k \geq B$, construct $\mathbf{P}$ by replacing every non-zero $(i,j)^{\text{th}}$ element of $\mathbf{P}$ in (2) with the $(i,j)^{\text{th}}$ element in $\mathbf{V}^{k\times B}$.
- If $k < B$, construct $\mathbf{P}$ by replacing every non-zero $(i,j)^{\text{th}}$ element of $\mathbf{P}$ in (3) with the $(i,j)^{\text{th}}$ element in $\mathbf{V}^{k\times B}$.

Let $C(\mathbf{V}^{k\times B})$ denote the $(n,k,T)_{\mathbb{F}}$-block code with generator matrix $\mathbf{G}$ as constructed above. If $C(\mathbf{V}^{k\times B})$ corrects any (B,N)-erasure and $k = T-N+1$, then $C(\mathbf{V}^{k\times B})$ is optimal by Definition 4. In search of a useful $\mathbf{V}^{k\times B}$, we define $\mathbf{V}_{\text{Cauchy}}^{(T-N+1)\times B} = \left[v_{ij}^{\text{Cauchy}}\right]_{\substack{0\leq i\leq T-N,\\ 0\leq j\leq B-1}}$ to be a $(T-N+1)\times B$ Cauchy matrix over GF(256) where $v_{ij}^{\text{Cauchy}} \triangleq (i+j+k)^{-1}$. Similarly, define $\mathbf{V}_{\text{Vand}}^{(T-N+1)\times B} = \left[v_{ij}^{\text{Vand}}\right]_{\substack{0\leq i\leq T-N,\\ 0\leq j\leq B-1}}$ to be a $(T-N+1)\times B$ Vandermonde matrix over GF(256) where $v_{ij}^{\text{Vand}} \triangleq 2^{i\times j}$. Using computer search, we obtain the following.

Proposition 3. *Let $\mathbb{F}$ = GF(256). For any $1 \leq N \leq B \leq T \leq 11$, the $(n,k,T)_{\mathbb{F}}$-block code $C(\mathbf{V}_{Cauchy}^{(T-N+1)\times B})$ corrects any (B,N)-erasure if $(T,B,N) \notin \{(10,8,4),(11,5,4)\}$. In addition, $C(\mathbf{V}_{Vand}^{(T-N+1)\times B})$ corrects any (B,N)-erasure if $(T,B,N) \in \{(10,8,4),(11,5,4)\}$.*

In view of Proposition 3, we define for any $1 \leq N \leq B \leq T \leq 11$ the parity matrix of an optimal $(n,k,T)_{\mathbb{F}}$-block code as

$$\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B} \triangleq \begin{cases} \mathbf{V}_{\text{Cauchy}}^{(T-N+1)\times B} & \text{if } (T,B,N) \notin \{(10,8,4),(11,5,4)\}, \\ \mathbf{V}_{\text{Vand}}^{(T-N+1)\times B} & \text{otherwise.} \end{cases}$$

Using Proposition 3 and the definition of $\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B}$, we conclude that $C(\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B})$ is an optimal block code. In addition, we can construct an optimal streaming code by periodically interleaving n copies of $C(\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B})$ as illustrated below (cf. Lemma 1).

Example 1. *Suppose we are given a $(3,3,5)_{\mathbb{F}}$-block code that corrects any $(3,2)$-erasure with generator matrix*

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 \end{bmatrix}.$$

The block code is optimal by Definition 4. Suppose we have a streaming message $\{\mathbf{s}_i\}_{i\in\mathbb{Z}_+}$ where $\mathbf{s}_i = \begin{bmatrix} s_i[0] & s_i[1] & s_i[2] \end{bmatrix} \in \mathbb{F}^3$. From channel use $i-2$ to $i+5$, the symbols yielded by the $(6,3,5)_{\mathbb{F}}$-code constructed by periodically interleaving the $(6,3,5)_{\mathbb{F}}$-block code according Lemma 1 are shown in Table 1. The symbols in Table 1 which are highlighted in the same color diagonally (in direction $\searrow$) are encoded using the same $(6,3,5)_{\mathbb{F}}$-block code. Given the fact that the $(6,3,5)_{\mathbb{F}}$-block code corrects any $(3,2)$-erasure, we can see from Table 1 that $\mathbf{s}_i = \begin{bmatrix} s_i[0] & s_i[1] & s_i[2] \end{bmatrix}$ can be perfectly recovered by channel use $i+5$ as long as a burst erasure of length no longer than 3 or no more than 2 arbitrary erasures occur in every sliding window of size 6.

By periodically interleaving n copies of $C(\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B})$, we can construct an (n,k,T)-code with $k = T-N+1$ and $n = k+B$, which is optimal if $T \leq 11$ by Proposition 3. For any $1 \leq N \leq B \leq T \leq 11$, we let $C_{T,B,N}$ denote the optimal (n,k,T)-code that is constructed by interleaving n copies of $C(\mathbf{V}_{\text{optimal}}^{(T-N+1)\times B})$. The optimal streaming codes $C_{T,B,N}$ are the building blocks for the network-adaptive streaming scheme described in the next section.

