Synchronization Markers for Error Containment in Compressed Data

A. Kiely,[1] S. Dolinar,[1] M. Klimesh,[1] and A. Matache[1]

We examine a specific strategy of using synchronization (sync) markers for error containment in compressed data, using a model that separates the data-compression and error-containment stages. We examine the trade-off between rate and error performance, determine classes of sync markers that give equivalent performance, and examine the optimum choice of sync marker and data block length. The performance of this strategy on additive white Gaussian noise (AWGN) uncoded and convolutionally encoded channels is analyzed.

I. The Error Containment Problem

A drawback of nearly all data compression schemes when used on noisy channels is that a single channel error can corrupt a long sequence of encoded data. This presents little difficulty for computer storage devices, where bit errors are extremely rare, or on networks, where effective error detection and retransmission strategies are used. However, this is a significant concern when attempting to use data compression on deep-space links. The elimination of data compression altogether is not usually an efficient solution unless the source is nearly incompressible in the first place, so if we are to use compression effectively, we must incorporate a method of containing the effects of channel errors.

Channel codes can help matters by reducing the channel-error probability. In fact, information theory tells us that we can use very long channel codes to make the fraction of data corrupted by errors arbitrarily small. However, practical constraints on complexity and decoding delay limit the attainable channel-error probability. Therefore, we assume that a channel code has already been fixed, and our interest then is in the performance of low-complexity error-containment methods to be used outside the layer of error correction.

One solution is to restrict ourselves to data-compression methods that map fixed-length input sequences onto fixed-length output sequences, where each input sequence is encoded independently of the others. This ensures that the effects of a channel error are confined to the block in which the error occurred. However, such a technique usually cannot be lossless, and more significantly, the efficiency of such an encoder usually is worse than that for an encoder with variable-length codewords.

Instead, we turn our attention to a simpler strategy, which is the use of special sequences, called synchronization (or "sync") markers, to separate the variable-length blocks of compressed data. This

[1] Communications Systems and Research Section.

strategy already has been used, for example, on Galileo [2]. Our aim is to determine how to choose such markers and analyze the impact on overall performance of such strategies.

Other authors have examined problems related to synchronization. Gilbert [3] examines the design of fixed-length codes that essentially use a sync-marker prefix to identify the beginning of each codeword. Stiffler [6, Chapter 11] discusses general conditions for variable-length source codes to be synchronizable— that is, following a channel error, codeword boundaries can be identified within bounded time. In this article, we examine a specific simple error-containment strategy applied after a variable-length data-compression stage.

II. A Simple Scheme for Error Containment

Figure 1 shows the block diagram of a communications system using data compression and error containment. The source is a device that produces discrete outputs, e.g., a charge-coupled device (CCD) camera that produces integer pixel intensity values. We make the simplifying assumption that the source outputs are independent and identically distributed (i.i.d.) random variables. The data-compression stage maps each source output to a binary codeword using a variable-length code (e.g., a Huffman code). Compression might be lossless or lossy, i.e., the mapping may or may not be one to one.

Fig. 1. Block diagram for a communications system using data compression with error containment.

At the output of the data-compression stage, we assume that zeros and ones are equally likely and that if an error were to occur at this point there would be no way to detect it. This first condition is a good approximation for any good compression scheme— if zeros and ones are not equally likely, then an additional layer of compression could be applied to further reduce the data volume. The second condition is satisfied by any exhaustive code, such as a Huffman code. Any source code that maximizes the amount of compression cannot be used to detect errors, because if the source code can detect errors, then it contains redundancy that could be removed to improve compression efficiency. These assumptions about the data-compression stage allow our analysis of the error-containment stage to be largely independent of the details of the data compressor.

It is the goal of the error-containment stage to introduce redundancy into the compressed-data stream in a way that allows us to contain the effects of channel errors. Since we are adding redundancy for the sake of error control, the error-containment scheme is essentially an extra layer of channel coding, but the amount of redundancy is dependent on the output of the data compressor.

The error-containment strategy we wish to analyze is the use of sync markers. Under this scheme, source-data samples are partitioned into blocks of a fixed number, B, of samples, and each such block is compressed independently of the others.[2] The sync-marker sequence is transmitted between compressed blocks. A channel code is applied, but for our analysis, we can treat the channel encoder and decoder as part of the channel.

At the receiving end, the decompressor receives bits from the channel and reconstructs the source symbols. When it has reconstructed B symbols, it expects to observe another sync marker in the sequence. If none is observed, the receiver concludes that a channel error has occurred and looks for the sync marker

[2] In our idealized model, we assume that compression performance is unaffected by partitioning of the source output stream. In most practical schemes, however, we pay an increasing penalty in compression performance as we make the block size smaller.

with the hope of identifying the next data block. In this manner, following a channel error, we can contain the effects of the error. Thus, through the transmission of redundant data (the sync marker), we hope to trade communication rate for error protection.

Knowing the expected number of compressed bits per source symbol and selection of a *fixed* block length, B, allows us to control the *expected value* of the variable number of compressed bits per block, n. For now, we focus on the problem of transmitting a block of n compressed bits, treating n as a fixed quantity.

Because all binary sequences are possible from the compressor, the sync-marker sequence may occur by chance within the compressed data sequence. Following a channel error, we may mistake such a "bogus" sync marker for a true sync marker. Typically, long sync markers are used to make the probability of bogus sync markers small. In our case, we consider both long and short markers, so we take specific steps to eliminate bogus sync markers from the transmitted sequence altogether.

To eliminate bogus sync markers, we simply insert bits in the transmitted data sequence wherever needed to prevent their occurrence. For example, if the sync marker is 1101, whenever we observe 110 in the transmitted data stream, we insert an additional 0 preceding the next transmitted bit, x. Note that this is necessary whether x is 0 or 1 so that, when the receiver observes 110$\underline{0}$, it knows that the underlined bit is an inserted bit, not a data bit. That is, the bit insertion process must be causal. In general, when we observe the first $n-1$ bits of the sync marker, we must insert a bit to prevent the appearance of a bogus sync marker.

We also may require the insertion of bits immediately before or after the sync marker. For example, if the sync marker is 0000, we obtain a bogus sync marker if the data sequence begins or ends with 0. A causal method of preventing this is to insert a 1 following the sync marker, and when the compressed data sequence ends with 0, to insert a 1 preceding the next sync marker.

This strategy prevents us from using certain sync markers. For example, consider what happens if we attempt to use sync-marker 0001 and our data sequence contains 000. This would necessitate inserting a never-ending sequence of zeros to avoid duplicating the marker. For this reason, for any $m \geq 1$, sync markers $0^{m-1}1$ and $1^{m-1}0$ are called "catastrophic" (we use 0^i and 1^i to denote length i runs of 0 and 1, respectively). These are the only catastrophic sync markers, but not all sync markers offer the same performance. In the next section, we show how to compute the expected number of inserted bits for any noncatastrophic sync marker.

This problem has some similarities with coding for run-length-constrained channels, where the technique traditionally employed is to use long codewords that are designed to ensure that only "allowed" sequences are transmitted [1]. However, our constraint is easier to satisfy, and our strategy of inserting bits to prevent marker replication is much simpler than the design of long run-length-constrained codes.

III. Analyzing Sync-Marker Performance

A. Performance Analysis Based on State Diagrams

We can describe the bit-insertion procedure using a state diagram that is essentially the same as the one in [3], which was used for combining sync markers with fixed-length codes. For a sync marker of length m, the states are numbered 0 through $m-1$. The encoder is in state $m-i$ if the sync marker can be produced by the transmission of i additional bits but no fewer.

For example, Fig. 2 shows the state diagram for bit insertion when the sync marker is 1111. No path through the state diagram contains the sync marker, and every other sequence corresponds to some path through the diagram. Note that each state has two output transitions except for state 3, which

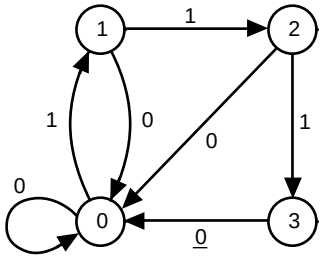

Fig. 2. State diagram for sync marker 1111.

corresponds to the case when the preceding 3 bits are 111 and a 0 must be inserted. We refer to this state as "uninformative" because the transition out of this state conveys no information, i.e., it represents an inserted bit rather than an information bit. The inserted bit is underlined to distinguish it from the information bits. At any time, the probability that we have to insert a bit (and hence expand the data) is equal to the probability of occupying the uninformative state.

To compute the expected number of inserted bits for a given sync marker and block length, we would like to count the number of inserted bits as a function of the number of information bits. To facilitate this, we modify the state diagram by eliminating the uninformative state, merging the edges into and out of this state to form a single edge. Following this modification, each edge corresponds to a single information bit. We also add a "start" state with an output edge that accounts for any "start-cost," i.e., inserted bits that may be required immediately following the sync marker at the beginning of the block. To account for the "end-cost," i.e., insertions that may occur at the end of the block, we use a different state diagram with transitions to an "end" state. Once we have arrived at the end state, we can transmit the sync marker to signal the boundary between data blocks.

Figure 3 illustrates the modified state diagrams used for sync-marker 1111 to transmit a block of n information bits. Figure 3(a) shows the state diagram for time index $t = 1, 2, \cdots, n-1$, and Fig. 3(b) shows it for $t = n$.

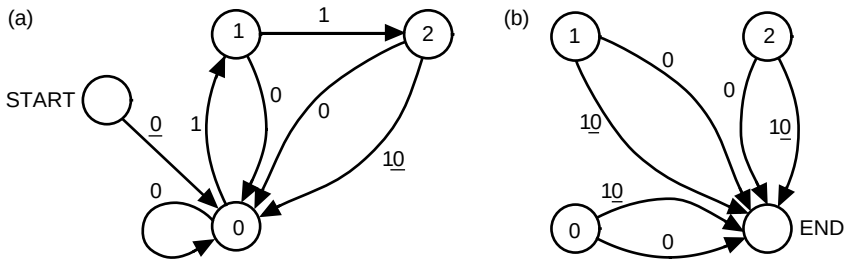

Fig. 3. The modified state diagram for sync marker 1111 for time index (a) *t* < *n* and (b) *t* = *n*.

Continuing with this example, let $p_i(j)$ denote the probability of being in state i after j information bits have been transmitted. From the state diagram, clearly $p_1(0) = p_2(0) = 0$ and $p_0(0) = 1$ (note that no information bit is transmitted on the edge out of the start state). The probability distribution at time index i is

$$\mathbf{p}(i) \triangleq \begin{bmatrix} p_0(i) \\ p_1(i) \\ p_2(i) \end{bmatrix} = \underbrace{\begin{bmatrix} 1/2 & 1/2 & 1 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} p_0(i-1) \\ p_1(i-1) \\ p_2(i-1) \end{bmatrix} = \mathbf{A}^i \mathbf{p}(0)$$

where $\mathbf{A}$ is the state transition matrix and $\mathbf{p}(0)$ is the initial probability distribution, which is simply a vector indicating which state is connected to the start state.

Given a block of n information bits, let $I_n^{(j)}$ denote the expected cumulative number of inserted bits when we have transmitted the jth information bit (including the inserted bit, if any, following this information bit). Our goal is to compute $I_n \triangleq I_n^{(n)}$, the expected total number of bits inserted in the block.

Let δ denote the number of bits inserted before the first information bit. It is not hard to show that δ equals 1 if and only if the sync marker is $000\cdots0$ or $111\cdots1$ and δ equals 0 otherwise.

In this example, we find that

$$I_n^{(j)} = \begin{cases} \delta, & j = 0 \\ \delta + \dfrac{1}{2}, & j = n = 1 \\ I_n^{(j-1)} + \dfrac{1}{2}p_2(j-1), & 0 < j < n \\ I_n^{(n-1)} + \dfrac{1}{2}, & j = n \neq 1 \end{cases}$$

and for $j < n$, this simplifies to

$$I_n^{(j)} = \delta + \sum_{i=0}^{j-1} \frac{1}{2}p_2(i) = \delta + \frac{1}{2}\begin{bmatrix}0 & 0 & 1\end{bmatrix}\sum_{i=0}^{j-1}\mathbf{A}^i\mathbf{p}(0)$$

The main quantity of interest is

$$I_n = \delta + \frac{1}{2}\begin{bmatrix}0 & 0 & 1\end{bmatrix}\sum_{i=0}^{n-2}\mathbf{A}^i\mathbf{p}(0) + \frac{1}{2}$$

In general, to compute I_n for a sync marker s, we have a term δ to account for insertions at the beginning of the block, a term $(1/2)\mathbf{e}^{\mathrm{T}}\mathbf{p}(n)$ to account for insertions at the end, and a sum to account for insertions in the interior:

$$I_n = \begin{cases} \delta + \dfrac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{p}(0), & n = 1 \\ \delta + \dfrac{1}{2}\left(\mathbf{v}^{\mathrm{T}}\sum_{i=0}^{n-2}\mathbf{A}^i + \mathbf{e}^{\mathrm{T}}\mathbf{A}^{n-1}\right)\mathbf{p}(0), & n \geq 2 \end{cases} \tag{1}$$

where $\mathbf{A}$ is the transition matrix for the modified state diagram, $\mathbf{p}(0)$ is a column vector indicating which state is connected to the start state, $\mathbf{v}$ is a column vector indicating which state can cause insertions for time $t < n$, and $\mathbf{e}$ is a column vector indicating which states can cause insertions at time $t = n$.

