Data Synchronization with Timing: The Variable-Rate Case

Navin Kashyap and David L. Neuhoff

Abstract

This paper extends the theory of data synchronization with timing for fixed-rate codes, previously developed by the authors, to the variable-rate case. Given a source code, a class of sync-timing codes called variable-rate cascaded (VRC) codes is considered that "wrap around" the source code in such a way as to enable the decoder to not only resynchronize rapidly when the encoded bits are corrupted by insertion, deletion or substitution errors, but also produce estimates of the time indices of the data symbols encoded by the source code. The estimates of the time indices are modulo-T reductions of the actual time indices, for some integer T called the timing span of the code. These sync-timing codes are analyzed on the basis of the maximum timing span achievable for a given coding rate R and permissible resynchronization delay D. It is shown that the timing span of VRC codes is upper bounded by $2^{D(1-R)+o(D)}$, and that this upper bound is achievable asymptotically in D. This exponential rate of growth of timing span with delay is the same as that found previously for certain fixed-rate sync-timing codes, *e.g.* (fixed-rate) cascaded codes.

Index Terms

cascaded codes, resynchronization delay, sync-timing codes, timing span, variable-rate cascaded (VRC) codes

Manuscript originally submitted September 2001, revised April 2004; re-submitted June 2007. This work was supported in part by NSF Grants NCR-9415754 and CCR-9815006, and was carried out while the first author was at the University of Michigan. Portions of this work were published in the proceedings of the IEEE International Symposium on Information Theory, Sorrento, Italy, June 2000.

Navin Kashyap is with the Department of Mathematics and Statistics at Queen's University, Kingston, ON K7L 3N6, Canada. (email: nkashyap@mast.queensu.ca).

David L. Neuhoff is with the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI 48109, USA. (email: neuhoff@eecs.umich.edu).

I. INTRODUCTION

An important aspect of communications that has not received much serious attention in the information theory literature pertains to the encoding and decoding of the time indices of the transmitted data. The transmission of time indices of the data, along with actual data values themselves, is necessary in most situations where conventional data synchronization (*cf.* [1],[2],[3]) is needed. A typical such situation is when a sequence of data symbols must be encoded into bits and transmitted across a channel that can make arbitrary insertion, deletion and substitution errors. A particularly pertinent scenario is when decoding commences in the middle of the transmitted stream, as happens, for example, when a television set is switched on. From the standpoint of the receiver, this may be viewed as the deletion of an arbitrary prefix of the encoded bitstream. It should be noted that while insertion and deletion errors can cause synchronization problems regardless of whether the encoding of the data is fixed-length or variable-length, substitution errors can cause loss of synchronization only with variable-length encoding of the data. Thus, variable-length source codes are particularly susceptible to synchronization errors.

A typical data synchronization scheme consists of a mechanism that enables the decoder to re-establish synchronization, usually by locating the start of a valid codeword in the encoded sequence, when there is a loss of synchronization caused by channel errors. Re-establishment of synchronization means that in the absence of further errors, some suffix of the sequence of data symbols produced by the decoder agrees with some suffix of the original (unencoded) sequence of data symbols. There is a long history of such synchronization schemes, primarily for variable-length codes, in the literature [4]–[11].

However, in order for the recovery of synchronization to be useful in practice, it is equally important for the decoder to have a means of establishing the positions of the decoded symbols relative to the original data sequence. In other words, synchronization schemes must not only enable the decoder to resynchronize rapidly upon cessation of channel errors, but also must allow the decoder to produce estimates of the time indices of the decoded data symbols. Synchronization, when achieved along with timing recovery, is known as *strong synchronization* [12],[13] or *synchronization with timing* [14],[15].

A good motivation for synchronization with timing can be given in the context of multimedia communications. In multimedia broadcasts, we may have multiple, related data streams being transmitted simultaneously, for example, audio and video streams in videoconferencing. If one of the streams gets affected by synchronization errors, and there is no mechanism for timing recovery, then conventional synchronization within each stream alone will not prevent a mismatch between the streams. This can result in the undesirable situation of sound not corresponding to video. Indeed, the importance of synchronization

with timing has been recognized since the early days of image/video coding standards. For example, there is a provision made within the JPEG/MPEG standards for the introduction of synchronizing markers, *e.g.*, end-of-line and end-of-frame markers, at regular intervals in the encoded sequence as a means of encoding the time indices of the data. Other examples illustrating the need for synchronization with timing can be found in [14] and [16].