5 NETWORK-ADAPTIVE ALGORITHM

Algorithm 1: Estimating conservative B and N

Result: $\hat{B}_i$ and $\hat{N}_i$ are generated at the destination for every packet i where $0 \leq i \leq L-1$. All correctible length-$(T+1)$ erasure patterns that occur by channel use i can be perfectly recovered by any code that corrects all $(\hat{B}_i, \hat{N}_i)$-erasures.

Inputs : T, L and e^L denoting decoding delay, duration, and length-L erasure pattern respectively.

Outputs : $\hat{B}_i$ and $\hat{N}_i$ for every packet i.

previous_seq_# $\leftarrow -1$
$(\hat{B}_{-1}, \hat{N}_{-1}, N_{\max}) \leftarrow (0,0,0)$
for $i \leftarrow 0$ ***to*** $L-1$ **do** `// packets 0 to L − 1 sent`
  if $e_i = 0$ **then** `// packet i not erased`
    current_seq_# $\leftarrow i$
    for $j \leftarrow$ previous_seq_# $+1$ ***to*** current_seq_# **do**
      $\mathcal{W} \leftarrow \{j-T, j-T+1, \ldots, j\}$
      $\bar{B}_j \leftarrow \max\{\text{span}(e_{\mathcal{W}})\}$
      $\bar{N}_j \leftarrow \max\{\text{wt}(e_{\mathcal{W}}), \hat{N}_{j-1}\}$
      $N_{\max} \leftarrow \max\{\text{wt}(e_{\mathcal{W}}), N_{\max}\}$
      if $\bar{N}_j = 0$ or $\bar{N}_j = T+1$ **then** `// trivial`
        $(\hat{B}_j, \hat{N}_j) \leftarrow (\hat{B}_{j-1}, \hat{N}_{j-1})$
      else `// calculate 3 hypothetical rates`
        $R_B \leftarrow \begin{cases} 0 & \text{if } \bar{B} = T+1, \\ \text{C}(T, \bar{B}_j, \max\{\hat{N}_{j-1}, 1\}) & \text{if } \bar{B} < T+1 \end{cases}$
        $R_N \leftarrow \text{C}(T, \max\{\bar{B}_{j-1}, \bar{N}_j\}, \bar{N}_j)$
        $R_{\text{MDS}} \leftarrow \text{C}(T, N_{\max}, N_{\max})$
        switch $\max\{R_B, R_N, R_{\text{MDS}}\}$ **do**
          case R_B **do** `// R_B largest`
            $(\hat{B}_j, \hat{N}_j) \leftarrow (\bar{B}_j, \max\{\hat{N}_{j-1}, 1\})$
            break
          case R_N **do** `// R_N largest`
            $(\hat{B}_j, \hat{N}_j) \leftarrow (\max\{\hat{B}_{j-1}, \bar{N}_j\}, \bar{N}_j)$
            break
          case R_{MDS} **do** `// R_MDS largest`
            $(\hat{B}_j, \hat{N}_j) \leftarrow (N_{\max}, N_{\max})$
        end
      end
    end
    previous_seq_# $\leftarrow$ current_seq_#
  end
end

We first present a conservative algorithm illustrated by Algorithm 1 that estimates conservative coding parameters B and N in L channel uses. Before the algorithm tracks any packet erasures, the decoding delay denoted by T and the duration of the algorithm denoted by L are fixed, and the initial estimates for B and N, denoted

Channel use	$i-2$	$i-1$	i	$i+1$	$i+2$	$i+3$	$i+4$	$i+5$
symbol 0	$s_{i-2}[0]$	$s_{i-1}[0]$	$s_i[0]$	$s_{i+1}[0]$	$s_{i+2}[0]$	$s_{i+3}[0]$	$s_{i+4}[0]$	$s_{i+5}[0]$
symbol 1	$s_{i-2}[1]$	$s_{i-1}[1]$	$s_i[1]$	$s_{i+1}[1]$	$s_{i+2}[1]$	$s_{i+3}[1]$	$s_{i+4}[1]$	$s_{i+5}[1]$
symbol 2	$s_{i-2}[2]$	$s_{i-1}[2]$	$s_i[2]$	$s_{i+1}[2]$	$s_{i+2}[2]$	$s_{i+3}[2]$	$s_{i+4}[2]$	$s_{i+5}[2]$
symbol 3	$\ddots$	$\ddots$	$\ddots$	$s_{i-2}[0]$	$s_{i-1}[0]$	$s_i[0]$	$\ddots$	$\ddots$
symbol 4	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$s_{i-2}[0]+s_{i-1}[1]+s_i[2]$	$s_{i-1}[0]+s_i[1]+s_{i+1}[2]$	$s_i[0]+s_{i+1}[1]+s_{i+2}[2]$	$\ddots$
symbol 5	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$s_{i-1}[1]+2s_i[2]$	$s_i[1]+2s_{i+1}[2]$	$s_{i+1}[1]+2s_{i+2}[2]$

Table 1: Symbols yielded by a $(6,3,5)_{\mathbb{F}}$-code through interleaving a $(6,3,5)_{\mathbb{F}}$-block code.

by $\hat{B}_{-1}$ and $\hat{N}_{-1}$ respectively, are both set to 0. In other words, the channel is assumed to be initially ideal which introduces no erasure. In addition, the variable $N_{\max}$ which keeps track of the maximum number of arbitrary erasures is set to 0.