For practical computations, a couple of simplifications sometimes can facilitate calculation of I_n. If $\mathbf{A}$ has distinct eigenvalues, we can write $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{C}\,\mathbf{\Lambda}\mathbf{C}^{-1}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues λ_j and the columns of $\mathbf{C}$ are the corresponding eigenvectors.[3] Thus, for $n \geq 2$, Eq. (1) reduces to

$$I_n = \delta + \frac{1}{2}\left(\mathbf{v}^{\mathrm{T}}\mathbf{C}\sum_{i=0}^{n-2}\mathbf{\Lambda}^i\mathbf{C}^{-1}\mathbf{p}(0) + \mathbf{e}^{\mathrm{T}}\mathbf{C}\mathbf{\Lambda}^{n-1}\mathbf{C}^{-1}\mathbf{p}(0)\right)$$

$$= \delta + \frac{1}{2}\left(\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{\Lambda}'\mathbf{C}^{-1}\mathbf{p}(0) + \mathbf{e}^{\mathrm{T}}\mathbf{C}\mathbf{\Lambda}^{n-1}\mathbf{C}^{-1}\mathbf{p}(0)\right)$$

where $\mathbf{\Lambda}'$ is a diagonal matrix with diagonal elements $(1-\lambda_j^{n-1})/(1-\lambda_j)$, when $\lambda_j < 1$, or $n-1$, when $\lambda_j = 1$.

Fast numerical computation of I_n can be accomplished recursively. From Eq. (1), the difference between two successive values of I_n is

$$I_n - I_{n-1} = \frac{1}{2}\left\{(\mathbf{v}-\mathbf{e})^{\mathrm{T}}\mathbf{A}^{n-2}\mathbf{p}(0) + \mathbf{e}^{\mathrm{T}}\mathbf{A}^{n-1}\mathbf{p}(0)\right\} = \frac{1}{2}\left\{(\mathbf{v}-\mathbf{e})^{\mathrm{T}}\mathbf{p}(n-2) + \mathbf{e}^{\mathrm{T}}\mathbf{p}(n-1)\right\}$$

[recall that $\mathbf{p}(n) = \mathbf{A}^n\mathbf{p}(0)$], so we can compute I_n recursively using $I_1 = \delta + (1/2)\mathbf{e}^{\mathrm{T}}\mathbf{p}(0)$, and for $n > 1$:

$$I_n = I_{n-1} + \frac{1}{2}\left\{(\mathbf{v}-\mathbf{e})^{\mathrm{T}}\mathbf{p}(n-2) + \mathbf{e}^{\mathrm{T}}\mathbf{p}(n-1)\right\}$$

where $\mathbf{p}(n)$ is calculated recursively by $\mathbf{p}(n) = \mathbf{A}\mathbf{p}(n-1)$. We evaluate I_n for some special cases in Section III.C.

The rate of an error-containment strategy is the ratio of the number of information bits to the total number of bits transmitted. The best asymptotic rate (for long sequences of information bits) obtainable while avoiding certain sequences is well known; Shannon [5] outlined a general method of computing this rate. Our method of avoiding the sync-marker sequence cannot achieve this asymptotic rate since our state diagrams do not have the optimal transition probabilities.

However, our method gives asymptotic rates that are close to optimal. For example, for the sync marker 1111, the best possible asymptotic ratio of increase in length to information bits is about 0.056, while our method gives $1/14 \approx 0.071$. For the marker 11111111, the corresponding numbers are 0.0029 and $1/254 \approx 0.0039$. Furthermore, as we will see later, the asymptotic rate of a strategy for using a specific sync marker does not dominate the performance of the strategy until the block length is sufficiently long that performance is improved by switching to a longer sync marker.

B. Equivalent Sync Markers

To compare the performance of different sync markers, it is convenient to modify our notation for the number of inserted bits to explicitly show dependence on the choice of marker. We let $I_n(s)$ denote the average number of inserted bits when the block size is n and the sync marker is s, and we use this notation throughout the remainder of this article.

Two length-m sync markers, s and t, are said to be equivalent when

[3] If $\mathbf{A}$ does not have distinct eigenvalues, a similar diagonalization can be used.

$$I_n(s) = I_n(t) \quad \text{for all } n > 0 \tag{2}$$

The significance of this relationship is that two equivalent sync markers will give identical performances in the error-containment scheme. This definition makes no explicit reference to state diagrams, and in fact two equivalent sync markers may have quite different state diagrams.

This equivalence relation partitions all length-m binary strings, $1s_2s_3\cdots s_m$, into classes, which we refer to as "s-classes." Clearly, a sync marker and its complement (all bits inverted) are always equivalent, so, for convenience and without loss of generality, we assume all sync markers begin with a 1. By convention, the catastrophic sync marker $11\ldots110$ forms its own s-class. We do not compare sync markers of different lengths in this section.

Note that in practice there may be a preference of which sync marker to employ among equivalent markers because the incoming bits may not be completely random. For example, the sync markers 10000001 and 11010001 are equivalent, but the latter may be preferable if the incoming bits could have a propensity to include runs of 0's.

Knowledge of the equivalence classes for a given marker length, m, is useful for evaluating the performance of all markers of length m, because it becomes necessary to test only the performance of one member from each s-class. However, checking for equivalence directly with Condition (2) involves evaluating the performance for each marker anyway, so this does not save any work. We will describe an easy-to-compute, sufficient condition for two sync markers to be equivalent. It appears likely that this condition also is necessary, and in fact its necessity has been verified for sync-marker lengths $m \leq 8$.

In order to describe this condition, we introduce the concept of the overlap set of a bit string. The overlap set of the bit string $s = s_1s_2\cdots s_m$ is the set of all i, with $1 \leq i \leq m-1$, such that the string $s_{m-i+1}\cdots s_m$ is identical to the string $s_1\cdots s_i$. In other words, i is in the overlap set if s can be written twice with i bits overlapping and the overlapping bits are identical. For example, the overlap set of 10101101 is $\{1, 3\}$.

Let $V_1(s)$ denote the overlap set of s, and let $V_2(s)$ denote the overlap set of $s_1\cdots s_{m-1}\overline{s_m}$ (s with the last bit inverted). If s and t are length-m sync markers for which $V_1(s) = V_1(t)$ and $V_2(s) = V_2(t)$, then s and t are said to be in the same "v-class." For example, Table 1 shows the partitioning of all of the length-6 sync markers into v-classes. We observe in Table 1 that length-6 v-classes come in pairs corresponding to two v-classes differing from each other only in the last bit of their members. In fact, Corollary A-2 in Appendix A shows that this is true for all sync-marker lengths.

The significance of the v-classes is given by the following theorem:

Theorem 1. *If s and t are length−m sync markers that are members of the same v-class, then s and t are in the same s-class.*

This result and other related results are proved in Appendix A.

Fortunately, the number of v-classes is considerably smaller than the number of sync markers for a given length. Table 2 shows the number of v-classes for sync-marker lengths up to 13.

C. Three Important Special Cases

Given a block length n, for any fixed sync-marker length $m \leq 8$, we have verified empirically (and we conjecture that it is true for all m) that the optimal sync marker must belong to one of three classes.

Table 1. The 16 v-classes of length-6 sync markers. These are identical to the length-6 s-classes.

Markers ending in 0		Markers ending in 1	
Marker s	$(V_1(s), V_2(s))$	Marker s	$(V_1(s), V_2(s))$
100000	$(\emptyset, \{1\})$	100001	$(\{1\}, \emptyset)$
101000	$(\emptyset, \{1\})$	101001	$(\{1\}, \emptyset)$
110000	$(\emptyset, \{1\})$	110001	$(\{1\}, \emptyset)$
110100	$(\emptyset, \{1\})$	110101	$(\{1\}, \emptyset)$
111000	$(\emptyset, \{1\})$	111001	$(\{1\}, \emptyset)$
111100	$(\emptyset, \{1\})$	111101	$(\{1\}, \emptyset)$
100100	$(\{3\}, \{1\})$	100101	$(\{1\}, \{3\})$
101100	$(\emptyset, \{1, 3\})$	101101	$(\{1, 3\}, \emptyset)$
100010	$(\{2\}, \{1\})$	100011	$(\{1\}, \{2\})$
100110	$(\{2\}, \{1\})$	100111	$(\{1\}, \{2\})$
101110	$(\{2\}, \{1\})$	101111	$(\{1\}, \{2\})$
101010	$(\{2, 4\}, \{1\})$	101011	$(\{1\}, \{2, 4\})$
110010	$(\emptyset, \{1, 2\})$	110011	$(\{1, 2\}, \emptyset)$
111010	$(\emptyset, \{1, 2\})$	111011	$(\{1, 2\}, \emptyset)$
110110	$(\{3\}, \{1, 2\})$	110111	$(\{1, 2\}, \{3\})$
111110	$(\emptyset, \{1, 2, 3, 4, 5\})$	111111	$(\{1, 2, 3, 4, 5\}, \emptyset)$

Table 2. Number of v-classes as a function of sync-marker length $|s|$.

$\lvert s\rvert$	Number of v-classes
2	2
3	4
4	6
5	10
6	16
7	22
8	30
9	42
10	58
11	76
12	96
13	116

These classes are those containing 10^{m-1}, $10^{m-2}1$, and 1^m, which we refer to as class 1, class 2, and class 3, respectively. In Appendix B, we derive the following expressions for the expected number of inserted bits for each of these classes.

Theorem 2. *For class-1 markers (e.g., 10^{m-1}), class-2 markers (e.g., $10^{m-2}1$), and the class-3 sync marker 1^m, the average number of inserted bits $I_n(s)$ takes the following form wherever it is nonzero:*

$$I_n(s) = \frac{n}{a(s)} + b(s) + c_n(s)2^{-n}$$

where $a(s)$and $b(s)$ are independent of n, and $c_n(s)$ is periodic in n with a short period on the order of the length of s.

(a) *For any length-m class-1 sync marker,*

$$a(s) = 2^{m-1} - 2$$

$$b(s) = -\frac{(m-1)2^{m-2} - 1}{a(s)(2^{m-2} - 1)}$$

and

$$c_n(s) = 2^{r_{n-1}(m-2)} \frac{(m-2)2^{m-2} - r_{n-1}(m-2)(2^{m-2} - 1)}{a(s)(2^{m-2} - 1)}$$

where $r_{n-1}(m-2)$ is the remainder when we divide $n-1$ by $m-2$. For this marker, $I_n(s) = 0$ for all $n < m-1$, and the expression in the theorem is valid for all $n \geq m-1$, for any $m \geq 3$.

(b) *For any length-m class-2 sync marker,*

$$a(s) = 2^{m-1}$$

$$b(s) = -\frac{(m-4)}{a(s)}$$

and

$$c_n(s) = 0$$

For this marker, $I_n(s) = 0$ for all $n < m-2$, and the expression in the theorem is valid for all $n \geq m-2$, for any $m \geq 2$.

(c) *For the length-m class-3 sync marker* 1^m,

$$a(s) = 2^m - 2$$

$$b(s) = \frac{3}{2} - \frac{(m-2)2^{m-1}+1}{a(s)(2^{m-1}-1)}$$

and

$$c_n(s) = 2^{1+r_{n-1}(m-1)} \frac{(m-3)2^{m-1}+2-r_{n-1}(m-1)(2^{m-1}-1)}{a(s)(2^{m-1}-1)}$$

where $r_{n-1}(m-1)$ *is the remainder when we divide* $n-1$ *by* $m-1$. *For this marker,* $I_n(s) \geq 3/2$ *for all* n, *and the expression in the theorem is valid for* $m \geq 2, n \geq 1$.

D. Performance Comparisons Between Different Sync Markers

In order to compare the performances of our error-containment strategy for sync markers of different lengths, we compute the total amount of "data expansion" resulting from error containment, i.e., the average number of extra bits that is added to each data block for synchronization purposes. This includes both the length of the marker and the average number of bits inserted to avoid inadvertently duplicating the marker. For a sync marker s and a length-n data block, the average data expansion is given by

$$X_n(s) = |s| + I_n(s)$$

where $|s|$ denotes the length of sync marker s.

We say that a sync marker s is globally optimum for a block length n if it minimizes $X_n(s)$ among all sync markers of any length. Figure 4 shows the average data expansion for all sync markers of length 5 or less. We see from the figure that only four equivalence classes are ever globally optimum for $n \leq 30$. These are the classes containing the markers 100, 1000, 10000, and 10001.

For any two sync markers, s and s', the difference between the average amounts of data expansion caused by the two markers is $X_n(s) - X_n(s')$. In the following subsections, we compute these performance differences for various pairs of sync markers.

1. Class-2 Markers Compared for Different Lengths. The easiest comparisons involve the class-2 sync markers, e.g., $s = 10^{m-2}1$, because the terms in the performance expression for this class of markers are either constant or linear in n. Comparing $10^{m-2}1$ with $10^{m-1}1$, we find the following result (derived in Appendix C, Section I).

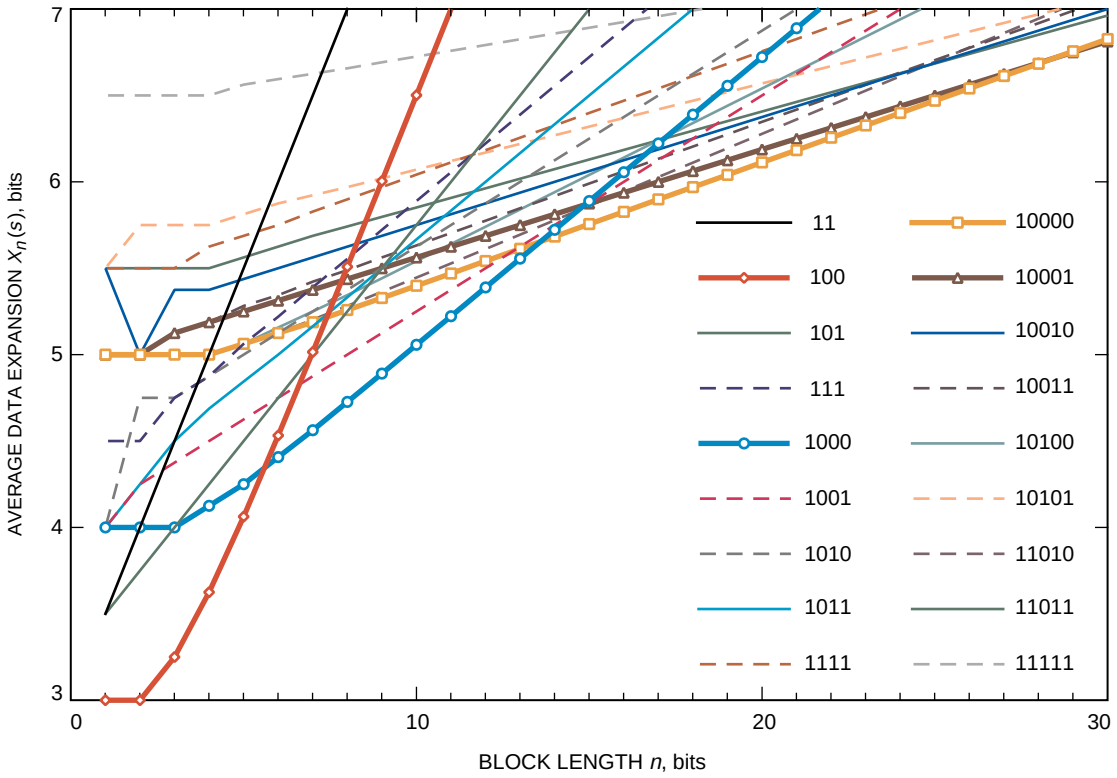

Fig. 4. Average data expansion $X_n(s) = |s| + I_n(s)$ as a function of block length n for all sync markers of length 5 or less. One sync marker from each equivalence class is identified.

