TABLE I
WEIGHT DISTRIBUTION OF (145, 32) CODE

| weight i | number of code words of weight i |
|---|---|
| 0 | 1 |
| 44 | 18,270 |
| 46 | 26,970 |
| 48 | 104,545 |
| 50 | 515,620 |
| 52 | 1,705,200 |
| 54 | 5,115,600 |
| 56 | 13,043,040 |
| 58 | 31,380,620 |
| 60 | 65,578,280 |
| 62 | 124,708,990 |
| 64 | 211,074,325 |
| 66 | 318,312,120 |
| 68 | 430,763,970 |
| 70 | 521,295,880 |
| 72 | 568,126,240 |
| 74 | 549,956,000 |
| 76 | 478,598,890 |
| 78 | 376,022,990 |
| 80 | 262,505,390 |
| 82 | 164,645,180 |
| 84 | 92,201,585 |
| 86 | 46,480,040 |
| 88 | 20,550,560 |
| 90 | 8,034,740 |
| 92 | 2,947,560 |
| 94 | 899,290 |
| 96 | 298,410 |
| 98 | 56,840 |
| 112 | 145 |
| 116 | 5 |

The minimum distance of $C$ is found through a computer to be 44, which is two more than the best code given in [2] as of May 3, 1999. The weight distribution of code $C$ is shown in Table I.

## REFERENCES

[1] W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1972.

[2] A. E. Brouwer, "Bounds on the minimum distance of linear codes," database at URL [Online]. Available WWW: http://www.win.tue.nl/~aeb/voorlincod.html.

# Linear-Time Binary Codes Correcting Localized Erasures

Alexander Barg and Shiyu Zhou

***Abstract*—We consider a communication model over a binary channel in which the transmitter knows which bits of the $n$-bit transmission are prone to loss in the channel. We call this model channel with localized erasures in analogy with localized errors studied earlier in the literature.**

**We present two constructions of binary codes with $t(1+\epsilon)$ check bits, where $t = \alpha n$ is the maximal possible number of erasures. One construction is on-line and has encoding complexity of order $n/\epsilon^4$ and decoding complexity of order $n/\epsilon^2$. The other construction is recursive. The encoding/decoding algorithms assume a delay of $n$ bits, i.e., rely on the entire codeword. The encoding/decoding complexity behaves roughly as $n/\epsilon^2$ and $n/\epsilon$, respectively.**

***Index Terms*—Defects, linear-time codes, localized erasures.**

## I. INTRODUCTION

We consider a communication model in which binary information is transmitted from one party to the other over a channel which can occasionally erase some of the transmitted bits. Information is transmitted by $n$-bit blocks. In contrast to the usual channel with erasures we assume that the sender knows the part of the block where erasures can (but not necessarily will) occur. Therefore, we call such erasures *localized*. Information is transmitted by $n$-bit blocks. Both the sender and the receiver know that at most $t$ out of $n$ bits can be possibly erased in the channel. The $t$ bit positions of possible erasures become known to the sender before transmitting a block and can vary from transmission to transmission. This is the only difference of our model from the standard erasure channel. The receiver, as usual, can only detect that certain bits of the $n$-bit block are lost upon receipt, i.e., erasures are visible at the receiving end.

This problem has a game-like flavor. Namely, supposing that the channel always destroys exactly $t$ bits would make the problem trivial: just send $n - t$ message bits outside these $t$ positions and write anything in bits that will be erased. However, if some of these $t$ bits arrive unscathed, the receiver would read them off and mix with the actual information since it cannot tell the bits that are prone to loss from the reliable bits of the block.

Our correspondence shows that the transmission can be organized with a data overhead of $t(1+\epsilon)$ bits ($\epsilon$ is any positive number), which is essentially the optimum, and that the delays introduced by the sender/receiver are linear in $n$.

Typically, codes correcting erasures are discussed in connection with packet routing in networks employing the asynchronous transfer mode and in the Internet. Generally, the most common causes for the loss of packets are traffic congestion and buffer overflows, which result in excessive delays. If a part of the message is not received within a certain time interval, the receiving node issues a retransmission request. However, in on-line applications such as, for instance, real-time video transmission, retransmission is impossible.

Manuscript received March 8, 1998; revised March 8, 1999. The material in this correspondence was presented in part at the 35th Annual Allerton Conference, Monticello, IL, September 1997.

A. Barg is with Bell Laboratories, Lucent Technologies, 2C-375, Murray Hill, NJ 07974 USA (e-mail: abarg@research.bell-labs.com).

