An Optimum Buffer Management Strategy for Sequential Decoding

J. W. Layland
Communications Systems Research

Sequential decoding has been found to be an efficient means of communicating at low undetected error rates from deep space probes, but another failure mechanism known as erasure or computational overflow remains a significant problem. The erasure of a block occurs when the decoder has not finished decoding that block at the time that it must be output.

The erasure rate can be unacceptably high even when the decoder is spending over half of its time idly awaiting incoming data. By drawing upon analogies in computer time-sharing, this article develops a buffer management strategy which reduces the decoder idle time to a negligible level, and therefore improves the erasure probability of a sequential decoder. For a decoder with speed advantage of 10 and buffer size of 10 blocks, operating at an erasure rate of 10^{-2}, use of the new buffer management strategy reduces the erasure rate to less than 10^{-4}.

I. Introduction

Convolutional encoding and sequential decoding have been shown to be a useful technique for communicating at low error rates from deep space probes. This coding method has been flown successfully on *Pioneer 9* and will be used on *Pioneers F* and *G*, *Helios*, and possibly many other spacecraft in the future. While low undetected error rates are relatively easy to obtain with this technique, another failure mode, known as computational overflow or erasure, limits operation of the decoder to data rates that are less than half of channel capacity. Increasing the speed of the decoding computations provides some improvement, but only of a linear order. Despite operation with the very high-speed multiple omission sequential decoder (Ref. 1), the frequency of erasures was found to be in excess of two orders of magnitude over undetected errors. Pure decoding speed is not an adequate answer to the erasure problem. Heller (Ref. 2) suggested that more complete knowledge of the erasure mechanism and decoder buffer behavior might allow some improvement in erasure performance.

The investigation described in this article was initiated as a result of the observation that a sequential decoding machine spends much of its time idling, waiting for incoming data, but at the same time is unable to decode an intolerably large fraction of those data blocks. A method was sought and found which reduces decoder idle time to an insignificant level. This increased efficiency of the decoder results in a slight-to-moderate improve-

ment in erasure probability at high-to-moderate erasure rates, and a very marked improvement at low erasure rates.

II. Computation Problem

The sequential decoding of convolutionally encoded data proceeds by making local best estimates of the data sequence, based upon both the received symbol sequence and data estimates preceeding the current one in the block. If the received symbols are relatively noiseless, decoding proceeds rapidly with no searching. If the received symbols are noisy, some of the local estimates will be wrong. To make the ultimate bit-error probability low, the decoder must eventually recognize that an error has been made and search systematically backward through the local estimates to correct those in error. The amount of searching that must be done depends upon the amount of noise in the received data.

A considerable amount of experimental and theoretical work has been expended in determining the statistical behavior of the decoding search (Refs. 2 [pp. 36–45], 3, 4). It has been shown that the number of computations c performed by a sequential decoder in incrementing by one the number of bits for which a local estimate has been made has a Pareto distribution, i.e.,

$$P_r\{c > N\} \sim kN^{-\alpha} \tag{1}$$

The exponent α is a positive increasing function of the bit signal-to-noise ratio ST_b/N_0. The expected value of c becomes infinite for $\alpha \leq 1$. The signal-to-noise ratio at which $\alpha = 1$ is known as the computational cut-off point for sequential decoding.

The constant k in Eq. (1) has been estimated by Heller to be 1.9 (Ref. 2, p. 41). It has been further argued (Ref. 3) that the distribution of computations on an L-bit block of data is linear in L, so that

$$P_r\{c_L > N\} \sim kLN^{-\alpha} \tag{2}$$

is the probability that more than N computations are required to decode a given block of received data. For the remainder of this article, it will be assumed that Eq. (2) defines the per-block distribution of computation.

III. Lower Bound to Erasure Probability

Suppose a sequential decoder has been implemented which performs μ computations during the time one bit is received, has an infinitely large buffer to hold data waiting to be decoded, and a "magic genie" which informs the decoder before it begins decoding which blocks it should immediately erase in order to be able to decode the largest possible fraction of all blocks received. It is clear that in performing this task, the genie simply erases all blocks whose number of required computations exceeds a threshold T_0, and no blocks which require less than T_0. For if one block were erased which required fewer computations than one which was decoded, we could reverse the positions of these two blocks, decode the easier of the two, and have some number of decoder operations available to complete the decoding of one or more other blocks. This contradicts the known condition that the magic-augmented decoder is already decoding the largest possible fraction of the received data, and hence proves the correctness of our assessment of the genie's selection method.