Theorem 3.

(a) *For any $m \geq 2$, a class-2 marker of length m is optimum within the set of class-2 markers of arbitrary lengths only for block lengths within the interval from $n = 2^{m-1} + m - 6)$ to $n = 2^m + (m - 5)$.*

(b) *For all $n \geq 0$, if the length m of a class-2 marker is chosen optimally according to part (a), then the average number of inserted bits $I_n(s)$ is bounded by*

$$1 - \frac{1}{2^{m-2}} \leq I_n\left(10^{m-2}1\right) \leq 2 - \frac{1}{2^{m-1}}$$

(c) *For all $n \geq 8$, if the length m of a class-2 marker s is chosen optimally according to part (a), then the average data expansion $X_n(s)$ is bounded by*

$$\log_2 n + \kappa(m) \leq X_n(10^{m-2}1) \leq \log_2 n + 2$$

where $\kappa(m) = \log_2(2e \ln 2) - (m-4)(1/2)^{m-1} \approx 1.91$ for large m.

The first part of this theorem defines the interval of block sizes n where a class-2 marker of a given length m outperforms class-2 markers of all other lengths. For different values of m, these intervals of optimality are nonoverlapping except at their endpoints, where either of the corresponding optimum values of m is equally good. As m becomes large, these intervals of optimality are approximately the

intervals between successive powers of 2. We shall see below that, for any $m \geq 5$, within some portion of this interval a class-2 marker of length m is in fact optimum among sync markers of all lengths from the three special classes and probably is globally optimum in the same portion of the interval.

The second part of the theorem shows that, if the length m of a class-2 marker is optimized to minimize the average data expansion $X_n(s)$, then the average number of inserted bits, $I_n(s)$, is always less than 2 bits, no matter how large the block size n. Inside the interval of optimality for any given m, $\min_m I_n(10^{m-2}1)$ increases from just under 1 bit at the lower endpoint to just under 2 bits at the upper endpoint. Then, at the next higher value of n, the optimum marker length is increased by 1 bit, while the average number of inserted bits starts anew another slow climb from approximately 1 bit to approximately 2 bits at the next endpoint. This oscillation continues indefinitely.

The third part of the theorem gives tight upper and lower bounds on the average data expansion of a class-2 marker with optimized length, as a simple function of block size n. The lower bound is within 0.1 bit of the upper bound for large m, because $\log_2(2e \ln 2) \approx 1.91$. The upper bound, $X_n(s) \leq \log_2 n + 2$, is of course an upper bound also on the performance of the globally optimum sync marker, and it turns out that this upper bound also is very tight when applied to the performance of the globally optimum sync marker (see Fig. 7 below). For very small values of block size ($n < 8$), the upper bound is not applicable to optimum class-2 marker performance, but, in these cases, class-2 markers are never globally optimum, and the globally optimum marker still obeys the upper bound for all $n \geq 2$.

2. Class-1 Markers Compared With Class-3 Markers. Another useful comparison can be made between a class-1 marker of length m and the class-3 marker of length $m-1$. In Appendix C, Section II, we derive the following result:

Theorem 4. *For all $m \geq 3$, the average data expansion for the class-3 marker of length $m-1$ exceeds that of a class-1 marker of length m by at least $1/2$ bit. Specifically, this performance difference is evaluated as*

$$X_n(1^{m-1}) - X_n(10^{m-1}) = \frac{1}{2} + \frac{1}{2^{m-2}-1}\left[1 - 2^{1+r_{n-1}(m-2)-n}\right]$$

where $r_{n-1}(m-2)$ is the remainder when we divide $n-1$ by $m-2$.

We note that the performance difference between the two markers stays constant over blocks of $(m-2)$ consecutive values of n. The limiting difference, as $n \to \infty$ for fixed m, is $1/2 + 1/(2^{m-2}-1)$. If $m \to \infty$ along with n, then this difference approaches a limiting value of $1/2$ bit.

The main conclusion from this theorem is that, while the average number of inserted bits for the class-3 marker of length m has the slowest asymptotic growth rate among all markers of the same length, this type of marker never is globally optimum with respect to the criterion of minimizing the average data expansion, $X_n(s)$. The explanation for this paradoxical result is that the class-3 marker has such a high start cost and end cost (an average of $3/2$ bits) that, as n is increased, it always pays to invest an extra start-up bit in lengthening a different marker rather than waiting for the class-3 marker's superior asymptotic growth rate to dominate. This conclusion is not too surprising in light of the result in Theorem 3(c) that the average number of inserted bits for an optimum marker, restricted to class-2 markers, oscillates between approximately 1 and 2 bits no matter how large n is; thus, start and end costs of $3/2$ bits are never negligible compared with the optimum average number of inserted bits, even for arbitrarily large n. This trade-off between a marker's start and end costs and its asymptotic cost growth rate is discussed further for more general markers in Section III.E.

3. Class-1 Markers Compared With Class-2 Markers. A comparison between class-1 markers and class-2 markers is interesting because there are distinct regions of block sizes n where each type

of marker is slightly superior to the other. The following theorem (proved in Appendix C, Section III) summarizes this comparison.

Theorem 5. *For any $m \geq 3$, there is a single crossover point,*

$$n_{\times}(m) = 3 \times 2^{m-2} + 2m - 6$$

such that the average data expansion for a class-2 marker of length m compared with that of a class-1 marker of length m is higher for all $n \leq n_{\times}(m)$and lower for all $n > n_{\times}(m)$. Furthermore, there is a single crossover point,

$$n_{+}(m) = 2^{m} + m - 6$$

such that the average data expansion for a class-2 marker of length m compared with that of a class-1 marker of length $m+1$ is higher for all $n \geq n_{+}(m)$and lower for all $n < n_{+}(m)$.

Note that $n_{\times}(4) = 14 = n_{+}(4)$, and in this case, the theorem states that the class-2 marker 1001 must be inferior to the class-1 marker 1000 for all $n \leq 14$ and inferior to the class-1 marker 10000 for all $n \geq 14$, i.e., the class-2 marker 1001 never can be globally optimum. (This is confirmed in Fig. 4.) A similar result holds for the class-2 marker 101. However, for all $m \geq 5$, there is a nonempty region between $n_{\times}(m)$ and $n_{+}(m)$ where the class-2 marker $10^{m-2}1$ is superior to any class-1 marker. For large m, the size of this region is nearly equal (proportionately) to the region where a class-1 marker of the same length is superior to any class-2 marker.

Combining the results of Theorem 5 for $m \geq 5$ with numerical comparisons among class-1 markers for $m < 5$, we arrive at Fig. 5, which shows the regions of relative optimality among class-1 and class-2 markers of different lengths. The indicated markers of different lengths are relatively optimum (i.e., inside the universe of class-1 markers and class-2 markers only) within their respective open or closed n-intervals with upper and lower boundaries as shown in the figure. Figure 5 has additional significance in that our numerical computations (see Section III.F) have shown that class-1 and class-2 markers of different lengths as shown in the figure are in fact globally optimum for all $m \leq 12$, and we conjecture that this is true for all m.

Another interesting observation is that the transition point $n = n_{+}(m)$ between class-2 markers of length m and class-1 markers of length $m+1$ is exactly 1 less than the transition point $n = 2^{m} + m - 5$ between class-2 markers of lengths m and $m+1$ derived in Theorem 3. This implies that the maximum penalty for using an optimum-length class-2 marker in place of an optimum-length class-1 marker of length $m \geq 6$ cannot be larger than the difference between the average data expansion for class-2 markers of lengths $n = n_{+}(m) + 1$ and $n = n_{+}(m) - 1$, which equals $1/2^{m-2}$. Thus, for large m, it is not very suboptimum to restrict the marker choice to class-2 markers, and a similar result can be shown if the choice is restricted to class-1 markers.

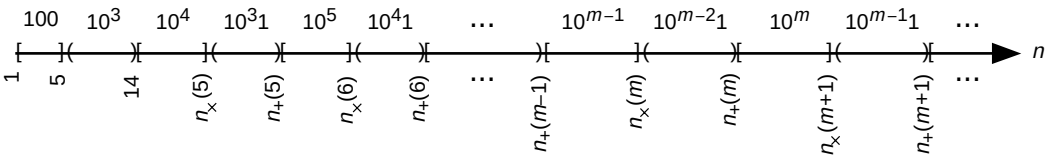

Fig. 5. Regions of relative optimality for class-1 and class-2 markers of different lengths. A representative from each class is shown.

E. General Asymptotic Performance Expressions

The asymptotic growth rate, as a function of n, of the average number of inserted bits, $I_n(s)$, can be neatly evaluated for a completely general sync marker, s, in terms of the marker's overlap sets, $V_1(s)$ and $V_2(s)$. The following theorem states this result.

Theorem 6. *Let s be a length-m sync marker that is not the catastrophic marker. Then*

$$\lim_{n\to\infty} \frac{1}{n} I_n(s) = \left(2^{m-1} - 1 + \sum_{i \in V_1(s)} 2^{i-1} - \sum_{i \in V_2(s)} 2^{i-1} \right)^{-1}$$

This result is proved in Appendix D.

Theorem 6 shows that the asymptotic performance of a sync marker, s, will be best if $V_1(s)$ is large and $V_2(s)$ is small. This proves that the class-3 marker of length m has the best asymptotic performance among all markers of the same length, because $V_1(s)$ is as large as possible and $V_2(s)$ is as small as possible for the class-3 marker. However, we already have seen that a marker's asymptotic performance might dominate only for large enough n that it would be better to use a longer marker.

In general, it does appear that, for good performance, $V_2(s)$ should be as small as possible, no matter what the block length. Intuitively, this is because each element of $V_2(s)$ indicates a possible way, in the few bits following an inserted uninformative bit, that a new insertion could be needed. Although the members of $V_1(s)$ improve the asymptotic performance, they also increase the chance that bits need to be inserted near the beginning and end of a block. These effects seem to balance at a block length of about $3 \times 2^{m-1}$. However, at that block length, it is better to use a longer marker. This observation is similar to the more specific conclusion in Theorem 3(a) that a class-2 marker of length m never can be globally optimum for $n > 2^m + (m-5) < 3 \times 2^{m-1}$. Thus, since the larger start and end costs for markers with large $V_1(s)$ seem to outweigh the effects of the smaller asymptotic growth rate, we conclude that the most useful markers probably have both $V_1(s)$ and $V_2(s)$ as small as possible.

In addition to the result in Theorem 6 about the asymptotic growth rate of $I_n(s)$ for arbitrary sync markers s, we have the following stronger conjecture that generalizes to arbitrary markers the exact expressions derived in Theorem 2 for the three special cases.

Conjecture 1. *Let s be a length-m sync marker that is not catastrophic. Let*

$$a(s) = 2^{m-1} - 1 + \sum_{i \in V_1(s)} 2^{i-1} - \sum_{i \in V_2(s)} 2^{i-1}$$

and

$$b(s) = \frac{1}{a(s)} \left(-1 + 3 \sum_{i \in V_1(s)} 2^{i-1} \right) + \frac{1}{a(s)^2} \left(-(m-2)2^{m-1} - \sum_{i>1 : i \in V_1(s)} (i-2)2^{i-1} + \sum_{i>1 : i \in V_2(s)} i \times 2^{i-1} \right)$$

and

$$p_s(x) = x^{m-2} + \sum_{i>1 : i \in V_1(s) \cup V_2(s)} x^{i-2} - \sum_{i \in V_2(s)} x^{i-1}$$

Then for all $n > 0$,

$$I_n(s) = \frac{n}{a(s)} + b(s) + c_n(s) \times 2^{-n} \tag{3}$$

where the term $c_n(s)$ *is* $O(|\lambda|^n)$ *and* λ *is the largest-magnitude zero of* $p_s(x)$. *Note that* $|\lambda|$ *is often* 0 *(in which case* $c_n = 0$ *when* $n > m - 2$*) or* 1 *(in which case, the* c_n *sequence is periodic for* $n > m - 2$*). Otherwise,* $|\lambda|$ *is a number slightly larger than* 1 *(probably the largest it can be is* $|\lambda| \approx 1.46577$*, which occurs for* $s = 101100$*). Thus, in all cases, the last term of Eq. (3) decays to* 0 *rapidly as* n *increases. The characteristic polynomial of the* $(m-1)$*-state modified state transition matrix is* $(x-1)p_s(x/2)$.

If $I_n(s)$ *is known exactly for* $1 \le n \le m-2$*, then the values of* $c_n(s)$ *can be computed for* $n \ge 1$ *by*

$$c_n(s) = 2^n \left(I_n(s) - \frac{n}{a(s)} - b(s) \right), \qquad (1 \le n \le m-2) \tag{4}$$

and

$$c_n(s) = - \sum_{i>1 : i \in V_1(s) \cup V_2(s)} c_{n-m+i} + \sum_{i \in V_2(s)} c_{n-m+1+i}, \qquad (n > m-2) \tag{5}$$

We have verified that Eq. (3), using Eqs. (4) and (5), holds exactly for all sync markers of length $m \le 15$ (for $1 \le n \le 50$), so the conjecture appears to be highly likely to be true. Due to time constraints, we have not attempted to find a proof.