Every packet transmitted at channel use $i \in \mathbb{Z}_+$ is assumed to either reach the destination in the same channel use or be erased. In practice, packets that are either dropped in a network or received in the wrong order are erased. For every non-erased packet received at channel use $i \in \mathbb{Z}_+$, the algorithm first deduces the erasure pattern $e_{\mathcal{W}} \triangleq (e_{j-T}, e_{j-T+1}, \ldots, e_j) \in \{0,1\}^{T+1}$ for each sliding window $\mathcal{W} = \{j-T, j-T+1, \ldots, j\}$ of size $T+1$ such that $j \le i$, where an element of $e_{\mathcal{W}}$ equals 1 if and only if the corresponding packet is erased. Let $\mathrm{wt}(e_{\mathcal{W}}) \triangleq \sum_{\ell \in \mathcal{W}} e_\ell$ and

$$\mathrm{span}(e_{\mathcal{W}}) \triangleq \begin{cases} 0 & \text{if } \mathrm{wt}(e_{\mathcal{W}}) = 0, \\ p_{\text{last}} - p_{\text{first}} + 1 & \text{otherwise} \end{cases}$$

be the *weight* and *span* of $e_{\mathcal{W}}$ respectively, where p_{first} and p_{last} denote respectively the channel use indices of the first and last non-zero elements in $e_{\mathcal{W}}$. Intuitively speaking, $\mathrm{span}(e_{\mathcal{W}})$ is the minimum length over all intervals that contain the support of $e_{\mathcal{W}}$. For each deduced erasure pattern $e_{\mathcal{W}} = (e_{j-T}, e_{j-T+1}, \ldots, e_j)$, the algorithm first calculates $\mathrm{wt}(e_{\mathcal{W}})$ and $\mathrm{span}(e_{\mathcal{W}})$, and then assign the values to $(\bar{B}_j, \bar{N}_j, N_{\max})$ as shown at the beginning of the pseudocode. Then one of the following updates will occur:

(i) $\hat{B}_j$ will be assigned the value $\bar{B}_j$.
(ii) $\hat{N}_j$ will be assigned the value $\bar{N}_j$.
(iii) Both $\hat{B}_j$ and $\hat{N}_j$ will be assigned the value $N_{\max}$.

More specifically, the estimates $\hat{B}_j$ and $\hat{N}_j$ will be output according to the following three mutually exclusive cases:

Case $\bar{N}_j = 0$: In this case,

$$\bar{B}_j = \bar{N}_j = \mathrm{wt}(e_{\mathcal{W}}) = \mathrm{span}(e_{\mathcal{W}}) = \hat{N}_{j-1} = \hat{B}_{j-1} = 0,$$

which implies that no erasure has yet occurred upon the receipt of packet j. Then, Algorithm 1 sets $\hat{N}_j = \hat{B}_j = 0$, meaning that the estimates for N and B remain to be zeros.

Case $\bar{N}_j = T+1$: In this case, all the elements of $e_{\mathcal{W}}$ equal one, meaning that all the packets in the window $\{j-T, j-T+1, \ldots, j\}$ are erased, which implies that no $(n,k,T)_{\mathbb{F}}$-code can correct $e_{\mathcal{W}}$. Therefore, Algorithm 1 sets $\hat{N}_j = \hat{N}_{j-1}$ and $\hat{B}_j = \hat{B}_{j-1}$, meaning that the estimates for N and B remain unchanged.

Case $0 < \bar{N}_j \neq T+1$: In this case, every length-$(T+1)$ erasure pattern ε^{T+1} that has occurred up to channel use j can be classified into the following two types, with the terminology that ε^{T+1} is a (B,N)-erasure if either $\mathrm{span}(e_{\mathcal{W}}) \le B$ or $\mathrm{wt}(e_{\mathcal{W}}) \le N$ holds:

(i) ε^{T+1} consists of all ones, hence it is uncorrectable;
(ii) ε^{T+1} is simultaneously a $(\bar{B}_j, \max\{\hat{N}_{j-1}, 1\})$-erasure, a $(\max\{\hat{B}_{j-1}, \bar{N}_j\}, \bar{N}_j)$-erasure, and a $(N_{\max}, N_{\max})$-erasure.