Coding schemes that encode and decode data time indices, as well as data values, will henceforth be referred to as *synchronization with timing codes*, or *sync-timing codes* for short. There has been some recent interest in the design and performance analysis of sync-timing codes. A clear identification of the need for sync-timing in the context of synchronization of variable-rate source codes seems to have first appeared in the literature in [12]. This notion was further developed in [13], and an analysis of the relative merits of various sync-timing codes, assuming a probabilistic channel model, can be found in [16]. A theoretical framework for the performance analysis of sync-timing codes that are fixed-rate, in the sense that there are integers i and j such that j bits emerge from the encoder for every i bits that enter, was developed in [14] and [15]. The performance of several families of such codes were analyzed within this framework.

In this paper, we use a family of variable-rate sync-timing codes, which we call *variable-rate cascaded (VRC) codes*, to extend the theory developed in [14] to the variable-rate case. A VRC code is designed to "wrap around" a source code, as described in Section II. The decoder for a VRC code produces time index estimates of the decoded data symbols. In the absence of channel errors, these time index estimates are simply modulo-T reductions of the actual time indices, where T is some (typically, large) integer. Coding schemes similar to these codes have previously been considered; indeed, what we call a VRC code is essentially the coding scheme referred to as "Scheme A" in [16] used in conjunction with bitstuffing. What is new in this paper is the framework under which the theoretical limits to the performance of such coding schemes can be analyzed. Our framework and analysis differ fundamentally from the analysis of the probability of loss of strong synchronization for the schemes in [16].

The performance of a VRC code is measured in terms of its coding rate, resynchronization delay and timing span. Coding rate and timing span are straightforward to define, so we do that first. Coding rate is taken to be the ratio of the average number of source-coded bits entering the encoder for the VRC code to the average number of bits it produces as output in response. The timing span of a VRC code measures the ability of the code to produce estimates of the time indices of the decoded data symbols. This is nicely captured by the integer T modulo which the time index estimates are produced by the decoder for the VRC code, and so we define this integer T to be the timing span of the code.

Resynchronization delay is defined in the context of a scenario where the decoder loses synchronization due to channel errors, but the channel subsequently remains error-free for a sufficiently long period of time. In such a scenario, if the decoder re-establishes synchronization upon receiving, on an average, D bits, then D is taken to be the resynchronization delay of the code. It should be pointed out that re-establishment of synchronization now means that the decoder produces a sequence of $(\text{data-symbol}, \text{time-index-estimate})$ pairs such that the data-symbol sequence produced agrees with some suffix $\mathbf{s}$ of the original data-symbol sequence, and the corresponding time index estimates are the correct modulo-T reductions of the time indices[1] of the entries in $\mathbf{s}$. In practice, for a VRC code, the average number of bits needed to re-establish synchronization is roughly equal to the average length of a codeword. Therefore, for practical purposes, we can simply set the resynchronization delay of a VRC code to be equal to the average length of a codeword.

Henceforth, all uses of the words "rate" and "delay" of a VRC code will refer to coding rate and resynchronization delay as defined above. Note that our definitions of rate and delay both take into account the variable-rate nature of the VRC code. Explicit expressions, in terms of VRC code parameters, for the three performance measure of rate, delay and timing span will be given in Section II. We also point out that all three performance measures for a VRC code have been defined in a manner compatible with the corresponding definitions given in [14] for the case of fixed-rate sync-timing codes. This allows us to make direct performance comparisons between VRC codes and fixed-rate sync-timing codes.

Under suitable assumptions on the source code, we can derive, as we shall see in Section III, upper and lower bounds on the maximum timing span, $T_{VRC}(r, d)$, achievable by a VRC code with rate $R \geq r$ and delay $D \leq d$. These bounds imply that, for $r \in (0, 1)$ and sufficiently large d, $T_{VRC}(r, d) \sim 2^{d(1-r)}$. This shows that the timing span of variable-rate sync-timing codes can grow exponentially with resynchronization delay, just as in the fixed-rate case, and the rate of exponential growth is the same as that for fixed-rate codes.

We conclude the Introduction by emphasizing a point also made in [14]. In defining resynchronization delay above, we assumed a context in which the channel remains error-free for a sufficiently long period of time. A synchronization code (with or without a timing-recovery mechanism) by itself cannot provide satisfactory performance in situations where the channel makes errors as frequently as, say, once every D bits, D being the delay of the code. In such a situation, it is preferable to first use an error-correcting

[1]Time indices here mean the positions of entries in the sequence $\mathbf{s}$ relative to the beginning of the original data-symbol sequence, and *not* relative to the beginning of $\mathbf{s}$.