F. The Best Sync Markers

Using the general methods described in Section III.A, we computed the average data expansion, $X_n(s)$, for representative sync markers from every equivalence class for every marker length from 2 to 12. Figure 6 shows the lowest average expansion, $\min_{s:|s|=m} X_n(s)$, for markers of each given length m. Each of the curves in Fig. 6 consists of no more than three nearly linear segments. As an example, the $m = 7$ curve starts out, for small n, according to the performance expression in Theorem 2(b) for a class-1 marker of length 7. It continues until a nearly imperceptible inflection point at $n = 105$, where it transitions to the expression in Theorem 2(c) for a length-7 class-2 marker. Then it makes a final transition at $n = 196$ to the performance expression in Theorem 2(a) for the class-3 marker. The individual segments are nearly linear, reflecting the relative unimportance, except at very small n, of the final term, $c_n(s)2^{-n}$, in the expressions in Theorem 2 for the exact performance of these three classes of markers.

The average data expansion, $\min_s X_n(s)$, of the globally optimum sync marker is given by the lower envelope of the curves plotted in Fig. 6. Figure 7 shows the optimum sync marker as a function of block length.

The lower envelope of the curves in Fig. 6, though piecewise nearly linear, strongly resembles the logarithmic shape predicted by the upper and lower bounds of Theorem 3(b) for optimum class-2 markers. In Fig. 8, we plot the difference between the globally optimum average data expansion and $\log_2 n$,

$$\min_s X_n(s) - \log_2 n$$

and, for comparison, we also show the upper and lower bounds on $\min_m X_n(10^{m-2}1) - \log_2 n$ given in Theorem 3(b). We see that, for large n, the upper bound is repeatedly and ever more closely approached

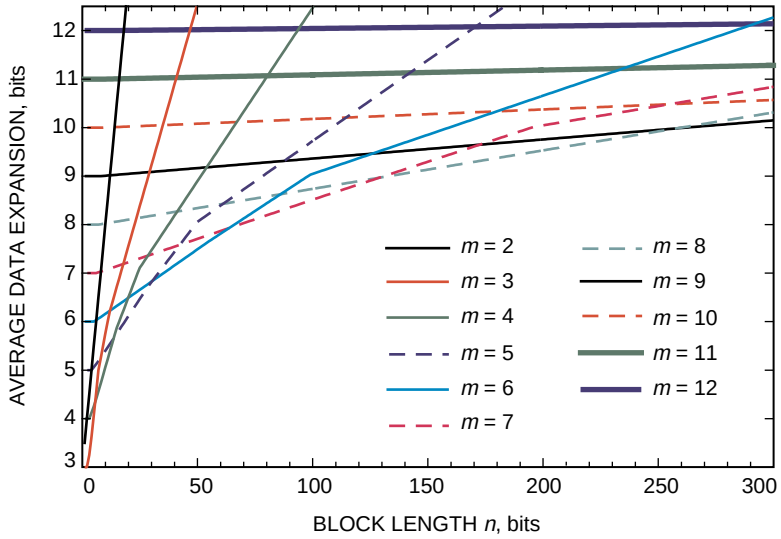

Fig. 6. Minimum average data expansion $\min_{s:|s|=m} X_n(s)$ as a function of block length n, shown for several values of sync-marker length m.

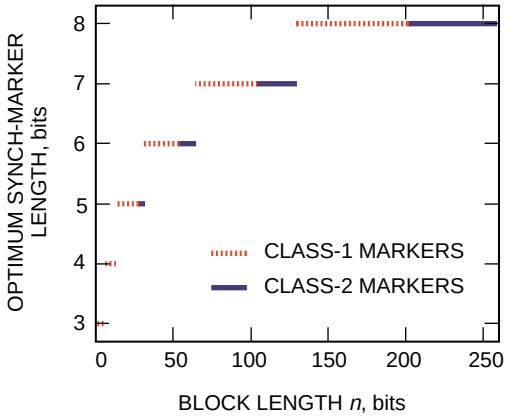

Fig. 7. Optimum sync marker as a function of block length n. The vertical axis indicates sync-marker length $|s|$.

at cusps occurring approximately at powers of 2. The lower bound, derived for class-2 markers only, is breached for small values of n, but, as n increases, it becomes a better approximation to the local minima that occur repeatedly between each pair of locally maximum cusps. Thus, for large n, the globally optimum average data expansion is confined to an extremely tight range of values between $\log_2 n + 1.9$ and $\log_2 n + 2$. At smaller values of n, a class-1 marker more frequently is optimum, but still the optimum $X_n(s)$ only drops as low as $\log_2 n + 1.625$ at $n = 4$.

G. Data Expansion When the Block Size Is Random

Up to now, our analysis has treated the size n of a compressed block as fixed. But if n were really fixed, there would be no need to insert sync markers after every block! In this section, we briefly discuss how the analysis can be easily adapted to the case when the compressed block size n is randomly varying.

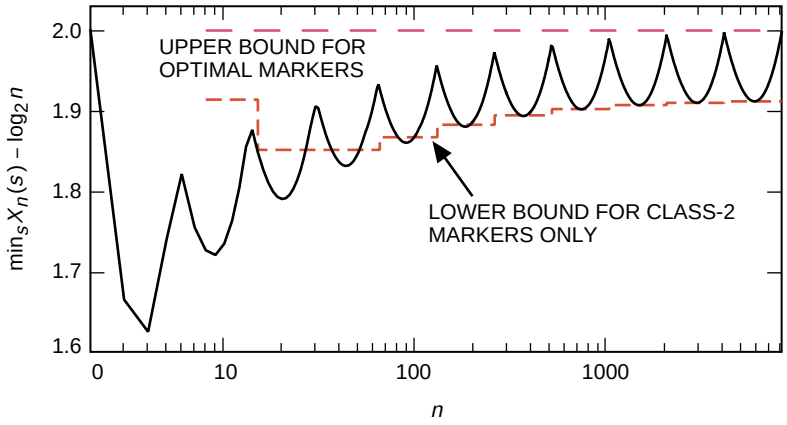

Fig. 8. Globally minimum average data expansion, $\min_s X_n(s)$, as a function of block length n, plotted as a difference relative to $\log_2 n$.

From Theorem 2 for the three special sync-marker classes, and from Conjecture 1 for more general sync markers, we see that the formula for the average data expansion, $X_n(s)$, varies nearly linearly with block size n if the sync marker is fixed. Thus, defining $\overline{X}_n(s)$ as the average (over n) data expansion for a given marker, s, we see that $\overline{X}_n(s)$ may be approximately obtained by formally substituting $\overline{n} = E[n]$ for n in the previously derived expressions for $X_n(s)$, i.e.,

$$\overline{X}_n(s) \approx X_{\overline{n}}(s)$$

(ignoring the technicality that $X_n(s)$ is actually defined only for integer values of n). If our objective is now to choose a sync marker s to minimize $\overline{X}_n(s)$, the solution is essentially the same as before, in terms of $\overline{n}$ instead of n.

When the block size is random, there is a counterpart to Fig. 6 that would look very much the same (except for very small values of average block size), as long as the reference point for the average data expansion is understood to be $\log_2 \overline{n}$. In other words, for large enough $\overline{n}$, the average data expansion for a marker optimally selected to contend with a randomly varying block size n will oscillate between $\log_2 \overline{n} + 1.9$ and $\log_2 \overline{n} + 2$.

IV. System Performance

In our analysis of error containment so far, we have focused primarily on minimizing the expected number of inserted bits. We now turn our attention to the effects of error containment on overall communication-system performance.

In error containment, just as in channel coding, we are transmitting redundant bits in return for increased protection from errors. The combination of data compression and error containment affects both the rate and reliability with which we communicate source symbols and, thus, influences the signal-to-noise ratio (SNR) required to achieve a given level of reliability. So, once we define a suitable error metric, we can evaluate the trade-off between SNR and error performance for data compression combined with error containment, as commonly is done in evaluating channel-code performance.

Ignoring for the moment the impact on reliability, the effect of the data-compression stage on system performance is to increase the rate at which source symbols are communicated, thereby reducing the required SNR. For example, a 2:1 compression ratio results in a 3-dB gain in SNR. This gain is partially

offset by a reduction in rate from overhead associated with the error-containment stage, as we have seen in the preceding sections.

From this perspective, we can isolate the effect of the compression stage simply by knowing the compression ratio provided. This is made possible by our assumption that compression performance is independent of source block length.[4] Thus, given a channel model and error metric, we can concentrate on analyzing the effects on system performance of the error-containment stage alone. In this section, we consider two channels and identify two candidate error metrics.

A. Optimum Sync Marker and Block Length

For a given noisy channel and error metric, we wish to choose the sync marker and source block length to optimize the trade-off between SNR and error performance. Our error-containment scheme allows us to select any (noncatastrophic) sync marker, but we have less direct control over the information block length, n, which is a random variable equal to the number of compressed bits from the source encoder for a block of source symbols. Control over n is accomplished by choosing the number of source symbols, B, for each block to produce the desired expected value of n. For now, we will analyze performance treating n as a parameter that can be fixed exactly. In Section IV.D, we will see that the results do not change much when n is a random variable.

Using sync marker s for a block of n information bits, we transmit on average a total of $n + |s| + I_n(s)$ bits, so the rate is

$$r = \frac{n}{n + |s| + I_n(s)} = \frac{n}{n + X_n(s)} \tag{6}$$

Thus, given n, selecting s to minimize average data expansion $X_n(s)$ (discussed in Section III) maximizes rate. A lower rate means that more energy is spent transmitting redundant bits, and this is taken into account in the subsequent calculations. If $\mathcal{E}_i$ denotes the energy per information bit and $\mathcal{E}_b$ denotes the energy per bit sent to the channel encoder (or to the channel in the case when we omit the channel code), then

$$\mathcal{E}_b = r\mathcal{E}_i \tag{7}$$

If we transmit symbols without data compression, then error containment is not needed, and the rate becomes $r = 1$.

Since our goal is the reliable communication of source symbols, a relevant error metric should measure the frequency that symbols or blocks of symbols are corrupted. As a starting point, consider the expected probability that a block of n information bits (and associated sync-marker overhead) encounters an error in transmission. For a binary symmetric channel (BSC) with error probability γ, this is approximately

$$P \approx 1 - (1-\gamma)^{n+|s|+I_n(s)} = 1 - (1-\gamma)^{n/r} \tag{8}$$

A BSC arises, for example, on the uncoded hard decision additive white Gaussian noise (AWGN) channel. A similar dependence on n and r occurs when the (7,1/2) convolutional code is used on the AWGN channel. For the uncoded channel, the channel-error probability γ decreases with increasing $\mathcal{E}_b$. Thus, given n, we see from Eq. (7) that selecting the sync marker to maximize rate will minimize channel-error probability

[4] Many compression schemes offer higher efficiency for longer block lengths, and it would be possible to extend the analysis to take this into account if a good model for this dependence is available.

γ and, from Eq. (8), this choice also minimizes the block-error probability. Thus, under this error metric, when n is fixed, the sync marker that maximizes rate is nearly optimum.

As we will see shortly, for other error metrics, the dependence on sync markers may not simply reduce to a dependence on rate, but in all the cases considered, this rate effect dominates the choice of sync marker when n is fixed, and choosing the sync marker to maximize rate is nearly optimal. So to a good approximation, our problem reduces to choosing block length n.

Larger block lengths lead to a higher rate, but longer blocks are harder to transmit without error. As might be expected, we will see in the following sections that longer block lengths generally are preferred at the higher SNRs necessary when higher reliability is required.

B. Error Metrics

In the following subsections, we will evaluate the cost of using sync markers for the AWGN channel uncoded and with the (7,1/2) convolutional code. We will consider two error metrics:

(1) We might be interested only in blocks of source symbols that the decoder correctly certifies to be error-free. So a reasonable metric is

$$P_{\mathrm{b}} = \text{ fraction of blocks correctly labeled error free}$$

At the receiving end, a block is declared to be error free when the decompressor observes the sync marker after decompressing B source symbols. For this to occur, we require an absence of errors in the block and in the sync markers immediately preceding *and* following the block. This last condition makes this metric slightly different from Eq. (8).

(2) In some applications, we might be interested in maximizing the number of source symbols (rather than blocks) that is correctly decoded, even if some of the symbols in the same block are corrupted. So an alternative metric is

$$P_{\mathrm{s}} = \text{ fraction of error free source symbols}$$

For example, if the sync marker preceding the block is error free, and the first half of symbols in a block are error free, but then an error occurs, we assume that half of the source symbols in this block are recovered.

Note that we are measuring the frequency with which symbols or blocks of symbols are corrupted. By contrast, channel-coding-error metrics frequently measure the probability that an individual bit is in error, which is not directly comparable to the metrics used here.

1. The Uncoded AWGN Channel. The uncoded AWGN channel has bit-error probability

$$\gamma = Q\left(\sqrt{\frac{\mathcal{E}_b}{N_0/2}}\right) = Q\left(\sqrt{2r\frac{\mathcal{E}_i}{N_0}}\right)$$

where $Q(x) = (1/2)\mathrm{erfc}(x/\sqrt{2})$. For a given sync marker s and block length n, the expected fraction of blocks in error is approximately

$$P_{\mathrm{b}}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right) = 1 - (1-\gamma)^{n+2|s|+I_n(s)}$$

and the expected fraction of symbols in error is approximately

$$P_{\rm s}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right) = 1 - \left(\sum_{i=0}^{n+I_n(s)-1} \frac{i(1-\gamma)^{|s|+i}\gamma}{n+I_n(s)} + (1-\gamma)^{|s|+n+I_n(s)}\right)$$

$$= 1 - \frac{(1-\gamma)^{1+|s|}}{[n+I_n(s)]\gamma}\left[1-(1-\gamma)^{n+I_n(s)}\right]$$

The summation accounts for the event that the first error occurs after the ith bit transmitted following the sync marker; the last term accounts for the event that no error occurs in the block.

