

Truncated tree codes for streaming data: Infinite-memory reliability using finite memory

Stark C. Draper

ECE Dept., University of Wisconsin
Madison, WI, 53076
sdraper@ece.wisc.edu

Ashish Khisti

ECE Dept., University of Toronto
Toronto, ON, M1B 5P1
akhisti@comm.utoronto.ca

Abstract—We present a finite-memory code construction for streaming data systems. In our model of a streaming data system, a sequence of independent and identically distributed messages arrives at the transmitter according to a deterministic arrival process. Each message must be estimated by the decoder after a fixed delay. Prior work on this model relied on the use of semi-infinite tree-codes with growing encoder and decoder memory. We show that the same reliability that was attained in those constructions, which was based on an error-exponent analysis, can also be obtained by a finite-memory construction. In our construction both encoder and decoder have finite memory, although the instantaneous constraint length of the code (and of the decoding process) is time-varying in a periodic manner. The closer to capacity one wants to operate, the greater the memory our construction requires to match the infinite-memory results. For a given rate and delay it is straightforward to solve for the memory required to attain the same reliability as the earlier strategies.

I. INTRODUCTION

Shannon’s model of communication is block-oriented. The full data message is available for transmission at time zero and the physical channel is only available for some fixed number of channel uses. In contrast, many modern applications are *streaming* in nature where a sequence of data messages is realized at the transmitter in real time, each of which must be delivered to the decoder within a fixed delay. Building suitable information theoretic models, understanding fundamental limits, and characterizing good engineering architectures of such systems poses novel and intriguing challenges.

The present paper studies delay constrained streaming over discrete memoryless channels (DMCs). The transmitter observes a sequence of independent and identically distributed messages. The transmitted signal produced by our streaming encoder can only be a causal function of the observed messages. The delay-constrained decoder is required to output each source message within some fixed delay.

The most closely related literature is the work by Sahai and his co-authors [1]–[3] on delay-universal streaming. These works investigate error exponent behavior for streaming systems, motivated by connections to control over communication channels. In those papers a tree code with a growing constraint length underlies the achievability results. However, the memory requirements (both at encoder and decoder) of such a code would eventually become untenable. Our immediate motivation came from our recent work [4] wherein we

fully characterize the diversity-multiplexing tradeoff (DMT) for streaming data over wireless channels. Subsequent to publication of [4] we developed finite-memory strategies that obtain the whole DMT. We decided to revisit the DMC setting to see whether similar strategies would work. We note that the finite-memory solution for the DMT is quite simple compared to the strategy presented herein.

Finally delay constrained streaming over erasure channels has been studied in [5]–[7]. These works also consider a model involving a streaming source and a delay constrained decoder, but the underlying channel is an erasure channel with a certain bust-model of erasures. A non-zero *streaming capacity* exists for such models and is achieved by a certain class of structured code constructions.

II. MODEL

We consider streaming codes for DMCs with finite input and output alphabets \mathcal{X} and \mathcal{Y} . A sequence of independent and identically distributed (i.i.d.) messages $\{w_k\}_{k \geq 0}$ are observed at the transmitter, one per channel use. A decision on each message must be made by the decoder after a delay of $T - 1$; i.e., the decoder estimates w_k at time $T_k = k + T - 1$.

Definition 1 (Streaming Code): A delay- T , rate- R , memory- M streaming code, $\mathcal{C}(R, T, M)$, consists of

1. A sequence of messages $\{w_k\}_{k \geq 0}$ each distributed uniformly over the set $\mathcal{I} = \{1, 2, \dots, 2^R\}$.
2. A sequence of encoding functions $\mathcal{F}_k : \mathcal{I}^M \rightarrow \mathcal{X}$,

$$x_k = \mathcal{F}_k(w_{k-M+1}, \dots, w_k), \quad k = 0, 1, \dots, \infty \quad (1)$$

that map a window of recent messages to channel inputs.