Knowing the selection method and the distribution of computation of the data, we are able to compute the fraction of data erased by the genie. On the average, the number of computations available to decode each block is $\mu \cdot L$, where μ is the speed advantage. The number actually expended is

$$\int_0^{T_0} N dP(N)$$

where $P(N) = P_r\{c_L > N\}$. Since the decoder is doing as well as possible, it must almost always be busy, hence,

$$\int_0^{T_0} N dP(N) = \mu \cdot L \tag{3}$$

Note that $dP(N) = 0$ for $k \cdot L \cdot N^{-\alpha} > 1$, or $N < N_0$. Equation (3) can be easily and mechanically solved first for T_0, and then for $P(T_0)$, the fraction deleted.

$$P(T_0) = e^{-\mu/k} \qquad \alpha = 1$$

$$P(T_0) = kL\left[(kL)^{(1/\alpha)-1} - \left(\frac{\mu}{k}\right)\left(1 - \frac{1}{\alpha}\right)\right]^{\alpha/(\alpha-1)} \qquad \alpha \neq 1 \tag{4}$$

$P(T_0)$ can be made zero by increasing μ for all $\alpha > 1$; while $\alpha = 1$, $P(T_0)$ is exponentially decreasing in μ. $P(T_0)$ represents a lower bound to the erasure for sequential decoding which depends only upon the speed of the decoder and the received ST_b/N_0. This bound can only be

approached if the amount of computation necessary to decode a block is estimated with fair accuracy before decoding is started.

IV. A Realistic Bound to Achievable Performance

Suppose instead of informing the decoder before decoding which blocks required excessive computation, the genie encountered in *Section III* becomes whimsical and halts computation only on noisy blocks after the decoder has expended as much effort toward decoding them as in decoding the noisiest block that does get decoded. Observing that the available computations are fully utilized on the average, we have the equality

$$T_1 P(T_1) + \int_0^{T_1} N dP(N) = \mu L \tag{5}$$

Equation (5) can be solved for T_1 and $P(T_1)$, yielding

$$P(T_1) = e^{-[(\mu/k)-1]} \qquad \alpha = 1$$

$$P(T_1) = kL\left[\frac{\alpha}{1+\alpha}(kL)^{(1/\alpha)-1} - \frac{\alpha-1}{\alpha+1}\frac{\mu}{k}\right]^{\alpha/(\alpha-1)} \qquad \alpha \neq 1 \tag{6}$$

$P(T_1)$ is conjectured to be a lower bound to erasure probability under the condition that no estimate of decoding effort is made on the received data prior to decoding.

V. A Time-Sharing Model

The whimsical genie's performance can be achieved, or at least closely approximated, by some models from computer time-sharing literature. Define each block received as a *job* which must be processed by a computational facility—our sequential decoder. The amount of work required by each job has the Pareto distribution of Eq. (2). Jobs enter the decoder at a fixed rate and with a fixed priority of 0. The priority of a job changes by an amount $-\delta$ for each time unit that it spends being serviced, while the priority remains unchanged while a job is waiting in the buffer. At all times, the highest priority job receives service. If more than one job has the same priority, they share the processor equally. This has been called the "last-come-first-served with pickup" model for time-shared computations (Ref. 5). The priority in this case is the negative of the decoding effort received. Blocks which have received small amounts of decoding effort have a higher priority than those which have received larger amounts, and hence are given preference in receiving additional computation. On the average, all jobs requiring less than some amount T_x of computation will be completed, while those needing T_x or more will accumulate in the infinite buffer after having received T_x units of computation. This behavior is exactly that achieved with the help of the whimsical genie of *Section IV*, hence $T_x = T_1$. One possible disadvantage to decoding in this fashion is that jobs are completed in an order dependent upon the amount of computation required, instead of strictly in the order in which they arrived.

VI. A Practical Memory Management Scheme

The steps necessary to approximate the model of *Section V* with a practical and effective memory management scheme for sequential decoding are: (1) to use a finite buffer size, and (2) to quantize the amounts of computation provided, so that the decoder does not spend all of its time switching between blocks. Details of the scheme are as follows: blocks which have received some decoder effort are stored on a queue in priority order inversely related to the amount of decoder effort which they have received. After a new block is received, the decoder immediately begins decoding upon that new block. The block which the decoder had been working upon is placed in the queue in proper priority order. Whenever a block is completely decoded, its storage is released to be used by some future data block. Whenever a data block arrives for which no buffer space is available, the lowest priority block (not necessarily the oldest) is erased and its space made available for the new block. This algorithm is very similar to the feedback queuing strategy for processor allocation in time sharing (Ref. 5), and so will be called feedback queuing memory management (FBQM) in the following. Figure 1 is a flow chart of the FBQM algorithm.