2. The (7,1/2) Convolutionally Coded AWGN Channel. The (7,1/2) convolutionally coded channel is modeled in [4]. In this model, the channel is either in the midst of an error burst or in the "waiting" mode, i.e., not producing errors. The probability of transitioning from waiting to burst mode is ε, and the probability of transitioning from burst to waiting mode is β. These values all depend on the channel SNR.

Empirical data have been collected for the average waiting time, $\overline{W}$, and average burst length, $\overline{B}$ [4]. To evaluate performance at SNRs for which data have not been tabulated, we fit the data using the following (somewhat arbitrary) functions:

$$\hat{W}(x) = -0.853126 + 475.051e^{-3x} - 135.891e^{-2x} + 76.6192e^{-x}$$

$$\hat{B}(x) = \exp\left(0.234571 + 3.5447\,x + 0.479935\,x^2\right)$$

where $10\log_{10} x$ is the signal-to-noise ratio in dB.

The probability of an error-free block equals the probability of starting in the waiting state and remaining in the waiting state during the entire block. Thus, the expected fraction of blocks in error is approximately

$$P_{\rm b}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right) = 1 - P_w \times (1-\varepsilon)^{n+I_n(s)+2|s|}$$

where $P_w = \beta/(\varepsilon+\beta)$ is the probability of being in the waiting state at the start of the block.

The expected fraction of symbols in error is approximately

$$P_{\rm s}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right) = 1 - P_w\frac{(1-\varepsilon)^{1+|s|}}{[n+I_n(s)]\varepsilon}\left[1-(1-\varepsilon)^{n+I_n(s)}\right]$$

3. Uncompressed Data. For comparison, if we have b-bit source symbols, we could transmit the symbols uncompressed and without error containment. Then the effective rate is $r = 1$, and the probability that a symbol is received without error is

$$P_{\rm uncomp}\left(\frac{\mathcal{E}_i}{N_0}\right) = (1-\gamma)^b$$

C. Results

Given a fixed SNR, we would like to choose s and n to minimize each error metric; that is, we wish to evaluate

$$P_{\mathrm{b}}^{*}\left(\frac{\mathcal{E}_i}{N_0}\right) = \min_{s,n} P_{\mathrm{b}}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right)$$

$$P_{\mathrm{s}}^{*}\left(\frac{\mathcal{E}_i}{N_0}\right) = \min_{s,n} P_{\mathrm{s}}\left(\frac{\mathcal{E}_i}{N_0}, s, n\right)$$

Figures 9 and 10 show P_{b}^{*}, P_{s}^{*}, and P_{uncomp} as a function of SNR for the AWGN channel uncoded and with the (7,1/2) code. Note that these graphs illustrate the cost in dB of error containment but do not take into account the SNR gain from compression, which should be included in a system analysis but must be computed separately because it depends on the particular source and compression method.

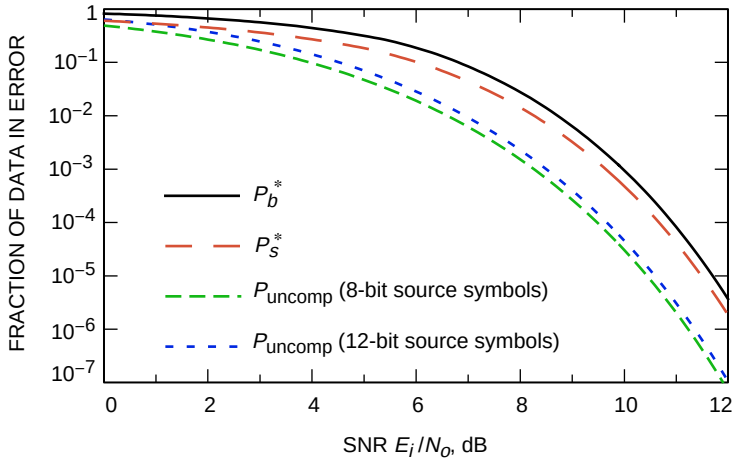

Fig. 9. Error-containment performance on the uncoded AWGN channel.

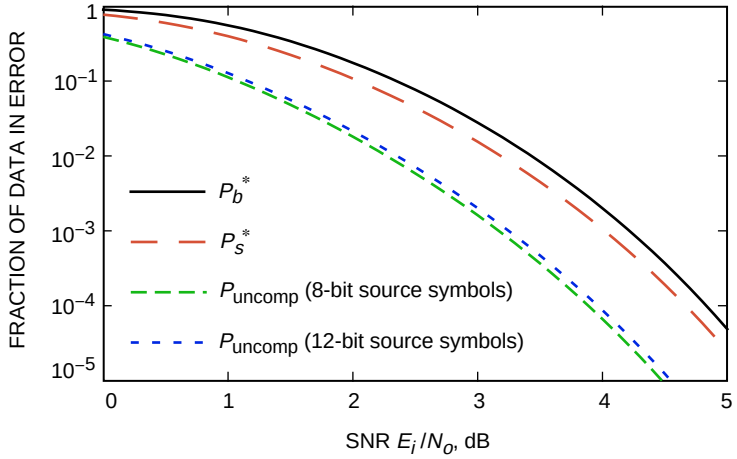

Fig. 10. Error-containment performance on the AWGN channel with the (7,1/2) convolutional code.

An example illustrates how to use the figures. Suppose we use the AWGN channel and (7,1/2) code with 8-bit source data, and a user requires that no more than one symbol in a thousand is in error. In this case, we compare the curves for P_s^* and P_{uncomp} (for $b = 8$) in Fig. 10 to find that, at this target reliability, the error-containment scheme costs about 0.85 dB compared with transmission without compression. This means that, if the proposed compression system achieves, say, a 2:1 compression ratio (i.e., 3 dB of compression gain), then the combined compression and error-containment scheme offers a 2.15-dB improvement in overall system performance.

This example does not illustrate all of the benefits of data compression. Transmitting symbols without compression means that the symbols in error are randomly distributed in the data and may be impossible to detect. When we add data compression and error containment, symbol errors will tend to be clustered within blocks, meaning that longer segments of error-free data will be produced. In addition, the receiver usually will be able to recognize blocks that contain errors, providing some error-detection capability that is not offered when we transmit symbols without compression.

In Fig. 11, we show the block length n that optimizes the SNR and error-performance trade-off for the block error metric P_b. (Compare to Fig. 8, where sync markers are selected to maximize rate.) For both the uncoded and coded channels, the optimum sync marker depends almost entirely on the source block length n, supporting our previous claim that selecting the sync marker to maximize rate is nearly optimal.

We see from Fig. 11 that, compared with the uncoded channel, when we use the (7,1/2) code, we use longer block lengths, and we see from Fig. 10 that the cost of the error-containment overhead is substantially smaller. This is not surprising considering that the code provides lower channel-error probability at a given SNR, and clearly this trend of reduced cost of error containment would be expected to continue for better channel codes.

D. Performance When the Block Size Is Random

Since the block length n is a random variable rather than a fixed parameter, the analysis of the preceding sections is not exact. We have seen that, when n is known, the sync marker that maximizes rate is nearly optimal and that $X_n(s)$ is very nearly linear in n, so a good approximation to rate is found using the expected value $E[n]$ in place of n in Eq. (6). It seems reasonable, therefore, to select the sync marker that maximizes rate when the block length equals $E[n]$.

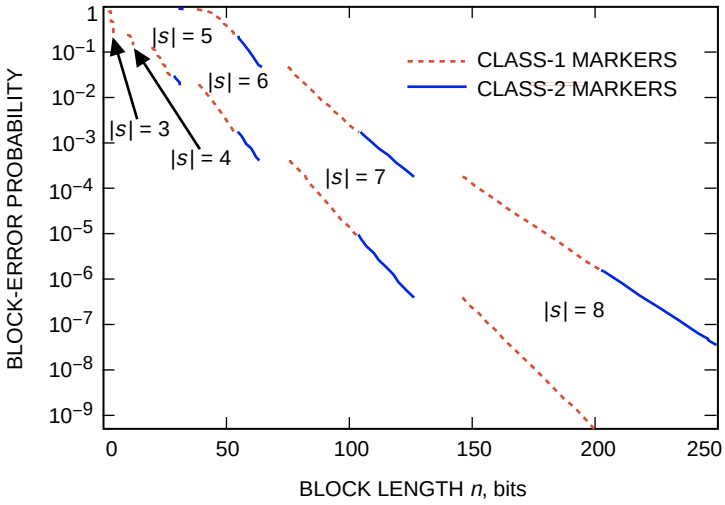

Fig. 11. Sync-marker type, length, and block length that minimize block-error metric P_b on the AWGN channel uncoded (lower curves) and with the (7,1/2) code (upper curves).

We have assumed that the source symbols are i.i.d. random variables, so the compressed lengths of source symbols, $\ell_1, \ell_2, \cdots$, are i.i.d. random variables, and the compressed block length is

$$n = \sum_{i=1}^{B} \ell_i$$

If ℓ_i has expected value μ and variance σ^2, then n has mean $E[n] = B\mu$ and variance $\mathrm{Var}[n] = B\sigma^2$. By the central limit theorem, for sufficiently large values of B, the random variable n is approximated by a Gaussian random variable.

For simulations we assign each value of n using a (quantized) Gaussian random number generator with mean $E[n]$ selected to match the optimizing block length at the desired SNR. (In practice the mean is controlled by adjusting B, the number of source symbols per block.) We simulated the case when $\sigma^2 = \mu/2$, for which $\mathrm{Var}[n] = E[n]/2$. The sync marker selected is the one that gives optimum performance for the expected block length. Figure 12 shows the results of the simulations compared with the curves previously shown in Fig. 10. Each point corresponds to a channel SNR for which the (7,1/2) code has a data point in [4], so the data from [4] were used rather than the curve fit used in the preceding calculations. The simulations and calculations are seen to be in good agreement.

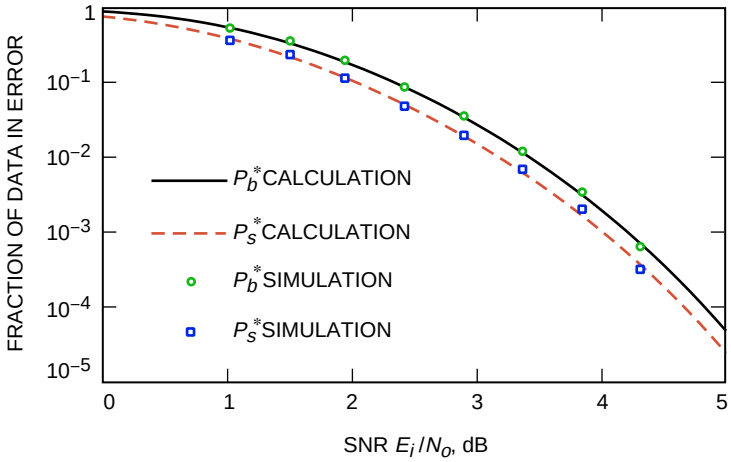

Fig. 12. Error-containment performance on the AWGN channel with the (7,1/2) code, comparing performance calculation treating *n* as fixed (curves) and Gaussian-distributed *n* (points) with $\sigma^2 = \mu/2$.

V. Conclusion

We conclude with a discussion of potential extensions of this work. Improved performance might be obtained by modifying the source encoder design to take into account the sync markers to be used. We have assumed so far that zeros and ones are equally likely from the source encoder, but good source codes such as Huffman codes typically contain small amounts of redundancy, so it is plausible that the probability of a zero might not be exactly 1/2. In this case, Eq. (1) still can be used to calculate the expected number of inserted bits, but transitions out of each state are not equally likely, so entries in the transition matrix **A** are no longer limited to values of 0, 1/2, and 1. Unfortunately, the equivalence classes of sync markers depend on zeros and ones being equally likely. However, the three representative sync markers we have identified as being candidates for optimizing performance all consist primarily of a run of the same symbol. So it seems likely that in any case the representatives from these three classes

would still be most desirable, and the cost of the error-containment strategy should be smaller when zeros and ones are not equally likely (provided, of course, that we know which symbol is more likely). So if it did not hurt compression efficiency, we might want to maximize the probability of a one. The minimum redundancy prefix-free code that maximizes the probability of a one cannot always be obtained from a Huffman algorithm. It would be of interest to develop a similar algorithm that finds such a code.

It seems reasonable to expect that further gains are possible using a more sophisticated method of optimizing the source encoder to take advantage of the details of the error-containment strategy. Future analyses could take into account the fact that practical sources frequently do not produce i.i.d. outputs and more efficient compression generally is possible as we increase the source block length.

A more accurate picture of the value of error containment could be obtained through alternative error metrics that consider the effect of the distribution and detection of errors on the usefulness of the data. We also might want to examine the distortion between the source and reproduced symbols. It also would be of interest to develop a metric that measures total science content returned rather than the fraction of data that is accurate.

Based on the results of Section IV, we can infer how performance might change for other codes on the AWGN channel. We have seen that on the coded channel we use longer block lengths, and the cost of the error containment overhead is substantially smaller. This trend of increasing benefit of error containment would be expected to continue for better channel codes. However, the specific error-containment strategy that we would use might be different for long block codes, such as Reed–Solomon codes or turbo codes. Since decoding for these codes is performed in blocks, we might want to employ an error-containment strategy that is closely integrated with the channel code. For example, the Reed–Solomon decoder usually correctly decodes the received codeword or declares failure, but almost never makes a codeword error. Thus, if we used our current sync-marker strategy, there would be no need to transmit a sync marker after the first sync marker within a codeword.

Finally, since error containment is in a sense an error-control code, it would be interesting to compare the amount of redundancy invested in error containment with the amount of redundancy invested in the error-correcting code used on the channel.

Appendix A
Equivalent Sync Markers

In this appendix, we will state some useful results on v-classes and overlap sets, and we will conclude with a proof of Theorem 1 of Section III.B.