3. A sequence of decoding functions $\mathcal{G}_k : \mathcal{Y}^{2M} \rightarrow \mathcal{I}$ that outputs message \hat{w}_k based on the past $2M$ observations:

$$\hat{w}_k = \mathcal{G}_k(y_{k+T-2M}, \dots, y_{k+T-1}), \quad k = 0, 1, \dots, \infty. \quad (2)$$

Note that the parameter M is the (maximum) memory of the encoding function and we will always have $M \geq T$. The decoding function has *twice* the memory of the encoding function. For simplicity of presentation and analysis we ignore integer requirements on the message set \mathcal{I} . This is not too great a restriction as one can re-normalize the time axis to correspond to the time scale of the source process – voice packets, video frames, control information, etc. – and redefine the DMC accordingly.

III. MAIN RESULT

Our main result is the following.

Theorem 1: For any DMC with random coding error exponent $E_r(\cdot)$, there exists a delay- T , rate- R , memory- M streaming code, $\mathcal{C}(R, T, M)$ that satisfies

$$\Pr[\hat{w}_k \neq w_k] \leq 2^{-TE_{st}(R)}$$

for every k , where

$$E_{st}(R) = \min \left\{ \frac{M - T + 1}{T} E_r \left(\frac{M}{M - T + 1} R \right), E_r(R) \right\}. \quad (3)$$

The bound in the above theorem is calculated with respect to the random code ensemble as well as to the message process and channel realization. We note now, and discuss in depth in Sec. V-B, that for M chosen sufficiently large the second term in (3) dominates, resulting in the same exponent as was found in the earlier (non-finite-memory) works.

IV. CODE CONSTRUCTION

We first provide details of the code construction and then analyze the performance of the proposed scheme. The central idea is to use a code with a time-varying constraint length. The constraint length cycles in a periodic manner. As it gets longer one can make more reliable decisions about earlier messages, decoding them reliably and stripping them off, thereby reducing the rate of later decisions. Our decoder is thus a decision-directed decoder and in this aspect is similar to the construction of [4]. However, by allowing the constraint length to get small, we mitigate the dependence across time. This bottlenecks the influence of earlier bad events, such as a long atypical sequence, on later decoding decisions. This allows us to reinitialize the coding process every M blocks. We refer to each decoding period, of length $2M$ as an “epoch”. This is the central idea of our construction – to limit the influence of earlier uncertain messages on our current decision, thereby mitigating the possibility of error propagation in the decoding process.

Perhaps most natural way to think about our scheme is in analog to convolutional codes. Each transmitted symbol corresponds to one symbol of the convolutional code while each message corresponds to one information symbol shifted into the code. In Figure 1 we plot the dependence between input information symbols and outputted channel symbols for a (finite-impulse response) convolutional code with a fixed constraint-length of 3. Each row correspond to a message index and each column to a channel. The first dot in row k corresponds to the first channel use that is affected by message w_k . Due to the causal nature of the encoder the index of this row must be at least k . The last dot in row k corresponds to the last channel use affected by message w_k . For a convolution code with constant constraint length 3, this is $k + 2$.

Figure 2 presents the analogous plot for one of our $\mathcal{C}(R, 3, 7)$ codes, which we will use to understand the time-varying constraint length of the code. For a $\mathcal{C}(R, 3, 7)$ code

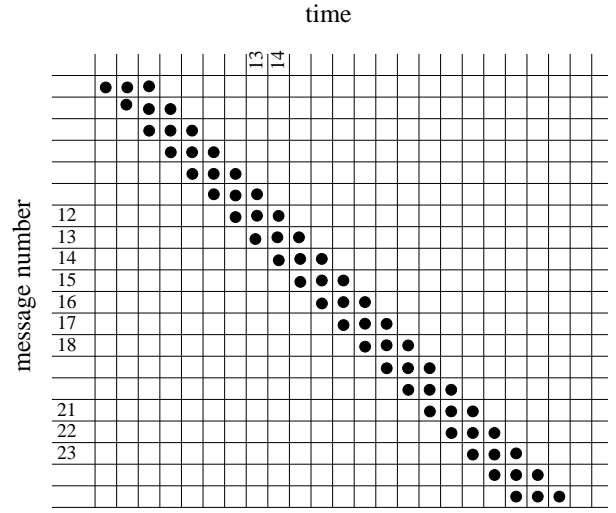


Fig. 1. A convolutional code with constraint length 3: each input message effects the channel input in the channel use in which it arrived as well as the following two in addition.