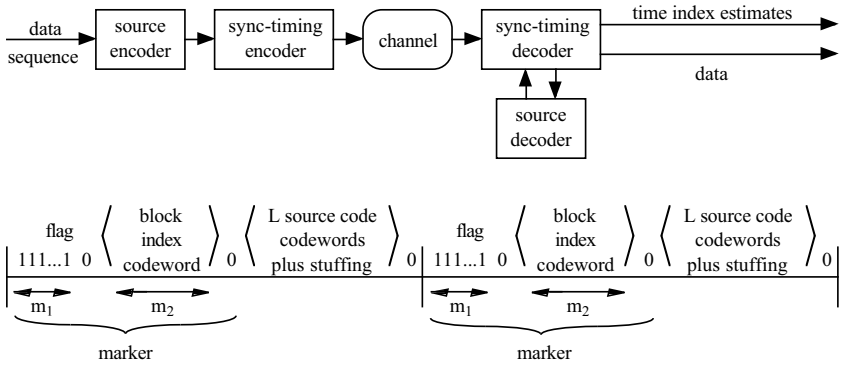

Fig. 1. Block diagram for a variable-rate cascaded code, and a part of the encoded sequence of bits.

code to create a channel that makes infrequent errors, over which a synchronization code can provide good performance. Simulation results exist in the literature [17] that demonstrate this point well.

II. VARIABLE-RATE CASCADED CODES

Consider a lossless, binary source code, parametrized by an integer $k \geq 1$ and a real number $R_s > 0$, for a data source with a discrete (countable) alphabet and an associated probability mass function. The source code is assumed to be a blocklength-k-to-variable-length prefix code with rate R_s, *i.e.*, the expected number of bits, per data symbol, produced by the source encoder is R_s. We shall further assume, for the time being, that each block of k data symbols is encoded independently of any other block. As we shall see, this assumption may be relaxed somewhat.

The encoder for a *variable-rate cascaded (VRC) code*, which we shall refer to as the *sync-timing encoder*, is designed to follow the source encoder (see Figure 1). Thus, data symbols from the source are first encoded (compressed) by the source encoder, and the output of the source encoder is then further encoded by the sync-timing encoder. The encoded bitstream is then transmitted across the channel, which can make arbitrary insertion, deletion or substitution errors. At the channel output, the received bitstream first passes through the *sync-timing decoder*, and subsequently through the source decoder.

A VRC code is characterized by positive integers p, m_1, m_2 and L, and whenever necessary, we shall refer to a VRC code with these parameters as a (p, m_1, m_2, L) VRC code. The sync-timing encoder operates on blocks of L source codewords at a time (or equivalently, on blocks of kL data symbols). In response to a block of L source codewords, the sync-timing encoder produces a *VRC codeword* that is composed of one of p distinct *markers*, followed by the block of L source codewords after it has

undergone *bitstuffing*, followed by a zero (see Figure 1). The marker consists of a *flag* of $m_1 > 1$ consecutive ones (denoted by 1^{m_1}), followed by a zero, followed by a *block index codeword*, followed by another zero. The block index codewords form a codebook of p distinct binary sequences of length m_2, none of which contains the flag 1^{m_1}. These codewords are used to encode the modulo-p block indices, *i.e.*, the VRC codeword produced in response to the jth block of L source codewords contains the block index codeword that encodes the integer $i = j - 1 \bmod p$. The integer p is referred to as the *period* of the code. Bitstuffing prevents the spurious appearance of a flag in each block of L source codewords by "stuffing" a zero immediately after each occurrence of 1^{m_1-1} in the codeword. The structure of this code is similar to that of the cascaded code described in [14], and is also similar to the coding schemes described in [16].

The sync-timing decoder locates the flags in the stream of received bits, and for each flag found, it reverses (if possible) the encoding procedure on the sequence up to the next flag. Thus, every successful reversal of the encoding yields the integer i encoded by the block index codeword, and a "destuffed" sequence of bits that is passed to the source decoder. If the destuffed sequence is a valid sequence of L source codewords, then the source decoder determines, and produces as output, the kL data symbols encoded by this block of codewords. The output of the source decoder is fed back into the sync-timing decoder. The sync-timing decoder checks to see if there are exactly kL data symbols in the sequence produced by the source decoder, in which case it assigns to the data-symbol sequence the sequence of time indices $(ikL + 1, ikL + 2, \ldots, (i + 1)kL)$. The output of the sync-timing decoder then consists of the sequence of kL data symbols and the time indices assigned to them.

Note that since $0 \leq i \leq p - 1$, the time indices produced by the sync-timing decoder are modulo-pkL reductions of the actual time indices of the data symbols. Therefore, by the definition given in Section I, the timing span of the VRC code is given by

$$T = pkL \tag{1}$$

We next derive expressions for the rate and delay of a VRC code in terms of the parameters of the source code and the VRC code. We will find it useful to first derive an expression for delay. From our description of decoding above, it should be clear that in a scenario where the channel remains error-free for a sufficiently long period of time after initial loss of decoder synchronization, the decoder will re-establish synchronization once it locates a flag in the error-free received bitstream. Therefore, the number of received bits that the decoder may erroneously decode or discard is (at worst) equal to the length of a VRC codeword. Therefore, in keeping with the definition of delay given in Section I, it makes sense

to set the resynchronization delay, D, of a VRC code to be equal to the average length, $\overline{\mathcal{N}}$, of a VRC codeword. We thus have

$$D = \overline{\mathcal{N}} = m_1 + m_2 + 3 + kLR_s + \overline{S} \tag{2}$$

where $\overline{S}$ is the expected number of stuffed bits in a VRC codeword.