Note that by construction, every length-$(T+1)$ erasure pattern up to channel use $j-1$ can be classified into either Type (i) or is an $(\hat{B}_{j-1}, \hat{N}_{j-1})$-erasure. Therefore, Algorithm 1 calculates the best estimates for $\hat{B}_j$ and $\hat{N}_j$ so that the following two conditions hold:

(I) Every length-$(T+1)$ erasure pattern up to channel use j can be classified into either Type (i) or is a $(\hat{B}_j, \hat{N}_j)$-erasure.
(II) The loss in the maximum achievable rate induced by updating the estimates from $(\hat{B}_{j-1}, \hat{N}_{j-1})$ to $(\hat{B}_j, \hat{N}_j)$ is minimized.

The presence of Condition (II) is essential because without Condition (II), the algorithm can always output the trivial estimates $\hat{B}_j = \hat{N}_j = T$ that lead to the lowest rate $\mathrm{C}(T,T,T) = \frac{1}{T+1}$.

In order to calculate the best estimates for $\hat{B}_j$ and $\hat{N}_j$ so that Conditions (I) and (II) hold, Algorithm 1 computes the three hypothetical rates $(R_B, R_N, R_{\mathrm{MDS}})$ based on the (T,B,N)-capacity as shown in the middle of the pseudocode, where R_B denotes the hypothetical maximum achievable rate if $\hat{B}_j$ is assigned the value $\bar{B}_j$ followed by $\hat{N}_j$ being assigned the value $\max\{\hat{N}_{j-1}, 1\}$ (note that any $(\hat{B}_j, \hat{N}_{j-1})$-erasure is also a $(\hat{B}_j, \max\{\hat{N}_{j-1}, 1\})$-erasure), R_N denotes the hypothetical maximum achievable rate if $\hat{N}_j$ is assigned the value $\bar{N}_j$ followed by $\hat{B}_j$ being assigned the value $\max\{\hat{B}_{j-1}, \hat{N}_j\}$ (note that any $(\hat{B}_{j-1}, \hat{N}_j)$-erasure is also a $(\max\{\hat{B}_{j-1}, \hat{N}_j\}, \hat{N}_j)$-erasure), and R_{MDS} denotes the hypothetical maximum rate if both $\hat{B}_j$ and $\hat{N}_j$ are assigned the same value $N_{\max}$. Finally, Algorithm 1 sets $(\hat{B}_j, \hat{N}_j)$ as shown at the end of the pseudocode so that the resultant maximum achievable rate $\mathrm{C}(T, \hat{B}_j, \hat{N}_j)$ equals $\max\{R_B, R_N, R_{\mathrm{MDS}}\}$.

Combining the above three cases, we conclude that for all $0 \le i \le L-1$, Algorithm 1 generates estimates $(\hat{B}_i, \hat{N}_i)$ such that Conditions (I) and (II) hold. Algorithm 1 provides conservative estimates for B and N in the sense that the algorithm yields a code that perfectly corrects all length-$(T+1)$ correctible erasure sequences that have been observed.

An obvious drawback of Algorithm 1 is that the sequence of recommended coding rates is monotonically decreasing over time. Therefore, we propose the following *network-adaptive algorithm* based on interleaving Algorithm 1 as follows: At each channel use $\ell = 0, L, 2L, \ldots$, an instance of Algorithm 1 denoted by $\mathcal{A}_\ell$ is initiated. Each $\mathcal{A}_\ell$ lasts for $2L$ channel uses, and let $(\hat{B}_j^{(\ell)}, \hat{N}_j^{(\ell)})$ denote the corresponding estimates at channel use j. Then at each

channel use j, the network-adaptive algorithm outputs the estimate $(\hat{B}_j^{(\ell)}, \hat{N}_j^{(\ell)})$ provided by $\mathcal{A}_\ell$ at channel use j where ℓ is the unique integer that satisfies $\ell + L \leq j < j + 2\ell$. In other words, each interleaved Algorithm 1 will run for $2L$ channel uses where the first L estimates are ignored by the algorithm and the last L estimates are output by the algorithm. Due to our construction, the coding rate generated by our network-adaptive algorithm is not monotonically decreasing over time. Particularly, if there are no erasures for consecutive $2L$ channel uses, the next estimates of (B, N) would be $(0, 0)$.

6 NETWORK-ADAPTIVE STREAMING CODE

Based on the estimates $\{(\hat{B}_j, \hat{N}_j)\}_{j \in \mathbb{Z}_+}$ suggested by the network-adaptive algorithm (which could be computed at the destination and then fed back to the source), the source adjusts the streaming encoder accordingly so that more erasure patterns can be corrected at the cost of a small rate loss. Initially starting from channel use 0, the network-adaptive algorithm outputs $(\hat{B}_0, \hat{N}_0) = (0, 0)$ and the source will use the trivial rate-1 encoder so that the transmitted codeword is identical to the generated message. The source continues to use the trivial rate-1 encoder until the algorithm updates the estimates for B and N to positive values.