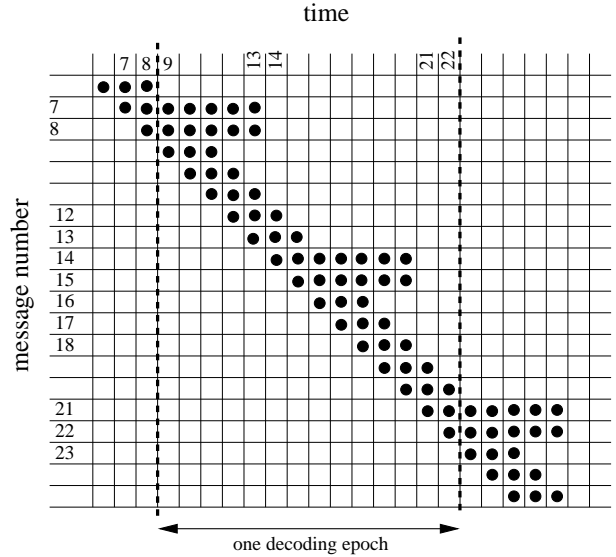


Fig. 2. A streaming code $\mathcal{C}(R, T = 3, M = 7)$: each message must be decoded after a delay of $T = 3$, a periodic subset of the messages, occurring every $M = 7$ messages, effects the channel inputs for longer. These messages can be decoded more reliably, thus allowing the decoding process to be reinitialized every M messages / channel uses. Each re-initialization corresponds to a new decoding “epoch”, one of which is shown. The epoch shown is of duration $2M = 14$, extending from time $M + T - 1 = 9$ to $3M + T - 2 = 22$. This is the epoch considered when decoding messages 14 through 20.

every message influences the output for at least $T = 3$ symbols, with every M th message having the longest-lasting influence, of $M = 7$ symbols. The encoding pattern is periodic with period $M = 7$ and in the figure we indicate one decoding epoch of length $2M = 14$, extending from channel input 9 through 22. We use this figure to illustrate the code construction and decoding algorithm, detailed in the following subsections.

A. Encoder

Our proposed streaming code, $\mathcal{C}(R, T, M)$, consists of a semi-infinite sequence of codewords $\{x_0, x_1, \dots, x_k, \dots\}$, where x_k is the k th channel input when messages (w_0, \dots, w_k) have already been realized. Due to the finite-memory of the encoder only a subset of these messages are involved in determining x_k .

To understand the following, rather involved, definitions of the time-varying nature of our code, we refer closely to the example given in Figure 2. We first define $\tau_{e,k}^+$, a function of the source symbol index k , to be the index of the *last* channel use that is a function of w_k . In terms of Figure 2 this is the index of the last non-zero column in row k .

We define $\tau_{e,k}^+$ to be

$$\tau_{e,k}^+ = \begin{cases} k - k \bmod M + M - 1 & \text{if } 0 \leq k \bmod M \leq T - 2 \\ k + T - 1 & \text{else} \end{cases}. \quad (4)$$

Note that the quantity $k - k \bmod M$ takes you to some integer multiple of M , to which the maximum memory is added. The end result is an interval of $T - 1$ messages whose influences on the channel inputs all terminate at a specific time, yielding the stair-step nature of the plot in Figure 2.

We refer to the quantity $\tau_{e,k}^+ - k$ as the ‘‘constraint length’’ of the code, which we note is time-varying and is at most M . Referring to the example of Figure 2, the constraint lengths of messages 6 and 9 through 13 is $T = 3$. Meanwhile $\tau_{e,7}^+ = \tau_{e,8}^+ = 13$ with message 7 having the maximal constraint length $M = 7$ and message 8 having constraint length 6.