We use $s = s_1 \cdots s_m$ and occasionally $t = t_1 \cdots t_m$ to denote length-m sync markers. In this appendix, we assume the first bit of all sync markers is a 1. (This assumption does not lose generality since inverting all bits of a sync marker does not change its performance.) In addition to $V_1(s)$ and $V_2(s)$ (defined in Section III.B), it is convenient to define $V_3(s)$ as the overlap set of $s_1 s_2 \cdots s_{m-1}$ (s with the last bit dropped).

The first theorem below follows easily from the definitions of $V_1(s)$ and $V_2(s)$:

Theorem A-1. *For a sync marker s, (a) the sets $V_1(s)$ and $V_2(s)$ are always disjoint and (b) exactly one of $V_1(s)$ and $V_2(s)$ contains* 1.

Theorem A-2. *For a sync marker s, the following relations hold:*

$$\left.\begin{aligned} V_1(s) &= \{i : (i = 1 \text{ or } i-1 \in V_3(s)) \text{ and } i \notin V_2(s)\} \\ V_2(s) &= \{i : (i = 1 \text{ or } i-1 \in V_3(s)) \text{ and } i \notin V_1(s)\} \\ V_3(s) &= \{i : i > 0 \text{ and } i+1 \in V_1(s) \cup V_2(s)\} \end{aligned}\right\} \tag{A-1}$$

Thus, any two of $V_1(s)$, $V_2(s)$, and $V_3(s)$ determine the third.

Proof. If $i > 1$ and $i \in V_1(s)$, then $i - 1 \in V_3(s)$. Similarly, if $i > 1$ and $i \in V_2(s)$, then $i - 1 \in V_3(s)$. Furthermore, if $i \in V_3(s)$, then either $i+1 \in V_1(s)$ or $i+1 \in V_2(s)$. The theorem follows from these facts and Theorem A-1. ❒

Formally, we define two length-m sync markers, s and t, to be in the same v-class if $V_1(s) = V_1(t)$, $V_2(s) = V_2(t)$, and $V_3(s) = V_3(t)$. Theorem A-2 immediately implies Corollary A-1.

Corollary A-1. *Let s and t be length-m sync markers. Then s and t are in the same v-class if and only if $V_i(s) = V_i(t)$ for any two choices of i from $\{1, 2, 3\}$.*

In light of Corollary A-1, we choose to characterize a v-class by the ordered pair $(V_1(s), V_2(s))$. (Note that m also must be specified to describe a v-class.)

We mention some simple properties of v-classes.

Theorem A-3. *Suppose $s = 1 s_2 s_3 \cdots s_m$ and $t = 1 t_2 t_3 \cdots t_m$ are in the same v-class. Let $s' = 1 s_2 s_3 \cdots s_{m-1} \overline{s}_m$ and $t' = 1 t_2 t_3 \cdots t_{m-1} \overline{t_m}$. Then (a) $s_m = t_m$, (b) $s_{m-1} = t_{m-1}$, and (c) s' and t' belong to the same v-class.*

Proof. Part (a) follows from the fact that $1 \in V_1(s)$ if and only if $s_m = 1$. Part (b) follows from the fact that $2 \in V_1(s) \cup V_2(s)$ if and only if $s_{m-1} = 1$. Part (c) results from the observation that $(V_1(s'), V_2(s')) = (V_2(s), V_1(s))$ and $(V_1(t'), V_2(t')) = (V_2(t), V_1(t))$. ❒

Note that, in part (c) of Theorem A-3, the v-class containing s' and t' is of course necessarily different from the v-class containing s and t. Parts (a) and (c) yield Corollary A-2.

Corollary A-2. *Length-m v-classes come in pairs corresponding to two v-classes differing from each other only in the last bits of their members; in one of these classes, each last bit is 0, and in the other each is* 1.

Part (b) of Theorem A-3 is mostly a curiosity since it does not divide the classes into pairs.

We now restate the result, originally stated in Section III.B, that motivated the introduction of v-classes.

Theorem 1. *If s and t are length-m sync markers that are members of the same v-class, then s and t are in the same s-class.*

Equivalently, Theorem 1 states that v-classes are subsets of s-classes. To prepare for proving this, we introduce some more concepts.

Let s be a sync marker that is not the class-3 marker or the catastrophic marker. When s is used, the state of the encoder can be classified by whether an uninformative bit has been sent and by the number of bits sent since the initial sync marker or since the last uninformative bit. Specifically, for $i \geq 0$, we define the state A_i to describe the condition in which the initial sync marker s has been sent, i additional information bits have been sent, no uninformative bits have been sent, and it is still to be determined if an uninformative bit needs to be added before the next information bit (or ending sync marker, if all of the information bits have been sent) is sent. That is, in state A_i, the bit stream contains

$$s_1 s_2 \cdots s_m b_1 b_2 \ldots b_i$$

and this stream contains no substrings matching $s_1 \cdots s_{m-1}$ except in the initial $m-1$ bits and possibly in the final $m-1$ bits.

For $i \geq 1$, we define the state B_i to describe the condition in which at least one uninformative bit has been sent, exactly i information bits have been sent since the last uninformative bit, and it is still to be determined if an uninformative bit needs to be added before the next information bit (or ending sync marker, if all of the information bits have been sent) is sent. In state B_i, the last $i+m$ bits in the bit stream are

$$s_1 s_2 \cdots s_{m-1} \overline{s_m} b_1 b_2 \cdots b_i$$

and this string does not contain a substring matching $s_1 \cdots s_{m-1}$ except in the initial $m-1$ bits and possibly in the final $m-1$ bits.

The (infinite) state diagram describing the transitions between these states is shown in Fig. A-1. Each transition corresponds to the transmission of one information bit. The transitions into state B_1 correspond to the transmission of an uninformative bit and an information bit. When all of the information bits have been sent, either the final sync marker is sent or an uninformative bit is sent and the final sync marker is sent (it can be shown that these are the only two possible ending cases). The transition probabilities (including the probabilities of adding a final uninformative bit before the final sync marker) depend on the sync marker. Some of the transitions may have probability zero.

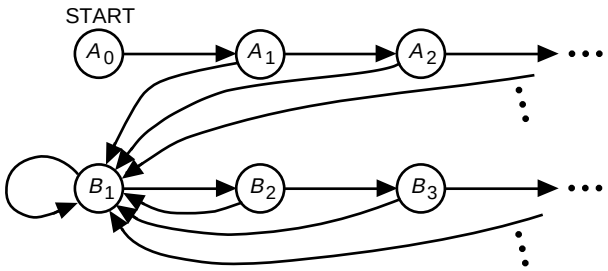

Fig. A-1. The generic state diagram.

In general, the transition probabilities are difficult to compute, so this description is not very useful for computing actual values of $I_n(s)$. However, it is of value because the transition probabilities depend only on the v-class of a marker, as will be shown later.

We define four useful functions. Let $f_s(i)$ be the number of sequences

$$s_2 s_3 \cdots s_m b_1 b_2 \cdots b_i \tag{A-2}$$

that contain no occurrence of $s_1 s_2 \cdots s_{m-1}$. (More precisely, $f_s(i)$ counts the number of choices of $b_1, \cdots, b_i$ for which Eq. (A-2) does not contain $s_1 s_2 \cdots s_{m-1}$.) Let $f_s^*(i)$ be the number of sequences

$$s_2 s_3 \cdots s_m b_1 b_2 \cdots b_i s_1 s_2 \cdots s_{m-1}$$

that contain no occurrence of s and no occurrence of $s_1 s_2 \cdots s_{m-1}$ that is entirely preceding the last information bit b_i. Let $g_s(i)$ be the number of sequences

$$s_2 s_3 \cdots s_{m-1} \overline{s_m} b_1 b_2 \cdots b_i$$

that contain no occurrence of $s_1 s_2 \cdots s_{m-1}$. Let $g_s^*(i)$ be the number of sequences

$$s_2 s_3 \cdots s_{m-1} \overline{s_m} b_1 b_2 \cdots b_i s_1 s_2 \cdots s_{m-1}$$

that contain no occurrence of s and no occurrence of $s_1 s_2 \cdots s_{m-1}$ that is entirely preceding the last information bit b_i.

Lemma A-1. *Let s be a sync marker that is not the class-3 marker or the catastrophic marker. Then the functions f_s, f_s^*, g_s, and g_s^* are all determined by the v-class of s.*

Proof. We first show that $f_s(i)$ can be determined from m, $V_1(s)$, and $V_3(s)$. Let $C_s(i)$ be the set of subsets of $\{1, \cdots, i+1\}$ such that U is in $C_s(i)$ if there exists a sequence $s_2 s_3 \cdots s_m b_1 b_2 \cdots b_i$ such that, for each $j \in U$, the string $s_1 s_2 \cdots s_{m-1}$ appears starting at position j. For example, $C_{1010}(4) = \{\emptyset, \{2\}, \{4\}, \{5\}, \{2,4\}, \{2,5\}\}$, where the inclusion of the sets $\{2,4\}$ and $\{2,5\}$ (and subsets of those sets) can be demonstrated by the strings 0101011 and 0101101, respectively. The set $C_s(i)$ is determined by m, $V_1(s)$, and $V_3(s)$ since the possible positions for the first occurrence of $s_1 s_2 \cdots s_{m-1}$ in such a string are determined by m and $V_1(s)$, and the possible positions of each subsequent occurrence (relative to the previous occurrence) are determined by m and $V_3(s)$.

Now for $U \in C_s(i)$ we define $\beta_s(i,U)$ to be the number of sequences $s_2 s_3 \cdots s_m b_1 b_2 \cdots b_i$ that have the property that the string $s_1 s_2 \cdots s_{m-1}$ occurs starting at each of the positions in U but nowhere else. Observe that $\beta_s(i,U)$ satisfies a relation of the form

$$\beta_s(i,U) = 2^k - \sum_V \beta_s(i,V) \tag{A-3}$$

with k indicating the number of undetermined bits (of the b_j's) in the string $s_2 s_3 \cdots s_m b_1 b_2 \cdots b_i$ when the string is constrained to contain $s_1 s_2 \cdots s_{m-1}$ at the positions in U, and the sum is over those $V \in C_s(i)$ that are strict supersets of U. Observe that Eq. (A-3) can be used to recursively compute $\beta_s(i,U)$, and this involves knowledge of only m and $C_s(i)$. Furthermore, $f_s(i) = \beta_s(i,\emptyset)$. Thus, $f_s(i)$ can be computed from m, $V_1(s)$, and $V_3(s)$.

Similar reasoning can be used to establish that $f_s^*(i)$ can be computed from m, $V_1(s)$, and $V_3(s)$; $g_s(i)$ can be computed from m, $V_2(s)$, and $V_3(s)$; and $g_s^*(i)$ can be computed from m, $V_1(s)$, $V_2(s)$, and $V_3(s)$. We omit the details. ❒

We are now ready to prove Theorem 1.

Proof of Theorem 1. It is clear that the catastrophic sync marker and the class-3 sync marker $111\cdots1$ are each in v-classes by themselves. Thus, we assume that s is a length-m sync marker that is not catastrophic or class 3. As such, the utilization of s may be described by the diagram in Fig. A-1.

There are four types of transition probabilities to consider. When there are still information bits to be sent, the probability of making the transition to A_{i+1} when $i \geq 1$ and the current state is A_i is

$$\Pr(\text{next state is } A_{i+1} | \text{ current state is } A_i) = \frac{f_s(i)}{2f_s(i-1)} \tag{A-4}$$

The number of choices of the first $i-1$ information bits $b_1 \cdots b_{i-1}$ that will cause state A_i to be reached is $f_s(i-1)$; the ith bit b_i does not affect whether state A_i is reached, so the denominator in Eq. (A-4) is the number of choices of the first i information bits that will cause state A_i to be reached. The numerator is the number of choices of the first i information bits that will cause state A_{i+1} to be reached. Equation (A-4) follows because all possible length-i bit sequences are assumed to be equally likely.

Similar reasoning shows that the probability of making the transition to B_{i+1} when the current state is B_i is

$$\Pr(\text{next state is } B_{i+1} | \text{ current state is } B_i) = \frac{g_s(i)}{2g_s(i-1)}$$

When no more information bits are to be sent, the probability that an extra bit does not need to be inserted before the final sync marker is

$$\Pr(\text{no need to insert a bit to finish} | \text{ current state is } A_i) = \frac{f_s^*(i)}{f_s(i)}$$

or

$$\Pr(\text{no need to insert a bit to finish} \mid \text{current state is } B_i) = \frac{g_s^*(i)}{g_s(i)}$$

We have shown that the transition probabilities can be computed from m, f_s, f_s^*, g_s, and g_s^*. Applying Lemma A-1 shows that the transition probabilities can be computed from m, $V_1(s)$, $V_2(s)$, and $V_3(s)$; that is, the transition probabilities depend only on the v-class of s. Clearly, $I_n(s)$ can be computed from the transition probabilities; thus, $I_n(s)$ depends only on the v-class of s. The stated result follows immediately. ❒

Appendix B

Sync-Marker Performance for Special Cases

In this section, we derive empirical expressions for the expected number of inserted bits for the sync-marker classes containing 10^{m-1}, $10^{m-2}1$, and 1^m, which were given in Section III.C as Theorem 2.

I. Class-3 Markers

As shown in Fig. B-1, the states for the class-3 sync marker 1^m are numbered $0, 1, \cdots, m-2$, and one of the two transitions out of state $m-2$ corresponds to an inserted bit. Let $p_i(t)$ denote the probability of the ith state at time t. Here $\delta = 1$ and the expected number of inserted bits at the end is $1/2$, so

$$I_n = \frac{3}{2} + \sum_{i=0}^{n-2} \frac{1}{2} p_{m-2}(i)$$

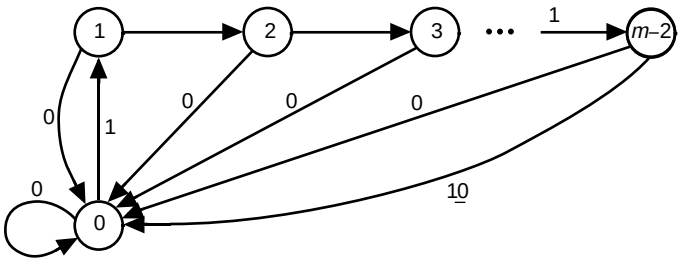

Fig. B-1. State diagram for the class-3 sync marker of length *m*.