Whenever the algorithm provides new estimates for (B, N) at channel use j denoted by $(\hat{B}_j, \hat{N}_j)$, the source will switch to the new encoder associated with the code $C_{T,\hat{B}_j,\hat{N}_j}$ (defined at the end of Section 4). In order to ensure a smooth transition from using an old encoder with parameters $(B_{\text{old}}, N_{\text{old}})$ to using a new encoder with parameters $(B_{\text{new}}, N_{\text{new}}) \neq (B_{\text{old}}, N_{\text{old}})$, the source has to ensure that every transmitted packet is protected by either the old or new encoder. The smooth transition is carried out as described below.

Suppose the source wants to use a new encoder associated with $C_{T,B_{\text{new}},N_{\text{new}}}$ starting from channel use i, it will use both the old and new encoders to encode the same message into old and new codewords from channel use i to channel use $i + T$. As a result, the messages generated before channel use $i + T$ are protected by the old codewords in $C_{T,B_{\text{old}},N_{\text{old}}}$, while the messages generated from channel use $i + T$ until the next transition will be protected by the new codewords in $C_{T,B_{\text{new}},N_{\text{new}}}$. During the next transition, the new encoder will be replaced by another encoder and treated as an old encoder, and the transition procedure repeats.

Since every message is protected by either an old encoder with parameters $(B_{\text{old}}, N_{\text{old}})$ or a new encoder with parameters $(B_{\text{new}}, N_{\text{new}})$ during the transition, any $(B_{\text{old}}, N_{\text{old}})$-erasure of length-$(T + 1)$ that occurs before the transition can be corrected by the old encoder and any $(B_{\text{new}}, N_{\text{new}})$-erasure of length-$(T + 1)$ that occurs during and after the transition can be corrected by the new encoder.

The prototype of our proposed network-adaptive streaming scheme is illustrated in Figure 3, which is explained as follows. The parameter estimator uses the network-adaptive algorithm to generate the estimates $(\hat{B}_i, \hat{N}_i)$ for each $i \in \mathbb{Z}_+$. At each channel use i, an FEC message is generated which consists of a data buffer, an integer specifying the size of the buffer, a sequence number and the latest available estimates $(\hat{B}, \hat{N})$. The FEC message is then encoded into an FEC codeword and transmitted through the erasure channel. Each FEC codeword consists of a codeword buffer, an integer specifying the size of the codeword buffer, the sequence

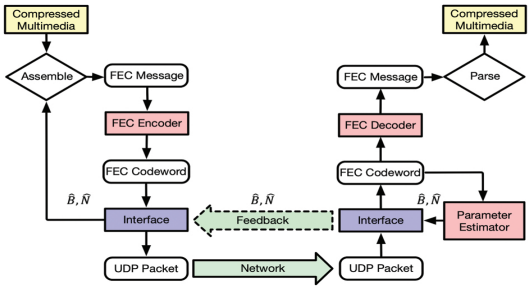


Figure 3: Prototype of network-adaptive streaming scheme

number originated from the corresponding FEC message, and the coding parameters (B, N). For every codeword received at channel use i, the destination decodes all the messages generated before channel use $i - T$ that have not been decoded yet, where the appropriate decoder can be chosen by the destination based on the coding parameters contained in all the received codewords up to channel use i. Therefore, every received codeword may result in more than one decoded message. For every reconstructed FEC message, the corresponding data buffer, the size of the buffer and the sequence number are extracted for further processing.

7 EXPERIMENTAL EVALUATION

Practical Implementation

In order to explore the potential of our proposed network-adaptive streaming scheme, we implement the proposed scheme for low-latency communication between two users over the Internet and present experimental results that compare our streaming codes with non-adaptive streaming strategies.

Suppose a source transmits a stream of compressed multimedia frames to a destination over the Internet. Each compressed multimedia frame could be generated from raw data by using a standard video or voice codec. Next, the compressed frame together with the estimated coding parameters received from the feedback channel is encapsulated in an FEC message. The FEC message is further encoded into an FEC codeword to be encapsulated in a network packet, which is then forwarded to the destination. Every network packet is either received by the intended destination or dropped (erased). At the destination side, every received FEC codeword is extracted from every received network packet and one or more FEC messages are recovered based on the codeword. A recovered compressed multimedia frame is further extracted from every recovered FEC message and then decompressed back to raw data by the video or voice codec.

The two interface modules between the streaming code and the network layer are illustrated in Figure 3. The first module is the interface at the source side that simultaneously encapsulates every FEC codeword into a UDP packet and forwards every estimated parameters received from the feedback channel to the message assembler. The second module is the interface at the receiver side that simultaneously extracts the codeword buffer in every network packet to form an FEC codeword and forwards every estimated parameters to the feedback channel over UDP.