When computing the k th channel input x_k , we also need to know the index of the earliest message in the domain of \mathcal{F}_k . This corresponds to the index of the first non-zero row in column k . We define this index to be $\tau_{e,k}^-$ where

$$\tau_{e,k}^- = \begin{cases} k - k \bmod M & \text{if } T - 2 \leq k \bmod M \leq M - 1 \\ k - T + 1 & \text{else} \end{cases}.$$

Since $k - \tau_{e,k}^- + 1 \leq M$, the encoder is finite-memory, justifying the definition of \mathcal{F}_k in (1). We note that the dimension of the input is at most M , and is often strictly smaller. Again we refer to Figure 2 for a concrete example. Here $\tau_{e,14}^- = 14 - T + 1 = 12$ and $\tau_{e,15}^- = 15 - T + 1 = 13$. On the other hand $\tau_{e,16}^- = \tau_{e,17}^- = \dots = \tau_{e,20}^- = 14$.

The sequence transmitted up to, and including, channel input k is denoted by

$$x_0^k(w_0^k) \triangleq \left\{ x_0(w_0), x_1(w_{\tau_{e,1}^-}^1), \dots, x_k(w_{\tau_{e,k}^-}^k) \right\}, \quad (5)$$

so, $x_0^k(w_0^k) \in \mathcal{X}^{(k+1)M}$.

We use an i.i.d. random code construction. The channel input at time k , for some $w_{\tau_{e,k}^-}^k = w_{\tau_{e,k}^-}^k$, $x_k(w_{\tau_{e,k}^-}^k)$, is chosen independently of all previous codeword symbols according to some input distribution. Further, for all $\tilde{w}_{\tau_{e,k}^-}^k \neq w_{\tau_{e,k}^-}^k$ the symbols $x_k(\tilde{w}_{\tau_{e,k}^-}^k)$ and $x_k(w_{\tau_{e,k}^-}^k)$ are independent. Finally, to make these definitions well defined for the first M channel symbols, we define known ‘‘dummy’’ messages w_{1-M}, \dots, w_{-1} that can be correctly stripped off by the decoder with probability one.

B. Decoder

Our decoder has a functional form that is periodic, resetting every M channel uses. Each such interval is one decoding epoch. When decoding message w_k at time $T_k = k + T - 1$ our decoder computes an estimate of a finite window of previous messages: messages $w_{\tau_{d,k}}, \dots, w_k$. The time $\tau_{d,k}$ indexes the start of the relevant decoding epoch and corresponds to the earliest channel observation relevant to estimating w_k .

We define $\tau_{d,k}$ to be

$$\tau_{d,k} = k - k \bmod M - M + T - 1. \quad (6)$$

Note that $T_k - \tau_{d,k} + 1 = M + 1 + k \bmod M \leq 2M$, so this is a finite-memory decoder. For any message w_k that corresponds to a particular $\tau_{d,k}$, the estimate of w_k is computed based on the received sequence $y_{\tau_{d,k}}^{T_k} = (y_{\tau_{d,k}}, \dots, y_{T_k})$.

As an example, for the $\mathcal{C}(R, 3, 7)$ code depicted in Figure 2 $\tau_{d,k} = 9$ for $14 \leq k \leq 20$, while $\tau_{d,k} = 16$ for $21 \leq k \leq 27$. The former corresponds to the epoch indicated in Figure 2. Interpreting (6) we see that $k - k \bmod M$ backs us up to the beginning of the encoding interval, just as in (4). Then we back up one more interval of M messages. Finally, we add the offset $T - 1$ to avoid having to consider the effect of messages prior to message $k - k \bmod M - M$.