FBQM has been simulated using the assumption that Eq. (2) is an accurate description of the per-block distribution of required decoder effort. A sequence of uniform (0, 1) pseudo-random variables was generated by a multiplicative-congruential generator and transformed to produce the distribution of Eq. (2). The resultant random variables, each representing a block of convolutionally encoded data, were supplied to a model of the FBQM structure, and simultaneously to a model of the linear buffer memory management traditionally used for sequential decoding. The results for one sequence of tests are shown in Fig. 2. Little improvement is seen at low values of μ and high erasure probabilities, but where performance of the linear buffer management is moder-

ately good, performance of FBQM is up to three orders of magnitude better. The one drawback is, as mentioned earlier, that data blocks are not completed in strictly time sequence but are completed in an order which depends both upon the order of arrival at the decoder and the amount of computation necessary to decode each. While this may be annoying to some, it should not be a significant drawback to deep-space science and video data which receive a large amount of post-flight non-real-time processing anyway, and to which could be added the task of reordering the data into strict time sequence; in the mission control center, perhaps.

VII. Summary

This article has developed both a lower bound to the erasure probability of a sequential decoder with an infinite buffer, and a memory management strategy (FBQM) for decoders with finite buffer which performs close to the bound. Both the erasure probability of FBQM and its lower bound exhibit exponential decrease with increasing decoder speed at the sequential decoding computational cut-off point, where the erasure probability of a conventional sequential decoder exhibits only inverse proportionality. For a decoder with speed advantage $\mu = 40$, the improvement is about 0.2 dB in required ST_b/N_0 at a fixed erasure rate of 10^{-3} or three orders of magnitude decrease in erasure probability at $\alpha = 1$. For a decoder with speed advantage of 10 and buffer size of 10 blocks, operating at an erasure rate of 10^{-2}, use of the new buffer management strategy reduces the erasure rate to less than 10^{-4}.

All of the results presented here depend strongly upon the validity of Eq. (2) as a definition of the per-block distribution of computation for a sequential decoder. Since some of the arguments which support Eq. (2) are asymptotic in nature, and the decoder performance with FBQM depends upon the distribution of computation at both large and small values of the computation variable, additional simulations with actual sequential decoding data will be needed to validate the results obtained.

References

1. Layland, J. W., "Information Systems: Multiple Mission Sequential Decoder—Comparing Performance Among Three Rate 1/2, $K = 32$ Codes," in *The Deep Space Network*, Space Programs Summary 37-64, Vol. II, pp. 50–52. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 31, 1970.
2. Heller, J. A., "Decoding and Synchronization Research: Description and Operation of a Sequential Decoder Simulation Program," in *Supporting Research and Advanced Development*, Space Programs Summary 37-58, Vol. III, p. 42. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 31, 1969.
3. Savage, J. E., "The Distribution of Sequential Decoding Computation Time," *IEEE Trans. Inform. Theory*, Vol. IT-11, pp. 143–147, April 1966.
4. Jacobs, I. M., and Berlekamp, E. R., "A Lower Bound to the Distribution of Computation for Sequential Decoding," *IEEE Trans. Inform. Theory*, Vol. IT-13, pp. 167–174, April 1967. (See also Jacobs, I. M., and Berlekamp, E., "A Lower Bound to the Distribution of Computation for Sequential Decoding," in *Supporting Research and Advanced Development*, Space Programs Summary 37-34, Vol. IV, pp. 270–276. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 31, 1965.)
5. Kleinrock, L., "A Continuum of Time-Sharing Scheduling Algorithms," in *AFIPS Conference Proceedings*, Vol. 36, 1970, pp. 453–458, Spring Joint Computer Conference, Atlantic City, N. J., 1970.