Parameters and Error Metrics

We compare the frame loss rates (FLRs) achieved by the uncoded scheme and our network-adaptive streaming scheme as described in Section 5. In addition, we illustrate the effectiveness of our proposed network-adaptive algorithm by comparing the FLRs achieved by our adaptive streaming scheme with a delay constraint of T frames and non-adaptive schemes with fixed coding parameters (B, N). To this end, we fix the frame duration and bit rate for the compressed multimedia frame to be 10 ms and 240 kbit/s respectively, which are practical as existing audio codecs typically have frame duration 2.5–60 ms and bit rate 6–510 kbit/s [8, 16]. Consequently, every 300-byte compressed frame is generated every 10 ms. The 10 ms frame duration and the delay constraint T must be carefully chosen so that the resultant playback delay $T\times 10$ ms in addition to the propagation delay must be smaller than the 150 ms delay required by ITU for interactive applications [9, 15]. For example, if the propagation delay is 100 ms, then the resultant playback delay must be less than $150-100=50$ ms, which can be achieved by choosing T and frame duration such that their product is below 50 ms.

For our experimental purpose, we assume the propagation delay is less than 50 ms and choose $T = 10$ so that the resultant playback delay $T \times 10 = 100$ ms in addition to the propagation delay is below 150 ms. We set $L = 1000$ for the network-adaptive algorithm described in Section 5. In other words, each interleaved Algorithm 1 will run for $2L\times 10/1000 = 20$ seconds where the $L = 1000$ estimates produced in the first 10 seconds are ignored by the algorithm and the next $L = 1000$ estimates are output by the algorithm.

Let $M = 360$ be the number of 10-second sessions throughout the transmission, which involves a total of $L \times M = 1000 \times M$ packets lasting for one hour. In each session, $L = 1000$ packets are transmitted from the source to the destination. For simplicity, let the sequence number of a packet be its channel use index, starting from 0 and ending at $LM - 1$. During each session $m \in \{1, 2, \ldots, M\}$, the source transmits packets with sequence number between $L(m-1)$ and $Lm-1$. For each session m, let ε_m denote the corresponding FLR achieved by our network-adaptive streaming scheme. More precisely, $L(1-\varepsilon_m)$ is the number of FEC messages with sequence number between $L(m-1)$ and $Lm-1$ which are perfectly recovered by the destination. We will express in the next section our experimental results in terms of $\varepsilon_{\text{avg}} \triangleq \frac{1}{M}\sum_{m=1}^{M}\varepsilon_m$ and $\varepsilon_{\text{low-fi}} \triangleq \frac{1}{M}\sum_{m=1}^{M}\mathbf{1}\{\varepsilon_m > 0.1\}$ where ε_{avg} characterizes the average FLR and $\varepsilon_{\text{low-fi}}$ characterizes the fraction of undesirable low-fidelity sessions with FLR larger than 10%.

Experimental Results

In our experiment, the source and the destination are connected to the same Wi-Fi network with capacity $\approx$ 30 Mbit/s subject to UDP cross traffic introduced by Iperf. We call the UDP cross traffic *offered load*. The average FLRs for our network-adaptive streaming scheme and the uncoded scheme are plotted against the percentage of the network capacity occupied by Iperf traffic in Figure 4. In addition, we use the packet loss traces recorded during the real-world experiments performed for our adaptive streaming scheme to simulate the average FLR for the best non-adaptive streaming code $C_{T,B,N}$ whose coding rate does not exceed the average coding rate of the adaptive scheme. The average FLR for the best non-adaptive streaming code $C_{T,B,N}$ with parameters (B, N) is plotted in Figure 4. Figure 4 shows that our adaptive streaming scheme achieves a significantly lower average FLR than the rest.

strategy	*redundancy*	*average FLR*	*low-fi fraction*
network-adaptive	19.03%	0.01190	0.03047
MDS-adaptive	20.17%	0.01757	0.03878
no coding	0%	0.03489	0.01856

(a) $T = 10$ **for 30%-capacity offered load**

network-adaptive	16.6%	0.02586	0.07202
MDS-adaptive	17.2%	0.02617	0.07756
no coding	0%	0.04003	0.17452

(b) $T = 9$ **for 30%-capacity offered load**

network-adaptive	16.10%	0.00486	0.00831
MDS-adaptive	17.24%	0.01040	0.01108
no coding	0%	0.02486	0.13573

(c) $T = 11$ **for 30%-capacity offered load**

Table 2: Performance of streaming strategies

For interactive audio, low-fidelity sessions lead to unclear speech or even call termination which directly affects user experience. Figures 5 shows that our adaptive streaming scheme provides a substantially better audio quality than the other two. In Figure 6a, we show the variation of average FLRs for our adaptive streaming scheme and UDP across the 360 sessions under 40%-capacity offered load (i.e., the cross traffic equals 40% of the network capacity), and the sessions with packet loss are highlighted in Figure 6b. It can be seen from Figure 6 that for more than 1/4 of the sessions that experience packet loss, our adaptive scheme achieves less than half of the UDP loss. In addition, we display the variation of FEC redundancy (i.e., one minus coding rate) in Figure 7, which demonstrates how quickly our adaptive algorithm reacts to erasures.