At time T_k , the decoder sequentially makes estimates of messages w_l for $l = \tau_{d,k}, \dots, k$. It first estimates message $w_{\tau_{d,k}}$, and proceeds to decode the others in order. Each estimate is a maximum likelihood (ML) estimate, condition on the estimates of earlier messages and the vector of observations $y_l, \dots, y_{\min\{\tau_{e,l}^+, T_k\}}$. The minimization $\min\{\tau_{e,l}^+, T_k\}$ is included as you may need to decode message k prior to the influence of w_l on channel inputs having ended, cf. Figure 2 and consider messages w_{15} through w_{17} .

Say that we are at step l and have estimated messages $\tau_{d,k}$ through $l - 1$ as $\bar{w}_{\tau_{d,k}}^{l-1}$. With the estimates of those messages fixed, there is uncertainty in the later messages, w_l, \dots, w_{T_k} . The decoder searches for the vectors of messages $\hat{w}_l^{\min\{\tau_{e,l}^+, T_k\}}$ that maximize the probability of the observations $y_l^{\min\{\tau_{e,l}^+, T_k\}}$ with respect to the codeword suffix

$$x_l^{\min\{\tau_{e,l}^+, T_k\}} \left(\bar{w}_{\tau_{d,k}}^{l-1}, \hat{w}_l^{\min\{\tau_{e,l}^+, T_k\}} \right). \quad (7)$$

If the ML codeword suffix is unique, the decoder concatenates \hat{w}_l with $\bar{w}_{\tau_{d,k}}^{l-1}$ to get $\bar{w}_{\tau_{d,k}}^l$. If it is not unique, but the message sequences corresponding to all the ML message suffixes share the same (unique) element \hat{w}_l , then the decoder also concatenates \hat{w}_l with $\bar{w}_{\tau_{d,k}}^{l-1}$ to get $\bar{w}_{\tau_{d,k}}^l$, otherwise an error is declared. When the process continues without declaring an error until an estimate of message w_k is made, then \bar{w}_k is declared to be the output message.

Both encoding and decoding functions are periodic in k . To simplify the analysis of our strategy we concentrate on a single decoding epoch, extending from $\tau_{d,k}$ to T_k . We re-index these time sample to extend from 0 to $2M - 1$, and the messages similarly. E.g., if we were to consider the epoch depicted in Figure 2, corresponding to messages w_{14}, \dots, w_{20} , we would

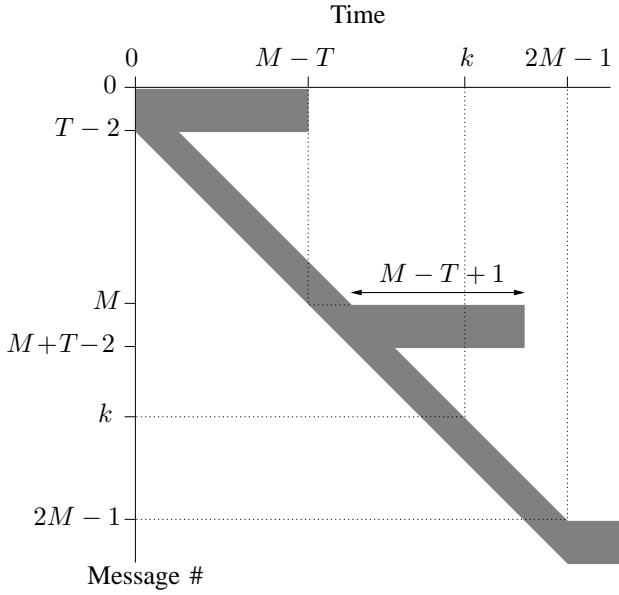


Fig. 3. A graphical depiction of the dependence between message number and time. This is an analog of Fig. 2, showing one decoding epoch. The shaded areas indicate the dependencies between messages and channel inputs. All message and time indices have been shifted by $jM + T - 1$ where j is the index of the decoding epoch, $j = -1, 0, 1, \dots$, hence message index and time are enumerated starting with 0. (The epoch $j = -1$ corresponds to decoding the first M messages where the dummy messages are shifted in.) The second half of the messages $M \leq k \leq 2M - 1$ are decoded by considering only the observations between 0 and $2M - 1$.