Finally, the coding rate, R, is the ratio of the average number of bits entering the sync-timing encoder to the average number of output bits. Thus,

$$R = \frac{kLR_s}{\overline{\mathcal{N}}} = 1 - \frac{m_1 + m_2 + 3 + \overline{S}}{\overline{\mathcal{N}}} \tag{3}$$

Clearly, a good VRC code must have rate close to 1, small delay and large timing span. The parameters p, m_1, m_2 and L of the VRC code are typically chosen so as to maximize $T = pkL$, given a desired coding rate and resynchronization delay. For a given target rate of $r \in (0, 1)$ and resynchronization delay of d, a good choice of parameters is as follows:

$$\begin{aligned} p &= \left\lfloor \frac{R_s}{dr}\, 2^{d(1-r)-\gamma(r,k,R_s)-\epsilon} \right\rfloor, \\ m_1 &= \lceil \log_2 \log_2 p \rceil + \alpha(r), \\ m_2 &= \lceil \log_2 p \rceil + 1, \text{ and} \\ L &= \left\lceil \frac{dr}{kR_s} \right\rceil, \end{aligned}$$

where $\gamma(r, k, R_s) = 2^{1-\alpha(r)} \frac{r}{1-r} + \alpha(r) - \log_2 \frac{r}{1-r} + kR_s + \log_2 R_s + 6$, and $\alpha(r) = \max(0, \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor)$. For values of r close to 1, $\alpha(r) \approx \log_2 \frac{2r}{1-r}$, and consequently, $\gamma(r, k, R_s) \approx kR_s + \log_2 R_s + 7$. Thus, for values of r close to 1, the expressions for p and m_1 above are well approximated by

$$\begin{aligned} p &= \left\lfloor \frac{1}{dr}\, 2^{d(1-r)-kR_s-7} \right\rfloor, \text{ and} \\ m_1 &= \lceil \log_2 \log_2 p \rceil + \log_2 \frac{2r}{1-r}. \end{aligned}$$

As we shall see in Section III, for $r \in (0, 1)$ and d sufficiently large, a VRC code with the choice of parameters given above has timing span provably close to the maximum theoretically achievable by codes with rate at least r and delay at most d.

We briefly remark upon the reasons for choosing the all-ones sequence, 1^{m_1}, as the flag. Firstly, it has been shown [18],[19] that the bitstuffing procedure for the all-ones flag is the most efficient in the following sense. Given a length-n IID random sequence of equiprobable bits, let $I_n(A)$ be the expected number of "stuffed" bits inserted to prevent the sequence $A = a_1 a_2 \ldots a_m$ from occurring within the random sequence (the complement of a_m is inserted whenever $a_1 a_2 \ldots a_{m-1}$ is observed). Then, among

all sequences A of fixed length m, the all-ones sequence 1^m minimizes the asymptotic average redundancy, $\lim_{n\to\infty} I_n(A)/n$. Secondly, there is a simple and efficient algorithm [20] for generating the set of all length-m_2 sequences (for the block index codebook) that do not contain m_1 consecutive ones.

It is also worth pointing out the reasons for the assumptions made on the source code at the beginning of this section. The source code was taken to be a prefix code so as to enable the sync-timing encoder to parse the source-coded bitstream into codewords, which would allow it to form blocks of L source codewords for further encoding. We had also assumed that each block of k data symbols was encoded independently of any other block. The reason for this is best illustrated as follows. Suppose that the encoding of the $(L+1)$th block of k data symbols depends on some of the first kL data symbols. Note that the VRC codeword that contains the first L source codewords is different from that which contains the $(L+1)$th source codeword. Now, suppose that the VRC encoded bitstream was corrupted by channel errors in such a way that the sync-timing decoder was unable to recover the first VRC codeword, but regained synchronization in time to decode the second VRC codeword perfectly. In such a situation, there is no way for the source decoder to recover the $(L+1)$th block of k data symbols, or indeed any subsequent k-blocks that depend on any of the first $L+1$ k-blocks. Thus, dependencies in the encodings of different data k-blocks by the source encoder may lead to decoding delays, in the event of channel errors, that cannot be attributed to loss of synchronization alone. The worst case is when the encoding of each block of k data symbols depends on the previous k-block. In this case, even a single channel error can cause a catastrophic loss of data which cannot be prevented by synchronization codes alone.