The state probabilities satisfy the following relationships:

$$p_i(t) = \frac{1}{2} p_{i-1}(t-1), \quad i > 0$$

$$\begin{aligned} p_0(t) &= \frac{1}{2}\left[p_0(t-1) + p_1(t-1) + \cdots + p_{m-3}(t-1)\right] + p_{m-2}(t-1) \\ &= \frac{1}{2}(1 + p_{m-2}(t-1)) \end{aligned} \tag{B-1}$$

$$\begin{aligned} p_{m-2}(t) &= \frac{1}{2} p_{m-3}(t-1) = \frac{1}{2^2} p_{m-4}(t-2) = \cdots = \frac{1}{2^i} p_{m-2-i}(t-i) \\ &= 2^{2-m} p_0(t-m+2) \end{aligned} \tag{B-2}$$

Using Eq. (B-2), we can rewrite I_n as

$$I_n = \frac{3}{2} + \sum_{i=0}^{n-2} \frac{1}{2} 2^{2-m} p_0(t-m+2) = \frac{3}{2} + 2^{1-m} \sum_{i=0}^{n-m} p_o(i)$$

Combining Eqs. (B-1) and (B-2), we have

$$p_0(t) = \frac{1}{2}\left(1 + 2^{2-m} p_0(t-m+1)\right)$$

This recursion, with initial conditions $p_0(0) = 1$ and $p(t) = 0, t < 0$ can be used to find a closed-form expression for $p_0(t)$. Define $A_i \triangleq p_0(i(m-1))$ and $B_i \triangleq p_0(1 + i(m-1))$. Then $p_0(i)$ takes on values $A_0, \underbrace{B_0, B_0, \cdots B_0}_{m-2}, A_1, \underbrace{B_1, B_1, \cdots B_1}_{m-2}, A_2, \cdots$.

To compute A_i and B_i, note that $A_0 = 1, B_0 = 1/2$, and, for $i > 0$, we have the recursions

$$A_i = \frac{1}{2}(1 + 2^{2-m} A_{i-1}) = \sum_{j=0}^{i-1} \frac{1}{2}\left(\frac{1}{2} 2^{2-m}\right)^j + \left(\frac{1}{2} 2^{2-m}\right)^i = \frac{1 + (1 - 2^{2-m}) 2^{i(1-m)}}{2 - 2^{2-m}}$$

$$B_i = \frac{1}{2}(1 + 2^{2-m} B_{i-1}) = \sum_{j=0}^{i} \frac{1}{2}\left(\frac{1}{2} 2^{2-m}\right)^j = \frac{2^{-1} - 2^{(1-m)(i+1)}}{1 - 2^{1-m}}$$

Now we can express $\sum_{i=0}^{n-m} p_0(i)$ as a sum of A_i and B_i. The largest index of A_i in this sum is $i = q \triangleq \lfloor (n-m)/(m-1) \rfloor$. Each B_i comes in groups of $m-2$; the number of complete groups is $q-1$; and the final group is, in general, incomplete, giving contribution $(n - m - q(m-1))B_q$. Thus,

$$\sum_{i=0}^{n-m} p_o(i) = (n-m-q(m-1))B_q + \sum_{i=0}^{q} A_i + (m-2)\sum_{i=0}^{q-1} B_i$$

$$= 1 + \frac{n-m}{2} + \frac{2(m-1)\left(2^{q(1-m)}-1\right)}{(2^m-2)^2}$$

$$+ \frac{3+n-2m-2^{q(1-m)}[3+n+q-m(2+q)]}{2^m-2}$$

Defining $r = n-m-q(m-1)$ (the remainder when we divide $n-m$ by $m-1$) gives

$$I_n = \frac{3}{2} - \frac{2+(m-2)2^m}{(2^m-2)^2} + \frac{n}{2^m-2} + \frac{1}{2^n}\left(\frac{2^{1+r}\left[(m-3-r)2^m+2(2+r)\right]}{(2^m-2)^2}\right)$$

This expression is valid for $n \geq 1$ and $m \geq 2$.

II. Class-1 Markers

We examine the sync marker $1000\cdots0$ of length $m+1$. Figure B-2 shows the state diagram for this sync marker, with the unusual convention of defining a state with index -1 to help illustrate the similarities with the class-3 marker (see Fig. B-1).

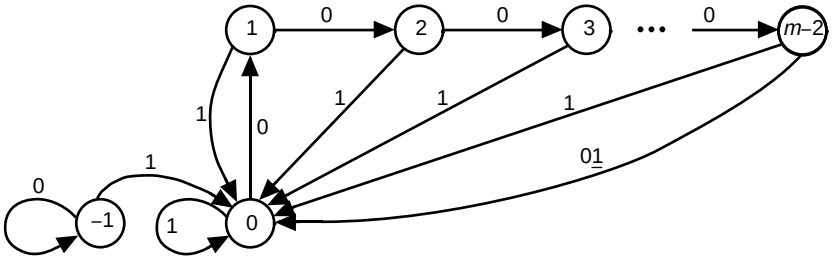

Fig. B-2. State diagram for the length-*m* sync marker 1000 ··· 0.

We find $p_{-1}(t) = 2^{-t}$, and, similar to the analysis for the all-zeros sync marker, we find

$$p_0(t) = \frac{1}{2}(1 + p_{m-2}(t-1)) = \frac{1}{2}(1 + 2^{2-m}p_0(t-m+1))$$

but the initial conditions here are $p_0(0) = 0$, $p_0(1) = 1/2$, and

$$I_n = 2^{1-m}\sum_{i=0}^{n-2} p_0(t-m+2)$$

The analysis is similar to that for a class-3 sync marker, producing *for a length $m+1$ sync marker*,

$$I_n = -\frac{m2^m-2}{(2^m-2)^2} + \frac{n}{2^m-2} + \frac{1}{2^n}\left(\frac{2^{1+\beta}[(m-\beta-1)2^m+2\beta]}{(2^m-2)^2}\right)$$

Here $\beta = n - 1 - (m-1)\lfloor (n-1)/(m-1) \rfloor$, which is the remainder when we divide $n-1$ by $m-1$.

III. Class-2 Markers

The length-m sync marker $100\cdots01$ has the property that $\mathbf{A}^n = \mathbf{A}^{m-2}$ for $n \geq m-2$. In this case, Eq. (1) gives

$$I_n = \begin{cases} \delta + \dfrac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{p}(0), & n = 1 \\ \delta + \dfrac{1}{2}\left(\mathbf{v}^{\mathrm{T}} \sum_{i=0}^{n-2} \mathbf{A}^i + \mathbf{e}^{\mathrm{T}}\mathbf{A}^{n-1}\right)\mathbf{p}(0), & n = 2, 3, \cdots, m-2 \\ \delta + \dfrac{1}{2}\left(\mathbf{v}^{\mathrm{T}} \sum_{i=0}^{m-3} \mathbf{A}^i + [\mathbf{e}^{\mathrm{T}} + (n-m+1)\mathbf{v}^{\mathrm{T}}]\mathbf{A}^{m-2}\right)\mathbf{p}(0), & n \geq m-2 \end{cases}$$

and for the length-m sync marker $1000\cdots01$, this reduces to

$$I_n = \begin{cases} 0, & n \leq m-3 \\ \dfrac{n-(m-4)}{2^{m-1}}, & n > m-3 \end{cases}$$

Appendix C

Derivations of Performance Comparisons Among Different Marker Classes

In this appendix, we derive Theorems 3, 4, and 5, stated in Section III.D.

I. Class-2 Markers Compared for Different Lengths

Comparing $10^{m-2}1$ with $10^{m-1}1$, we find that $X_n(10^{m-1}1) - X_n(10^{m-2}1) = 1 - [n-(m-5)]/2^m$ for $n \geq m-1$, $X_n(10^{m-1}1) - X_n(10^{m-2}1) = 0$ for $n \leq m-3$, and $X_{m-2}(10^{m-1}1) - X_{m-2}(10^{m-2}1) = 1 - 1/2^{m-2}$. Combining these results, we see that $X_n(10^{m-1}1) - X_n(10^{m-2}1) < 0$, i.e., the marker of length $m+1$ outperforms the marker of length m, whenever $n > 2^m + (m-5)$. This condition is valid for all $m \geq 2$.

Our conclusion is that a class-2 marker of length m, namely $10^{m-2}1$, cannot be the optimum marker for $n > 2^m + (m-5)$. Applying similar reasoning to $X_n(10^{m-2}1) - X_n(10^{m-3}1)$, we see also that $10^{m-2}1$ cannot be optimum if $n < 2^{m-1} + (m-6)$. Thus, we conclude that a class-2 marker of length m can be optimum only inside the interval from $n = 2^{m-1} + (m-6)$ to $n = 2^m + (m-5)$.

At the endpoints of its interval of possible optimality, the performance of a class-2 marker is evaluated as $X_n(10^{m-2}1) = m + 1 - (1/2)^{m-2}$ and $X_n(10^{m-2}1) = m + 2 - (1/2)^{m-1}$ for n at the lower and upper endpoints, respectively. Noting that $\log_2 n = m - 1 + \log_2(1 + (m-6)(1/2)^{m-1})$ at the lower endpoint and $\log_2 n = m + \log_2(1 + (m-5)(1/2)^m)$ at the upper endpoint, we see that the average number of extra bits is upper bounded by

$$X_n(10^{m-2}1) < \log_2 n + 2$$

at the upper endpoint if $m \geq 4$, and at the lower endpoint if $m \geq 5$.

Next we find that, inside the interval of possible optimality for $m \geq 4$, $X_n(10^{m-2}1) - \log_2 n$ decreases to a minimum value from its value at the lower endpoint, then rises monotonically to its value at the upper endpoint. This is easily verified by differentiating with respect to n. If n were allowed to vary continuously, this minimum would occur at $n = 2^{m-1}/\ln 2$, and the corresponding minimum value of $X_n(10^{m-2}1) - \log_2 n$ would be $\log_2(2e \ln 2) - (m-4)(1/2)^{m-1}$. Since n is restricted to integer values, this provides a lower bound on the minimum value of $X_n(10^{m-2}1) - \log_2 n$. Summarizing these results, we find that

(1) For all $n \geq 8$, if the length m is chosen optimally, then

$$\log_2(2e \ln 2) - (m-4)\left(\frac{1}{2}\right)^{m-1} \leq X_n(10^{m-2}1) - \log_2 n \leq 2$$

(2) For $n = 1, 2, 3, 4, 5, 6, 7$, if the length m is chosen optimally, then

$$X_n(10^{m-2}1) - \log_2 n = \frac{7}{2}, \frac{11}{4}, 2.415, \frac{9}{4}, 2.178, 2.165, 2.0676$$

respectively.

Since $\log_2(2e \ln 2) \approx 1.9139$, we see that the optimal $X_n(s)$ achievable by class-2 markers $s = 10^{m-2}1$ is tightly upper and lower bounded within a margin of less than 0.1 bit for large n.

II. Class-1 Markers Compared With Class-3 Markers

To compare the class-1 marker of length m with the class-3 marker of length $m-1$, we first note that the first term in the expression for $I_n(s)$ in Theorem 2 is identical for these two markers, because $a(10^{m-1}) = 2^{m-2} - 2 = a(1^{m-1})$. Thus, the difference in the average data expansion for the two markers is

$$X_n(1^{m-1}) - X_n(10^{m-1}) = (m-1) - m + \left[b(1^{m-1}) - b(10^{m-1})\right] + \left[c_n(1^{m-1})2^{-n} - c_n(10^{m-1})2^{-n}\right]$$

Further noting that the denominators are identical in the expressions in Theorem 2 for $b(1^{m-1})$, $b(10^{m-1})$, $c_n(1^{m-1})$, and $c_n(10^{m-1})$, and that the same remainder expression $r_{n-1}(m-2)$ appears in the expressions for $c_n(1^{m-1})$ and $c_n(10^{m-1})$, we can simplify the expression for the difference in the average data expansion to the expression given in Theorem 4.

III. Class-1 Markers Compared With Class-2 Markers

A. Comparing Markers of the Same Lengths

To compare the class-2 marker of length m with the class-1 marker of length m, we start with the expression in Theorem 2(b) for $I_n(10^{m-2}1)$, and we algebraically manipulate the expression in Theorem 2(a) for $I_n(10^{m-1})$ to obtain the average data expansion $X_n(10^{m-1})$ of class-1 markers in the form

$$X_n(10^{m-1}) = m + \frac{1}{2^{m-1}-2}\left[n-(m-1)-\frac{m-2}{2^{m-2}-1}\right] + c_n(10^{m-1})2^{-n}$$

Then defining a common denominator by

$$D \triangleq \left(2^{m-1}-2\right)2^{m-1}$$

and a normalized difference in average data expansion by

$$\Delta \triangleq \frac{1}{2}D\left[X_n(10^{m-1}) - X_n(10^{m-2}1)\right]$$

we calculate that

$$\Delta = n - 3\times 2^{m-2} - (2m-6) - \frac{m-2}{2^{m-2}-1} + Dc_n(10^{m-1})2^{-n-1}$$

To approximately locate the crossover point at $n = n_\times(m) \triangleq 3\times 2^{m-2} + 2m - 6$, we solve for n to make $\Delta = 0$ if the last two terms are ignored, and then explicitly evaluate Δ at $n = n_\times(m)$ and $n = n_\times(m)+1$. First, at $n = n_\times(m)$,

$$\Delta = -\frac{m-2}{2^{m-2}-1} + Dc_n(10^{m-1})2^{-n-1} < 0$$

because it can be shown that the negative term dominates the positive term for n near the crossover point. Then, at $n = n_\times(m)+1$,

$$\Delta = 1 - \frac{m-2}{2^{m-2}-1} + Dc_n(10^{m-1})2^{-n-1} > 0$$

because the magnitude of the negative term does not exceed 1. These results hold for any $m \geq 3$.