subtract $\tau_{d,k} = 9$ from each message number and time index. Figure 3 depicts a generic decoding epoch after re-indexing. After the re-indexing we discuss decoding message w_k for $M \leq k \leq 2M - 1$, i.e., the second half of the messages depicted in the figure (corresponding to messages 14 through 20 in Figure 2). The first $M - T + 1$ estimates (corresponding to messages 9 through 13), of re-indexed messages are “helper” estimates. We make these estimates to reinitialize the decoding process. The last $T - 1$ (corresponding to messages 21 and 22) can’t be decoded reliably.

V. ANALYSIS

In this section we analyze the probability of error in Sec. V-A. In Sec. V-B, provide design guidance on how to choose M as a function of T and the channel under consideration. We find that, for a large enough choice of M , the same exponent as was previously found for semi-infinite tree codes (whose constraint length increases with time) is achievable.

A. Error analysis

As mentioned above, since our decoder is periodic, resetting every M channel uses, without loss of generality we focus on a single decoding “epoch”. This allows us to simplify notation, indexing time from the first channel use in the epoch, indexed by zero, and consider the decoding of message w_k where $M \leq k \leq 2M - 1$. Figure 3 depicts the channel uses and messages of interest in a single epoch.

$$\begin{aligned} \Pr[\bar{w}_k \neq w_k] &\leq \Pr[\bar{w}_0^k \neq w_0^k] \\ &= \Pr[(\bar{w}_0^{T-2} \neq w_0^{T-2}) \cup (\cup_{l=T-1}^k (\bar{w}_0^{l-1} = w_0^{l-1} \cap \bar{w}_l \neq w_l))] \\ &\leq \Pr[\bar{w}_0^{T-2} \neq w_0^{T-2}] + \sum_{l=T-1}^k \Pr[\bar{w}_l \neq w_l | \bar{w}_0^{l-1} = w_0^{l-1}] \end{aligned} \quad (8)$$

By the randomness of the code construction we can assume that $w^{T_k} = 0^{T_k R}$.

We start by bounding the first term in (8). These messages effect the input of the channel only in the interval $0 \leq t \leq M - T$. Therefore the decoding of these messages is only a function of y_0, \dots, y_{M-T} , cf. Fig. 3. To make notation more compact we introduce the set

$$\begin{aligned} \mathcal{I}(y_l^{l'}) &= \{\tilde{w}_k^{k'} : p_{y^{l'-l+1}|x^{l'-l+1}}(y_l^{l'} | x_l^{l'}(\tilde{w}_k^{k'})) \\ &\geq p_{y^{l'-l+1}|x^{l'-l+1}}(y_l^{l'} | x_l^{l'}(0^{(k'-k+1)R}))\}, \end{aligned} \quad (9)$$

where we make sure that l, l' and k, k' are chosen in a compatible way. We note that the set $\mathcal{I}(y_l^{l'})$ is also a function of the codebook, a dependence we have suppressed. We do indicate the fact that the codebook is random via the random vectors $x_l^{l'}(\tilde{w}_k^{k'})$ and $x_l^{l'}(0^{(k'-k+1)R})$. We express the error as

$$\begin{aligned} \Pr[\bar{w}_0^{T-2} \neq 0^{(T-1)R}] &= \\ \Pr[\exists \tilde{w}_0^{M-1} \text{ s.t. } \tilde{w}_0^{T-2} \neq 0^{(T-1)R} \cap (\tilde{w}_0^{M-1} \in \mathcal{I}(y_0^{M-T}))], \end{aligned} \quad (10)$$

and (note the observation length is $M - T + 1$. We see that the cardinality of the set of misleading message sequences

$$\left| \left\{ \tilde{w}_0^{M-1} \text{ s.t. } \tilde{w}_0^{T-2} \neq 0^{(T-1)R} \right\} \right| = 2^{MR} - 2^{(M-T)R} \leq 2^{MR}. \quad (11)$$

To decode correctly, a message with the true message prefix $(0^{(T-1)R})$ must be distinguished from roughly 2^{MR} message sequences with the incorrect prefix. The error probability can be bounded using standard bounding techniques from error exponents of random coding for block codes where the block length is $M - T + 1$ and the rate is $\frac{M}{M-T+1}R$, giving

$$\Pr[\bar{w}_0^{T-2} \neq 0^{(T-1)R}] \leq 2^{-(M-T+1)E_r(\frac{M}{M-T+1}R)} \quad (12)$$

where $E_r(R)$ is the random coding error exponent.