Such delays in recovering data symbols can be avoided by requiring the source encoder to encode each block of k data symbols independently of any other block. In fact, one can even allow dependencies in the encoding of data k-blocks, provided this dependency does not cross VRC codeword boundaries, *i.e.*, in order to decode the source codewords contained in some VRC codeword, one does not need to decode the source codewords contained in other VRC codewords. This would require the parameter L of the VRC code to be known to the source encoder.

III. ANALYSIS

In this section, we shall assume that a source code with parameters k and R_s has been fixed, and determine how large a timing span T is achievable by a VRC code with desired rate r and desired delay d, designed to "wrap around" the fixed source code. Since we always assume a source code to be a prefix code which encodes each block of k data symbols independently of any other block of k data symbols, we may freely choose the parameter L of the VRC code.

Let $T_{VRC}(r,d)$ denote the maximum timing span achievable by any VRC code that wraps around the fixed (k, R_s) source code, with rate at least r and delay at most d. If there is no such VRC code with rate at least r and delay at most d, then we define $T_{VRC}(r,d)$ to be 0. The following is a simple upper bound on $T_{VRC}(r,d)$.

Theorem 1: For $r \in (0,1)$ and $d > 0$,

$$T_{VRC}(r,d) \leq \frac{d}{R_s}\, 2^{d(1-r)}$$

Proof: Consider any (p, m_1, m_2, L) VRC code with rate $R \geq r$ and delay $D \leq d$. In order to have p distinct block index codewords, each of length m_2, we must have $m_2 \geq \log_2 p$. It now follows from (2) and (3) that

$$D \geq kLR_s + \log_2 p = RD + \log_2 p$$

But this implies that $\log_2 p \leq D(1-R)$, and hence, $p \leq 2^{D(1-R)}$. Therefore, for any VRC code with $D \leq d$ and $R \geq r$, we have, by way of (1),

$$T = pkL = p\,\frac{DR}{R_s} \leq \frac{dR}{R_s}\, 2^{d(1-r)} \leq \frac{d}{R_s}\, 2^{d(1-r)}$$

the last inequality using the fact that $R \leq 1$. ■

The above bound shows that the maximum timing span of a VRC code grows at most exponentially with delay. We next derive a lower bound on $T_{VRC}(r,d)$, which will show that exponential growth of timing span with delay is in fact achievable by VRC codes. To do this, we make the following simplifying assumption on the source code.

> **IID Assumption**: The bits produced at the output of the source encoder form a sequence of independent, identically distributed random variables, with zeros and ones being equiprobable.

This assumption is a reasonable one to make for any good source code, since if the source-coded (compressed) sequence is not an IID sequence of equiprobable bits, then it would be possible to compress the sequence even further.

The IID assumption on the source code is used to obtain an upper bound on $\overline{S}$, the expected number of stuffed bits in a VRC codeword. Defining $\overline{S}_N$ to be the expected number of stuffed bits required to prevent the occurrence of 1^{m_1}, $m_1 > 1$, in a length-N IID sequence of equiprobable bits, we see that

$$\overline{S}_N = E\left[\sum_{i=m_1-1}^{N} X_i\right] = \sum_{i=m_1-1}^{N} E[X_i]$$

where each X_i is a random variable representing the number of bits stuffed between the ith and $(i+1)$th bits of the IID sequence. Now, $E[X_i]$ is the probability that all the bits between the $(i - m_1 + 2)$th and ith bits (both inclusive) of the IID sequence are ones, which is $2^{-(m_1-1)}$. Therefore, we have $\overline{S}_N = (N - m_1 + 2)\, 2^{1-m_1} \leq N\, 2^{1-m_1}$, since $m_1 > 1$. Now, $\overline{S} = E[\overline{S}_N]$, where the expectation is taken over the length N of the sequence formed by L source codewords prior to bitstuffing. Thus, we see that

$$\overline{S} \leq kLR_s 2^{1-m_1} \tag{4}$$

The above bound is used to derive the following result.

Theorem 2: Under the IID assumption on the source code with parameters $k \geq 1$ and $R_s > 0$, for each $r \in (1/3, 1)$,

$$T_{VRC}(r, d) \geq 2^{d(1-r)-\gamma(k,R_s)+o_d(1)}$$

where $\gamma(k, R_s) = kR_s + \log_2 R_s + 8$, and $o_d(1)$ is a correction term that, for any r, k and R_s, vanishes as $d \to \infty$. More generally, for each $r \in (0, 1)$,

$$T_{VRC}(r, d) \geq 2^{d(1-r)-\gamma(r,k,R_s)+o_d(1)}$$

where $\gamma(r, k, R_s) = 2^{1-\alpha(r)} \frac{r}{1-r} + \alpha(r) - \log_2 \frac{r}{1-r} + kR_s + \log_2 R_s + 6$, with $\alpha(r) = \max(0, \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor)$.