The reason why our adaptive scheme significantly outperforms non-adaptive ones can be explained with the help of Figure 8. As shown in Figure 8, 11 out of 40 of the network packets are dropped. Our adaptive coding scheme updates the code in this order: $C_{10,1,1}$ and $C_{10,5,2}$ before transmitting packets 4 and 18 respectively. Therefore, the subsequent five packets losses are all recovered by $C_{10,5,2}$ as shown in Figure 8, whereas the fixed-rate code $C_{10,4,4}$ can only recover one packet.

Finally, in order to demonstrate the advantage of using our constructed streaming codes over traditional MDS-based codes, we consider the following *MDS-adaptive scheme*: Instead of outputting the coding parameters $(\hat{B}, \hat{N})$ for an optimal block code which corrects a length-$\hat{B}$ burst erasure and $\hat{N}$ arbitrary erasures, the adaptive algorithm outputs a single coding parameter N of an MDS code which corrects only N arbitrary erasures such that the resultant coding rate $\frac{T-N+1}{T+1}$ is approximately $\mathrm{C}(T, \hat{B}, \hat{N})$ (cf. (1)). We use the recorded packet loss traces obtained from a repeated experiment with 30%-capacity offered load to compare the network-adaptive streaming scheme with the MDS-adaptive scheme for $T = 10$. Table 2a shows that although the two schemes have similar rates, the network-adaptive scheme achieves around 80% of the average FLR and 80% of the low-fidelity sessions achieved by the MDS-adaptive scheme, which implies that our constructed optimal streaming

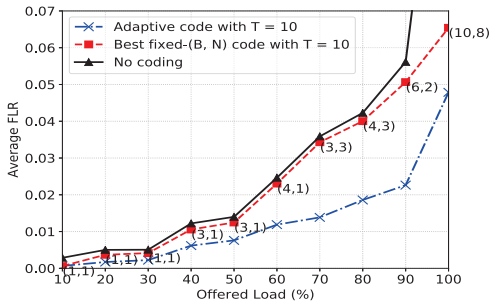


Figure 4: Average frame loss rate (FLR)

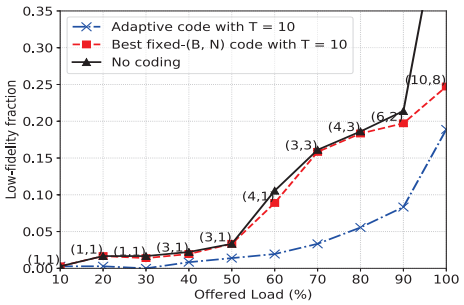
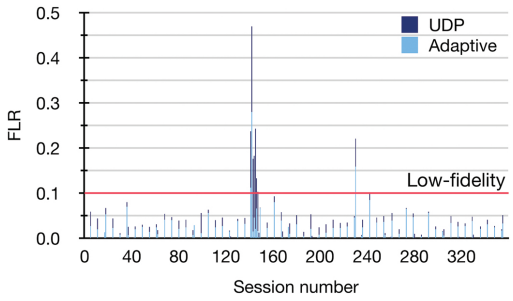
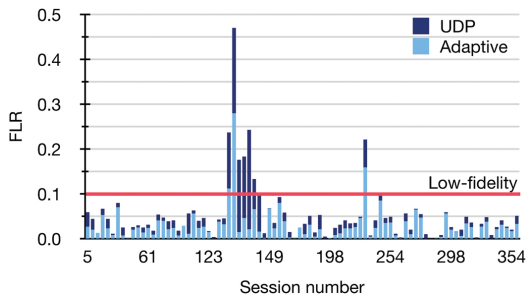


Figure 5: Low-fidelity fraction



(a) Original version with all frames



(b) Modified version without error-free frames

Figure 6: Average FLRs for adaptive FEC and UDP over time for 40%-capacity offered load

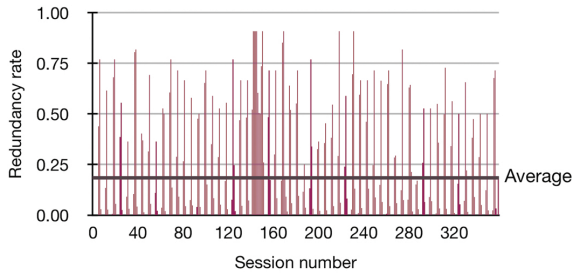


Figure 7: FEC redundancy for 40%-capacity offered load

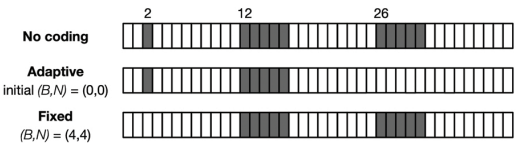


Figure 8: Packet losses recovered by different schemes

codes outperforms traditional MDS-based streaming codes in real-world networks. The reduction in FLR is not surprising because our optimal streaming codes treat burst erasures and arbitrary erasures differently while MDS-based codes do not differentiate them. Even if T slightly deviates from 10, our experimental results displayed in Tables 2b and 2c show that the network-adaptive scheme compared with the MDS-adaptive scheme can achieve a considerable reduction in FLR.