All remaining messages w_l , $T - 1 \leq l \leq k$ we decode conditionally, given the previous decisions \bar{w}_0^{l-1} and the observations $y_l^{T_l}$, where $T_l = l + T - 1$. Note that some messages, e.g., w_M affect channel inputs (and outputs) beyond time $M + T - 1$. However, while taking those outputs into account would increase the reliability of those decoding decisions, ignoring them simplifies the analysis and still meets the reliability of the dominant error events of our scheme.

$$\begin{aligned} \Pr[\bar{w}_l \neq 0^R | \bar{w}_0^{l-1} = 0^{lR}] &= \Pr[\exists \tilde{w}_{\tau_{e,l}}^{T_l} \text{ s.t.} \\ &\tilde{w}_{\tau_{e,l}}^{l-1} = 0^{(l-\tau_{e,l})R} \cap \tilde{w}_l \neq 0^R \cap (\tilde{w}_{\tau_{e,l}}^{T_l} \in \mathcal{I}(y_l^{T_l}))]. \end{aligned} \quad (13)$$

Given the conditioning, the only messages in $\tilde{w}_{\tau_{e,l}}^{T_l}$ that can be in error start from message l . The cardinality of the set of misleading message suffixes $\tilde{w}_l^{T_l}$ is upper bounded by $2^{(T_l-l+1)R} = 2^{TR}$. In contrast to (11) we now have $T_l - l + 1 = T$ observations and 2^{TR} messages. Drawing again upon standard error analysis for random codes we have

$$\Pr[\bar{w}_l \neq 0^R | \bar{w}_0^{l-1} = 0^{lR}] \leq 2^{-TE_r(R)}. \quad (14)$$

Substituting these bounds into (8) and noting that $k < 2M$,

$$\Pr[\bar{w}_k \neq w_k] \leq 2^{-(M-T+1)E_r(\frac{M}{M-T+1}R)} + M2^{-TE_r(R)}. \quad (15)$$

B. Choosing the system memory M

To minimize the error probability in (15) for fixed delay and fixed rate, we must choose M to make the two terms of (15) equal. Thus we solve for M to make the following identity hold:

$$(M - T + 1)E_r\left(\frac{M}{M - T + 1}R\right) = TE_r(R). \quad (16)$$

Clearly the optimizing M is a function of T , R , and the channel law. But one can immediately infer some properties. For any $R < C$ the right-hand side is positive. For the left-hand side to be positive, M must be chosen large enough that $\frac{M}{M-T+1}R < C$, or

$$M > \frac{(T-1)C}{C-R}, \quad (17)$$

and we define $M^* = \lceil (T-1)C/(C-R) \rceil$. The penalty to pay for our finite-memory decoding is that whenever we “reset” the decoder by entering a new decoding epoch, there is some additional rate overhead that we must amortize out. As long as the memory exceed M^* the effect of this overhead is below capacity.

To solve for the memory M for (16) we reexpress the relationship as

$$\left(\frac{M}{T} - 1 + \frac{1}{T}\right) E_r\left(\frac{1}{1 - \frac{T}{M} + \frac{T}{M} \frac{1}{T}}R\right) = E_r(R).$$

For channels such as the binary symmetric channel (BSC), for which we present results, the quantity $1/T$ will be small. So, we approximate (16) by dropping the term terms modulated by $1/T$. We note that for packet-type channels this approximation may not be appropriate. With the approximation we are left with

$$\left(\frac{M}{T} - 1\right) E_r\left(\frac{1}{1 - \frac{T}{M}}R\right) = E_r(R). \quad (18)$$

Of course, for (18) to be well defined the ratio M/T must be the ratio of two integers. If T is large, then such integer effects will not be significant and we ignore them in the following.