As we shall see, the proof of the theorem provides us with an explicit choice of parameters (p, m_1, m_2, L) for VRC codes that achieve this bound. Thus, while the statement of the theorem is primarily of theoretical interest, its proof has practical significance as it provides us with the means to construct VRC codes whose performance is almost optimal.

Before proceeding to the proof of the theorem, we record the following corollary, which is an immediate consequence of Theorems 1 and 2.

Corollary 3: Under the IID assumption on the source code, for any $r \in (0, 1)$,

$$\lim_{d\to\infty} \frac{\log_2 T_{VRC}(r, d)}{d} = 1 - r$$

This result, when compared to the corresponding results for fixed-rate sync-timing codes ([14], Corollaries 9 and 10), shows that the maximum timing span of VRC codes has the same asymptotic form as that for the families of fixed-rate sync-timing codes considered in [14], *i.e.*, both are asymptotically of

the form $2^{d(1-r)}$.

Proof of Theorem 2: Let $r \in (0,1)$ be fixed. We shall first show that given an integer $N \geq 0$ and any $\epsilon > 0$, there exists an integer $d_N(r,k,R_s,\epsilon)$ such that for all $d > d_N(r,k,R_s,\epsilon)$, there exists a (p,m_1,m_2,L) VRC code with

$$\begin{aligned} p &= \left\lfloor \frac{R_s}{dr}\, 2^{d(1-r)-\gamma_N(r,k,R_s)-\epsilon} \right\rfloor, \\ m_1 &= \lceil \log_2 \log_2 p \rceil + N, \\ m_2 &= \lceil \log_2 p \rceil + 1, \text{ and} \\ L &= \left\lceil \frac{dr}{kR_s} \right\rceil, \end{aligned} \tag{5}$$

that has rate $R \geq r$ and delay $D \leq d$, where $\gamma_N(r,k,R_s) = 2^{1-N}\frac{r}{1-r} + N - \log_2 \frac{r}{1-r} + kR_s + \log_2 R_s + 6$. The timing span of this code is given by $T = pkL = 2^{d(1-r)-\gamma_N(r,k,R_s)-\epsilon+o_d(1)}$, with the $o_d(1)$ correction factor arising from the elimination of the various ceilings and floors in the above choice of parameters. By merging the ϵ into the $o_d(1)$ factor, we thus see that for all $r \in (0,1)$,

$$T_{VRC}(r,d) \geq 2^{d(1-r)-\gamma_N(r,k,R_s)+o_d(1)}$$

The next step in the proof involves showing that $\gamma_N(r,k,R_s)$ is minimized, over all *integers* $N \geq 0$, by $N = \alpha(r) = \max(0, \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor)$. Finally, we show that when $r \geq 1/3$, then with $N = \alpha(r)$, we have $\gamma_N(r,k,R_s) \leq \gamma(k,R_s)$, where $\gamma(k,R_s)$ is as in the statement of the theorem.

So, given an integer $N \geq 0$ and an $\epsilon > 0$, let p, m_1, m_2 and L be as in (5). To show that there exists a VRC code with this choice of parameters, we need to show that there are at least p distinct sequences of length m_2 (for the block index codebook) that do not contain 1^{m_1}. Defining $g(m_1,\mathcal{L})$ to be the number of binary sequences of length $\mathcal{L}$ that do not contain m_1 consecutive ones, we need to show that $g(m_1,m_2) \geq p$. It has been shown ([14], Lemma B.2) that if $\mathcal{L} \leq 2^{m_1} + m_1 - 2$, then $g(m_1,\mathcal{L}) \geq 2^{\mathcal{L}-1}$. It is easily verified that if d is sufficiently large, so that p is sufficiently large, then we have $m_2 \leq 2^{m_1} + m_1 - 2$, and hence, $g(m_1,m_2) \geq 2^{\lceil \log_2 p \rceil} \geq p$.

Next, we have to verify that the rate, R, of this (p,m_1,m_2,L) code is at least r, and its delay, D, is at most d. Actually, it suffices to show that $D \leq d$, because it would then follow that

$$R = \frac{kLR_s}{\overline{\mathcal{N}}} = \frac{kLR_s}{D} \geq \frac{kLR_s}{d} \geq r$$

the last inequality being a consequence of the choice of L.

To verify that $D \leq d$, we first note that from (2) and (4), we have

$$\begin{aligned} D &\leq m_1 + m_2 + 3 + kLR_s(1 + 2^{1-m_1}) \\ &\leq \log_2 \log_2 p + \log_2 p + N + 6 + (dr + kR_s)(1 + 2^{1-\log_2 \log_2 p - N}) \\ &= \log_2 \log_2 p + \log_2 p + N + 6 + (dr + kR_s)(1 + \frac{2^{1-N}}{\log_2 p}) \end{aligned} \tag{6}$$

with the second inequality requiring the repeated use of $x \leq \lceil x \rceil < x + 1$, after replacing m_1, m_2 and L with their respective expressions from (5).