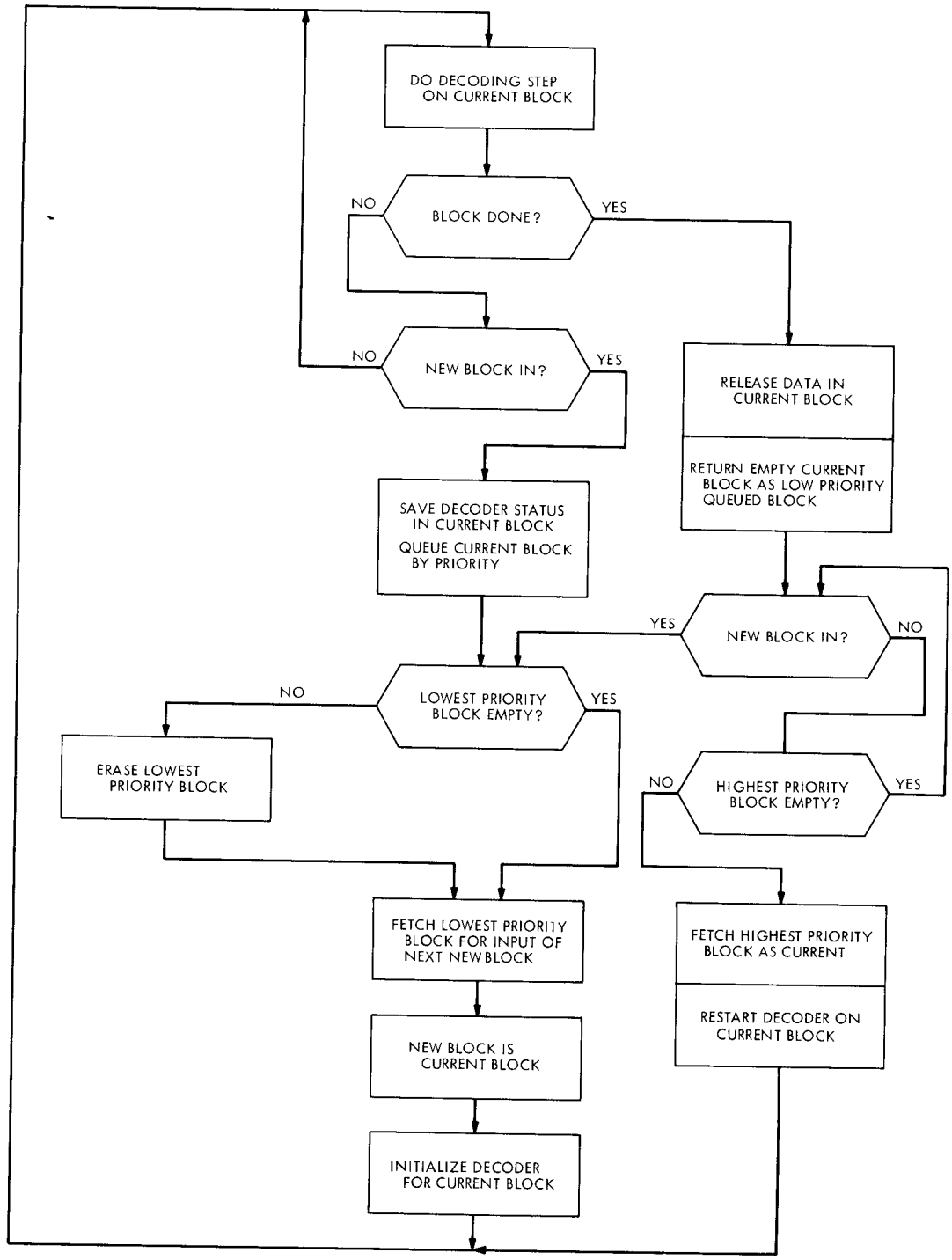

Fig. 1. Flow chart for FBQM

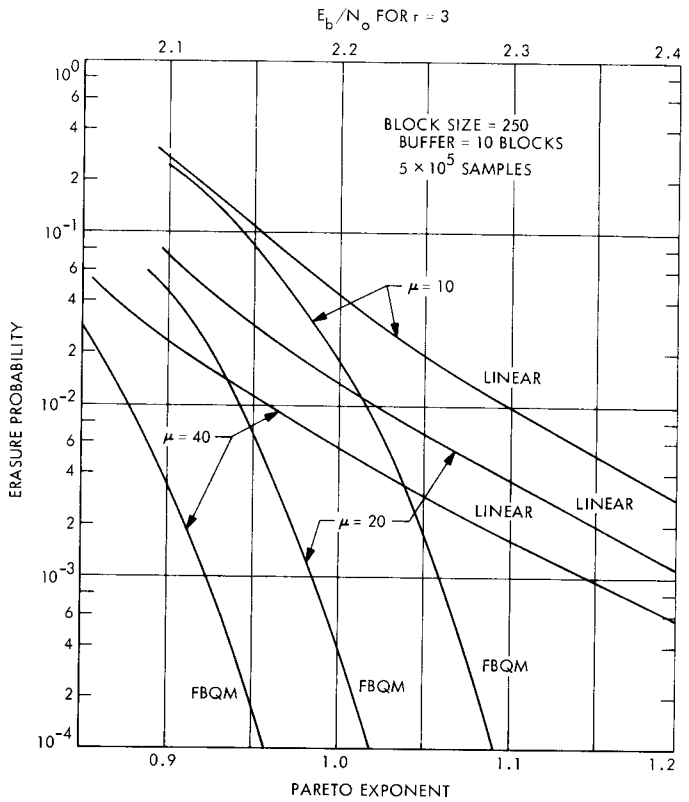

Fig. 2. Comparison of FBQM and linear buffer management