Solving for the ratio M/T numerically, we plot the results in Figure 4 when the channel of interest is the BSC with crossover probability 0.11. We note that at low rates the increased in rate overhead incurs a small memory overhead

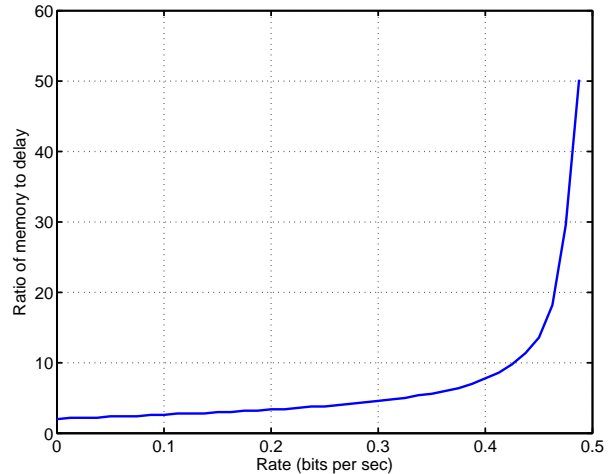


Fig. 4. Memory for the finite-memory construction to attain the same error exponent as the infinite memory construction over the binary symmetric channel with crossover probability 0.11 at various rates. The capacity of this channel is 0.5 bits per channel use. We plot the ratio M/T for the minimum ratio that satisfies (18).

to amortize it out completely (regaining the infinite-memory results). In particular, in the low-rate extreme of the Figure, the ratio $M/T = 2$. In contrast, at high rates the overhead becomes more expensive. Of course, in operation one may not need to recover the infinite-memory exponent.

VI. CONCLUSIONS

We studied the problem of delay-constrained streaming over a DMC. We present a finite-memory construction that can recapture the error exponent behavior of previous semi-infinite tree constructions given sufficiently large, but finite, memory. The associated decoder involves a decision-directed periodically time-varying decision rule. The time-varying and finite-memory nature of the construction prevents error propagation. These results pose obvious next steps in the design of low-complexity coding-theoretic constructions that have the same behavior.

REFERENCES

- [1] A. Sahai and S. Mitter, “The necessity and sufficiency of anytime capacity for stabilization of a linear system over a noisy communication link Part I: Scalar systems,” *IEEE Trans. Inform. Theory*, vol. 52, pp. 3369–3395, Aug. 2006.
- [2] A. Sahai, “Anytime information theory,” Ph.D. dissertation, Mass. Instit. of Tech., 2001.
- [3] S. C. Draper and A. Sahai, “Universal anytime codes,” in *Control over Commun. Channels Workshop, 5th Int. Symp. Modeling and Optimization in Mobile, Ad-Hoc, and Wireless Networks*, Limassol, Cyprus, Apr. 2007.
- [4] A. Khisti and S. C. Draper, “Streaming data over fading wireless channels: The Diversity-Multiplexing Tradeoff,” in *Proc. Int. Symp. Inform. Theory*, St. Petersburg, Russia, 2011.
- [5] E. Martinian and G. W. Wornell, “Universal codes for minimizing per-user delay on streaming broadcast channels,” in *Proc. Allerton Conf. Commun., Contr., Computing*, Monticello, IL, Sep. 2003.
- [6] E. Martinian and M. D. Trott, “Delay-optimal burst erasure code construction,” in *Proc. Int. Symp. Inform. Theory*, Nice, France, Jun. 2007.
- [7] A. Badr, A. Khisti, and E. Martinian, “Diversity embedded streaming erasure codes (DE-SCo): Constructions and optimality,” *IEEE J. Select. Areas Commun.*, May 2011.