Now, some simple algebraic manipulations show that we can re-write the expression for p in (5) as $p = \lfloor 2^{d(1-r) - \log_2 d - \beta} \rfloor$, where $\beta = 2^{1-N} \frac{r}{1-r} + N + \log_2(1-r) + kR_s + 6 + \epsilon$. Note that if $x \geq 2$, then $x/2 \leq x - 1 \leq \lfloor x \rfloor \leq x$. Hence, if d is sufficiently large, we have $2^{d(1-r) - \log_2 d - \beta - 1} \leq p \leq 2^{d(1-r) - \log_2 d - \beta}$. Using this, we can continue the chain of inequalities in (6) as follows:

$$\begin{aligned} D &\leq \log_2 \log_2 p + \log_2 p + N + 6 + (dr + kR_s)(1 + \frac{2^{1-N}}{\log_2 p}) \\ &\leq d(1-r) - \log_2 d - \beta + \log_2(d(1-r) - \log_2 d - \beta) + N + 6 \\ &\quad + (dr + kR_s)(1 + \frac{2^{1-N}}{d(1-r) - \log_2 d - \beta - 1}) \\ &= d + \log_2 \left((1-r) - \frac{\log_2 d + \beta}{d} \right) + N + 6 - \beta \\ &\quad + kR_s(1 + \frac{2^{1-N}}{d(1-r) - \log_2 d - \beta - 1}) + \frac{2^{1-N} r}{1 - r - \frac{\log_2 d + \beta + 1}{d}} \\ &\leq d + \log_2(1-r) + kR_s + \epsilon/2 + \frac{2^{1-N} r}{1-r} + \epsilon/2 + N + 6 - \beta \end{aligned} \tag{7}$$

the last inequality being true for all $d > d_N(r, k, R_s, \epsilon)$, where $d_N(r, k, R_s, \epsilon)$ is some sufficiently large integer. This is because $\frac{\log_2 d + \beta + 1}{d} \to 0$ and $\frac{2^{1-N}}{d(1-r) - \log_2 d - \beta - 1} \to 0$ as $d \to \infty$.

But, it now follows from (7) and the definition of β that when d is sufficiently large, the VRC code with parameters p, m_1, m_2 and L as above has delay $D \leq d$, and hence as shown previously, has rate $R \geq r$.

We next have to show that the choice of the integer $N \geq 0$ minimizing $\gamma_N(r, k, R_s)$ is $N = \alpha(r) = \max(0, \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor)$. Equivalently, we need to show that this choice of N minimizes $f(N) = 2^{1-N} \frac{r}{1-r} + N$ over all integers $N \geq 0$. It can be verified by differentiation that $f(N)$ is strictly decreasing for $N < \log_2(\frac{2r}{1-r} \ln 2)$ and is strictly increasing for $N > \log_2(\frac{2r}{1-r} \ln 2)$. Hence, the integer that minimizes $f(N)$ is the smallest integer N such that $f(N) < f(N+1)$. This inequality simplifies to $N > \log_2 \frac{2r}{1-r} - 1$, and the smallest N satisfying this is $N = \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor$. Note that $\left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor \geq 0$ if and only if $r \geq 1/3$,

and we are looking for the smallest *non-negative* integer that minimizes $f(N)$. So, in the case when $r < 1/3$, we note that $f(N)$ is strictly increasing for $N \geq 0$, and hence the smallest non-negative integer that minimizes $f(N)$ is 0.

Finally, we need to show that when $r \geq 1/3$, then with $N = \alpha(r)$, we get $\gamma_N(r, k, R_s) \leq kR_s + \log_2 R_s + 10$. It suffices to show that $f(\alpha(r)) \leq \log_2 \frac{2r}{1-r} + 1$, where f is the function defined earlier. Note that for $r \geq 1/3$, $\alpha(r) = \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor$. Defining δ to be the fractional part of $\log_2 \frac{2r}{1-r}$, *i.e.*, $\delta = \log_2 \frac{2r}{1-r} - \left\lfloor \log_2 \frac{2r}{1-r} \right\rfloor$, we see that $f(\alpha(r)) - \log_2 \frac{2r}{1-r} = 2^\delta - \delta$. Note that $0 \leq \delta < 1$. Now, it is easily verified that in this region, $2^\delta - \delta$ is maximized at $\delta = 0$, which shows that $2^\delta - \delta \leq 1$. We have thus shown that $f(\alpha(r)) \leq \log_2 \frac{2r}{1-r} + 1$, which completes the proof of the theorem. ■