8 CONCLUSION AND FUTURE WORK

We have designed a network-adaptive FEC streaming scheme which consists of (i) a network-adaptive algorithm for estimating the coding parameters of streaming codes that correct both burst and arbitrary network packet losses, and (ii) an explicit construction of low-latency optimal streaming codes over GF(256) for $T \leq 11$. The computation bottleneck of our network-adaptive streaming scheme is bounded above by Gauss-Jordan elimination, which is used for decoding a length-$(T+1)$ block code and has at most $O(T^3)$ complexity. More precisely, the computation bottleneck is close to $O(D^3)$ where D denotes the average number of lost packets in a sliding window of size $T+1$. Real-world experiments reveal that our adaptive streaming scheme significantly outperforms non-adaptive ones. There are several interesting directions for future investigation: (i) Finding the largest T such that optimal streaming codes exist over GF(256) remains open. (ii) Future work may explore the interplay between our adaptive streaming scheme and existing congestion control algorithms that adjust the sizes of streaming messages in real time. (iii) The investigation of how multiple instances of our network-adaptive scheme compete with each other in a congested environment is interesting. (iv) The choice of L for our heuristic adaptive algorithm, set to 1000 in this work, could be optimized with extra effort. Also, machine learning techniques could be used to develop new network-adaptive algorithms. (v) This work compares FEC schemes only. Considering additional complementary methods in existing industrial schemes (e.g., Skype, WebRTC and other ARQ-based schemes) such as Adaptive Media Playout, adaptive live encoder and retransmissions is meaningful.

REFERENCES

[1] A. Khisti A. Badr, W.-T. Tan, and J. Apostolopoulos. 2017. Perfecting Protection for Interactive Multimedia: A survey of forward error correction for low-delay interactive applications. *IEEE Signal Process. Mag.* 34 (2017), 95 – 113. Issue 2.

[2] European Telecommunications Standards Institute. 2014. *Digital video broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; Part 1: DVB-S2.* ETSI EN 302 307-1.

[3] European Telecommunications Standards Institute. 2016. *Digital video broadcasting (DVB); Transport of MPEG-2 TS based DVB services over IP based networks.* ETSI TS 102 034.

[4] S. L. Fong, A. Khisti, B. Li, W.-T. Tan, X. Zhu, and J. Apostolopoulos. 2019. Optimal Streaming Codes for Channels with Burst and Arbitrary Erasures. *IEEE Trans. Inf. Theory* 15, 7 (2019), 4274–4292.

[5] G. D. Forney. 1971. Burst-Correcting Codes for the Classic Bursty Channel. *IEEE Trans. Inf. Theory* 19, 5 (1971), 772 – 781.

[6] S. Holmer, M. Shemer, and M. Paniconi. 2013. Handling packet loss in WebRTC. In *Proc. IEEE Intl. Conference on Image Process.* Melbourne, VIC, Australia.

[7] T. Huang, P. Huang, K. Chen, and P. Wang. 2010. Could Skype be more satisfying? A QoE-centric study of the FEC mechanism in an Internet-scale VoIP system. *IEEE Netw.* 24, 2 (2010), 42 —- 48.

[8] International Telecommunication Union. 1998. *Pulse code modulation (PCM) of voice frequencies.* Recommendation G.711.

[9] International Telecommunication Union. 2003. *One-way transmission time.* Recommendation G.114.

[10] J. Korhonena and P. Frossard. 2009. Flexible forward error correction codes with application to partial media data recovery. *Signal Processing: Image Communication* 24, 3 (2009), 229 – 242.

[11] M. Nikhil Krishnan and P. Vijay Kumar. 2018. Rate-Optimal Streaming Codes for Channels with Burst and Isolated Erasures. In *Proc. of IEEE Intl. Symp. on Inf. Theory.* Vail, CO, USA, 1809 – 1813.

[12] F. J. MacWilliams and N. J. A. Sloane. 1988. *The Theory of Error-Correcting Codes* (1st ed.). North-Holland, Netherlands, Amsterdam, Holland.

[13] E. Martinian and C.-E. W. Sundberg. 2004. Burst erasure correction codes with low decoding delay. *IEEE Trans. Inf. Theory* 50, 10 (2004), 2494 – 2502.

[14] M. Nagy, V. Singh, J. Ott, and L. Eggert. 2014. Congestion control using FEC for conversational multimedia communication. In *Proc. the 5th ACM Multimedia Systems Conference.* Singapore, 191–202.

[15] T. Stockhammer and M. Hannuksela. 2005. H.264/AVC video for wireless transmission. *IEEE Wireless Commun.* 12 (Aug. 2005), 6–13.

[16] JM. Valin, K. Vos, and T. Terriberry. 2012. *Definition of the Opus Audio Codec.* RFC 6716. RFC Editor. https://tools.ietf.org/html/rfc6716

[17] J. Wang and D. Katabi. 2010. *ChitChat: Making Video Chat Robust to Packet Loss.* Technical Report. MIT Computer Science and Artificial Intelligence Lab (CSAIL).