B. Comparing Markers of Different Lengths

To compare the class-2 marker of length m with the class-1 marker of length $m+1$, we start with the expression in Theorem 2(b) for $I_n(10^{m-2}1)$, and we manipulate the expression for $I_n(10^m)$ to obtain the average data expansion $X_n(10^m)$ of class-1 markers in the form

$$X_n(10^m) = m + 1 + \frac{1}{2^m-2}\left[n-m-\frac{m-1}{2^{m-1}-1}\right] + c_n(10^m)2^{-n}$$

Then defining

$$D \triangleq (2^m - 2)\, 2^{m-1}$$

and

$$\Delta \triangleq \frac{1}{2} D \left[X_n(10^m) - X_n(10^{m-2}1) \right]$$

we calculate that

$$\Delta = \left(2^{m-2} - 1\right)(-n + 2^m + m - 6) - (m - 5) - \frac{m-1}{2^{m-1} - 1} + Dc_n(10^m)2^{-n}$$

To approximately locate the crossover point at $n = n_+(m) \triangleq 2^m + m - 6$, we solve for n to make $\Delta = 0$ if the last three terms are ignored, and then explicitly evaluate Δ at $n = n_+(m)$ and $n = n_+(m) - 1$. First, at $n = n_+(m)$,

$$\Delta = -(m - 5) - \frac{m-1}{2^{m-1} - 1} + Dc_n(10^m)2^{-n} < 0$$

because it can be shown that the negative terms dominate the positive term for n near the crossover point. Then, at $n = n_+(m) - 1$,

$$\Delta = \left(2^{m-2} - 1\right) - (m - 5) - \frac{m-1}{2^{m-1} - 1} + Dc_n(10^m)2^{-n} > 0$$

because the magnitude of the sum of the negative terms does not exceed the first positive term. These results hold for any $m \geq 5$.

Appendix D
Proof of Theorem 6

In this appendix, we prove Theorem 6 of Section III.E, which is restated below for convenience. We always use $s = s_1 \cdots s_m$ to denote the sync marker in the theorem statement. We use $t = t_1 \cdots t_k$ to denote a more generic binary string of length k. We also use some notation and simple results from Appendix A.

Theorem 6. *Let s be a length-m sync marker that is not the catastrophic marker. Then*

$$\lim_{n\to\infty} \frac{1}{n} I_n(s) = \left(2^{m-1} - 1 + \sum_{i \in V_1(s)} 2^{i-1} - \sum_{i \in V_2(s)} 2^{i-1} \right)^{-1} \tag{D-1}$$

Proof. Consider the $(m-1)$-state modified state diagram, as discussed in Section III.A but without the separate "start" state and with the states labeled $1, \cdots, m-1$, where in state i the last $i-1$ bits transmitted (informative or not) are equal to the first $i-1$ bits of the sync marker. Our plan is to show that the limiting probability distribution of these states gives a probability to state $m-1$ that is twice the right-hand side of Eq. (D-1). An alternate plan might involve considering what sequences of information bits result in an inserted uninformative bit; that plan may possibly yield a simpler proof, but our method produces some interesting identities involving binary strings.

Basic results from the theory of finite-state Markov chains and some simple observations on the state diagram can be used to show that a unique limiting distribution always exists. We omit the details but mention that the observations include the fact that any nontransitive state can be reached from any other state, and that there always exists a nontransitive state with a self-loop. (The only time a transitive state occurs is when $s = 100\cdots00$, in which case only state 1 is transitive but state 2 contains a self-loop. State 1 always contains a self-loop.)

For $1 \le i \le m-1$, let p_i denote the limiting probability of state i. Each transition in the reduced state diagram corresponds to the transmission of one information bit. The only transition that also corresponds to the transmission of an inserted uninformative bit is one of the transitions out of state $m-1$; this transition occurs with probability $1/2$ from state $m-1$. Thus, the limiting fraction of inserted bits satisfies

$$\lim_{n\to\infty} \frac{1}{n} I_n(s) = \frac{p_{m-1}}{2} \tag{D-2}$$

To prepare for computing p_{m-1}, we introduce some notation. For any length-k binary string t and $1 \le i \le k$, define $w_i(t) = \max(\{0\} \cup V_2(t_1 \cdots t_i))$. The number $w_i(t)$ is the largest overlap of $t_1 \cdots t_{i-1}\overline{t_i}$ with itself. Each $w_i(t)$ satisfies

$$0 \le w_i(t) < i \tag{D-3}$$

and in particular $w_1(t)$ is always 0. Also note that, if $i \le j \le k$, then $w_i(t_1 \cdots t_j) = w_i(t)$.

For a noncatastrophic length-m sync marker s, the $w_i(s)$'s are important because they determine the transitions in the reduced state diagram. In particular, if $1 \le i \le m-1$, then there is a transition from

state i to state $w_i(s)+1$. The other transition from state i is to $i+1$ if $i < m-1$ or to $w_m(s)$ if $i = m-1$. Suppose the states have relative limiting probabilities $a_1, a_2, \cdots, a_{m-1}$. (By relative probabilities we mean that $p_j/p_k = a_j/a_k$ when $p_k > 0$.) The relative probability of a state i is equal to half the sum of the relative probabilities of all of the states that have a transition to state i (adding a state's probability twice if both of its transitions are to state i). For state $i+1$ (where $1 \le i < m-1$), this is

$$a_{i+1} = \frac{1}{2}\left(a_i + \sum_{j:w_j(s)=i} \begin{cases} a_j & \text{if } j < m \\ a_{m-1} & \text{if } j = m \end{cases}\right) \tag{D-4}$$

Equation (D-4) becomes less awkward if we define $a_m = a_{m-1}$; we then have

$$a_{i+1} = \frac{1}{2}\left(a_i + \sum_{j:w_j(s)=i} a_j\right)$$

Solving for a_i yields

$$a_i = 2a_{i+1} - \sum_{j:w_j(s)=i} a_j \tag{D-5}$$

Because of Eq. (D-3), Eq. (D-5) is suitable for computing the a_i's recursively: We let $a_{m-1} = a_m = 1$ and apply Eq. (D-5) successively for $i = m-2, m-3, \cdots, 1$. Since the probabilities of all states must sum to 1, we have

$$p_{m-1} = \frac{1}{\sum_{i=1}^{m-1} a_i} \tag{D-6}$$

The denominator in Eq. (D-6) is difficult to work with directly, so we define a related function that is slightly simpler and better behaved. For an arbitrary length-k string t, let $f(t) = \sum_{i=1}^{k} b_i$, where the b_i's can be recursively computed by $b_k = 1$, and for $1 \le i < k$,

$$b_i = 2b_{i+1} - \sum_{j:w_j(t)=i} b_j$$

[note the similarity to Eq. (D-5)]. The b_i's in this calculation are "local" to the particular evaluation of $f(t)$. If t is the length-0 string (the empty string), then $f(t) = 0$.

Now we claim that

$$\sum_{i=1}^{m-1} a_i = f(s_1 \cdots s_{m-1}) - f(s_1 \cdots s_{w_m(s)}) \tag{D-7}$$

(the argument of the second f is the length-0 string if $w_m(s) = 0$). We use b_i's for the local variables used in the calculation of $f(s_1 \cdots s_{m-1})$ and c_i's for those used in the calculation of $f(s_1 \cdots s_{w_m(s)})$. Clearly, $a_i = b_i$ when $w_m(s) < i \le m-1$ since they are produced by essentially the same calculation.

For $1 \le i \le w_m(s)$, it turns out that $a_i = b_i - c_i$. This can be shown inductively, starting at $i = w_m(s)$ and working down to 1. We have

$$\begin{aligned} a_{w_m(s)} &= 2a_{w_m(s)+1} - \sum_{j:w_j(s)=i} a_j \\ &= 2b_{w_m(s)+1} - 1 - \sum_{j:w_j(s_1\cdots s_{m-1})=i} b_j \\ &= b_{w_m(s)} - 1 \\ &= b_{w_m(s)} - c_{w_m(s)} \end{aligned}$$

and for $1 \le i < w_m(s)$, assuming the higher cases hold, we have

$$\begin{aligned} a_i &= 2a_{i+1} - \sum_{j:w_j(s)=i} a_j \\ &= 2(b_{i+1} - c_{i+1}) - \sum_{j>w_m(s):w_j(s)=i} b_j - \sum_{j\le w_m(s):w_j(s)=i} (b_j - c_j) \\ &= 2b_{i+1} - \sum_{j:w_j(s_1\ldots s_{m-1})=i} b_j - 2c_{i+1} + \sum_{j:w_j(s_1\ldots s_{w_m(s)})=i} c_j \\ &= b_i - c_i \end{aligned}$$

Thus,

$$\begin{aligned} \sum_{i=1}^{m-1} a_i &= \sum_{i=1}^{m-1} b_i - \sum_{i=1}^{w_m(s)} c_i \\ &= f(s_1\cdots s_{m-1}) - f(s_1\cdots s_{w_m(s)}) \end{aligned}$$

so Eq. (D-7) holds.

Similar reasoning can be used to show that, for an arbitrary length-$k > 0$ string t,

$$f(t) = 1 + 2f(t_1\cdots t_{k-1}) - f(t_1\cdots t_{w_k(t)}) \tag{D-8}$$

We omit the details. Using Eq. (D-8), we show that (again for $k > 0$),

$$f(t) = 2^{k-1} + \sum_{j\in V_1(t)} 2^{j-1} \tag{D-9}$$

The proof is by induction on k (starting with $k = 1$ this time). The $k = 1$ case is simple to check. For $k > 1$, assuming the lower cases, we have

$$\begin{aligned} f(t) &= 1 + 2f(t_1 \cdots t_{k-1}) - f(t_1 \cdots t_{w_k(t)}) \\ &= 1 + 2\left(2^{k-2} + \sum_{j \in V_1(t_1 \cdots t_{k-1})} 2^{j-1}\right) - f(t_1 \cdots t_{w_k(t)}) \\ &= 1 + 2^{k-1} + \sum_{j \in V_3(t)} 2^j - f(t_1 \cdots t_{w_k(t)}) \end{aligned} \tag{D-10}$$

where we have used Eq. (D-8) and then the induction hypothesis. At this point, we note that Theorem A-2 implies that

$$1 + \sum_{j \in V_3(t)} 2^j = \sum_{j \in V_1(t)} 2^{j-1} + \sum_{j \in V_2(t)} 2^{j-1} \tag{D-11}$$

Applying Eq. (D-11) to Eq. (D-10) gives

$$f(t) = 2^{k-1} + \sum_{j \in V_1(t)} 2^{j-1} + \sum_{j \in V_2(t)} 2^{j-1} - f(t_1 \cdots t_{w_k(t)}) \tag{D-12}$$

If $w_k(t) = 0$, then $V_2(t) = \emptyset$, so the last two terms of Eq. (D-12) are zero. Otherwise, by the induction hypothesis, we have

$$f(t_1 \cdots t_{w_k(t)}) = 2^{w_k(t)-1} + \sum_{j \in V_1(t_1 \cdots t_{w_k(t)})} 2^{j-1} \tag{D-13}$$

But notice that

$$j \in V_2(t) \iff j = w_k(t) \text{ or } j \in V_1(t_1 \cdots t_{w_k(t)})$$

which with Eq. (D-13) implies that

$$f(t_1 \cdots t_{w_k(t)}) = \sum_{j \in V_2(t)} 2^{j-1} \tag{D-14}$$

Applying Eq. (D-14) to Eq. (D-12) yields the desired result, Eq. (D-9). Also note that Eq. (D-14) applies in general.

Now we are ready to compute the final result. We have

$$\left(\lim_{n\to\infty}\frac{1}{n}I_n(s)\right)^{-1} = 2\sum_{i=1}^{m-1} a_i$$

$$= 2f(s_1\cdots s_{m-1}) - 2f(s_1\cdots s_{w_m(s)})$$

$$= 2\left(2^{m-2} + \sum_{j\in V_1(s_1\cdots s_{m-1})} 2^{j-1}\right) - 2f(s_1\cdots s_{w_m(s)})$$

$$= 2^{m-1} + \sum_{j\in V_3(s)} 2^j - 2f(s_1\cdots s_{w_m(s)})$$

$$= 2^{m-1} - 1 + \sum_{j\in V_1(s)} 2^{j-1} + \sum_{j\in V_2(s)} 2^{j-1} - 2f(s_1\cdots s_{w_m(s)})$$

$$= 2^{m-1} - 1 + \sum_{j\in V_1(s)} 2^{j-1} - \sum_{j\in V_2(s)} 2^{j-1}$$

where the first equality follows from Eqs. (D-2) and (D-6), the second from Eq. (D-7), the third from Eq. (D-9), the fifth from Eq. (D-11), and the last from Eq. (D-14). Thus, Eq. (D-1) holds and the proof is complete. ❒

References

[1] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for Sliding Block Codes: An Application of Symbolic Dynamics to Information Theory," *IEEE Transactions on Information Theory*, vol. IT-29, pp. 5–22, 1983.

[2] K.-M. Cheung, M. Belongie, and K. Tong, "End-to-End System Consideration of the Galileo Image Compression System," *The Telecommunications and Data Acquisition Progress Report 42-126, April–June 1996*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–11, August 15, 1996.
http://tmo.jpl.nasa.gov/tmo/progress_report/42-126/126E.pdf

[3] E. N. Gilbert, "Synchronization of Binary Messages," *IRE Transactions on Information Theory*, vol. IT-6, pp. 470–477, 1960.

[4] R. L. Miller, L. J. Deutsch, and S. A. Butman, *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, JPL Publication 81-9, Jet Propulsion Laboratory, Pasadena, California, September 1, 1981.

[5] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[6] J. J. Stiffler, *Theory of Synchronous Communications*, Englewood Cliffs, New Jersey: Prentice-Hall, 1971.