We would like to make one final remark on the IID assumption made on the source in deriving the above lower bound on $T_{VRC}(r, d)$. It should be noted that this assumption is only used in deriving an upper bound on $\overline{S}$, the expected number of stuffed bits inserted in a VRC codeword. It can be seen by carefully going through the chain of inequalities in (6) and (7) that if the bound in (4) were to be replaced by $\overline{S} \leq C\, 2^{1-m_1}$ for some constant $C > 0$, then some slight modifications to the above argument would prove the statement of the theorem for a different choice of the γ's. In particular, Corollary 3 would still be valid. It is also conceivable that other versions of the theorem could be proved under still weaker bounds on $\overline{S}$, perhaps by choosing the parameters p, m_1, m_2 and L of the VRC code differently. We therefore believe that it may be possible to show that $T_{VRC}(r, d)$ asymptotically grows as $2^{d(1-r)}$ under weaker assumptions on the source code.

REFERENCES

[1] J.J. Stiffler, *Theory of Synchronous Communications*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[2] R.A. Scholtz, "Frame synchronization techniques," *IEEE Trans. Commun.*, vol. 28, no. 8, pp. 1204–1212, 1980.

[3] S.W. Golomb, J.R. Davey, I.S. Reed, H.L. van Trees and J.J. Stiffler, "Synchronization," *IEEE Trans. Commun. Systems*, vol. 11, no. 4, pp. 481–491, 1963.

[4] E.N. Gilbert and E.F. Moore, "Variable-length binary encodings," *Bell Syst. Tech. J.*, vol. 38, pp. 933–968, 1959.

[5] T.J. Ferguson and J.H. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Trans. Inform. Theory*, vol. 30, pp. 687–693, 1984.

[6] B.L. Montgomery and J. Abrahams, "Synchronization of binary source codes," *IEEE Trans. Inform. Theory*, vol. 32, no. 6, pp. 849–854, 1986.

[7] R.M. Capocelli, A. de Santis, L. Gargano and U. Vaccaro, "On the construction of statistically synchronizable codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 407–414, 1992.

[8] S-M. Lei, "The construction of efficient variable-length codes with clear synchronizing codewords for digital video applications," *Proc. SPIE Visual Comm. Image Proc. '91*, Boston, pp. 863–873, Nov. 1991.

[9] W.M. Lam and A. Reibman, "Self-synchronizing variable-length codes for image transmission," *Proc. ICASSP*, pp. III-477–III-480, Mar. 1992.

[10] D.W. Redmill and N.G. Kingsbury, "The EREC: an error-resilient technique for coding variable-length blocks of data," *IEEE Trans. Image Processing*, vol. 5, no. 4, pp. 565–574, 1996.

[11] M.R. Titchener, "The synchronization of variable-length codes," *IEEE Trans. Inform. Theory*, vol. 43, no. 2, pp. 683–691, 1997.

[12] O.D.J. Thomas, D.H. Smith and A. Ryley, "An adaptive error-tolerant and lossless data compressor for DDS," *J Inf. Recording*, vol. 23, pp. 547-557, 1997.

[13] S. Perkins and D.H. Smith, "Synchronization of variable length codes," *Discrete Applied Mathematics*, vol. 101, pp. 231-245, 2000.

[14] N. Kashyap and D.L. Neuhoff, "Data synchronization with timing," *IEEE Trans. Inform. Theory*, vol. 47, no. 4, pp. 1444–1460, 2001.

[15] N. Kashyap and D.L. Neuhoff, "Codes for data synchronization with timing," *Proc. Data Compression Conference*, Snowbird, Utah, pp. 443–452, 1999.

[16] S. Perkins, D.H. Smith and A. Ryley, "Robust Data Compression: Consistency Checking in the Synchronization of Variable Length Codes," *The Computer Journal*, vol. 47, pp. 309–319, 2004.

[17] M.Y. Cheung and J. Vaisey, "A comparison of scalar quantization strategies for noisy data transmission," *IEEE Trans. Commun.*, vol. 43, pp. 738–742, 1995.

[18] A. Kiely, S. Dolinar, M. Klimesh and A. Matache, "Synchronization Markers for Error Containment in Compressed Data," *TMO Progress Report 42-136*, Oct.–Dec. 1998, pp. 1–40, Feb. 15, 1999. `http://tmo.jpl.nasa.gov/tmo/progress_re port/42-136/136H.pdf`.

[19] A. Kiely, S. Dolinar, M. Klimesh and A. Matache, "Error Containment in Compressed Data Using Sync Markers," *Proc. Int. Symp. Inform. Theory (ISIT)*, Sorrento, Italy, p. 428, June 2000.

[20] W.H. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inform. Theory*, vol. 11, pp. 284–292, 1965.