S. Zhou was with Bell Laboratories, Lucent Technologies. He is now with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104-6389 USA (e-mail: shiyu@cis.upenn.edu).

Communicated by F. R. Kschischang, Associate Editor for Coding Theory.
Publisher Item Identifier S 0018-9448(99)07007-8.

In this context we face the problem of retrieving the lost data at the receiving end at the expense of having a certain data overhead in the transmission. Consequently, coding theory solutions for load balancing and decreasing delays in global networks have been a recurrent topic in the literature [3], [8], [12], [13], and [15]. A general idea of these works is the same. Namely, it is suggested to encode the message with maximum distance separable (MDS) codes. If the code has length $n$ and is used to encode $k$ message symbols, then the message can be recovered from any $k$-part of the codeword, and the data overhead is $n-k$ symbols. There are three general methods of constructing MDS codes known: redundant-residue codes, Cauchy matrices, and Reed–Solomon (RS) codes (see [11, Sec. 10.9, Exercise 11.7, and Ch. 10], respectively). These methods are utilized in [3] (redundant-residue codes), [13] (Cauchy matrices), and [8], [15] (Reed–Solomon codes). The complexity of these constructions is of order $n^2$ $q$-ary operations ($q$ is the alphabet size) for encoding and decoding. The encoding is performed either by matrix multiplication or by polynomial evaluations; the decoding is basically Lagrange interpolation for RS codes or Chinese remaindering for redundant-residue codes. The alphabet size $q$ is related to the transmission protocol. In [8], $q$ equals the length of the packet; in [15] it is the number of packets in the image.

In our situation, the sender knows the set $U$ of positions of possible erasures. This information is used in the encoding. More precisely, the encoding mapping, which assigns a codeword to a given message $m$, depends both on the set $M$ of possible messages and the set $U$ of positions of possible erasures. This implies that to every message we associate a subset of codewords, and the code is formed by the union of these subsets. A binary code $C$ of length $n$ corrects $t=\alpha n$ localized erasures if for different messages these subsets are disjoint (then the encoding mapping is invertible). Precise definitions are given in the next section.

Two transmission problems studied previously are close to our problem. The first is called codes for channels with (memory) defects [9]. Under this model, codewords are stored in memory with some cells stuck at either 0 or 1. Therefore, no matter what is written in these bits by the "sender," the "receiver" always reads a fixed value. The second model deals with codes correcting localized errors [5]. The definition is the same as of codes correcting localized erasures except that now the transmitted bits inside a given set $U$ are either received correctly or *flipped* in the channel.

There is a substantial difference between the minimal possible number of check bits in binary codes correcting localized erasures and binary codes correcting usual erasures. Namely, to correct $t$ nonlocalized erasures one needs binary codes with minimum distance at least $t+1$. By the Varshamov–Gilbert bound, the redundancy of the best known binary codes with distance $t\sim\alpha n$ is $nH_2(\alpha)$. On the contrary, the minimum redundancy of a code of length $n$ that corrects $t$ localized erasures is $t$ (see Proposition 2.1).

Our results show that this bound is essentially tight. The same lower bound is valid for codes correcting defects [9]. For localized errors the upper and lower bounds on the number of checks have order $nH_2(\alpha)$ [5]. A more detailed survey of the bounds and complexity results for is given in [4].

As remarked above, for larger alphabets there exist MDS codes that correct $t$ usual erasures with $t$ check symbols. The encoding/decoding time is of order $O(tn)$ for RS codes and $O(n)$ for codes of Alon and Luby [2], which use $t(1+\epsilon)$ check symbols. However, an easy consequence of the Singleton bound [11] shows that the size $q$ of the alphabet of an MDS code must be large, namely, $q\geq\max(t+1,\, n-t+1)$, where $t$ is the number of erasures.

Note also that the assumption of deterministic encoding/decoding methods that ensure recovery of binary sequences with at most $t$ erasures using $t(1+\epsilon)$ check bits; see Luby *et al*. [10].

A number of recursive constructions of low-complexity codes for different transmission models is known in the literature; see Ahlswede *et al*. [1], Alon and Luby [2], Dumer [7], and Spielman [14]. The idea of constructing low-complexity codes for all models is essentially the same. Namely, first the code length $n$ is partitioned into a number of shorter segments. A recursion is applied on each or some of these segments. The recursion stops after reaching constant length or some $o(n)$ length depending in the complexity of codes used on these small segments, so that the encoding and decoding of an optimal code can be implemented by exhaustive search. This general construction varies for different models since optimal codes are constructed in different ways.

This idea was used by Spielman [14] to construct codes of length $n$ and size $2^{n/4}$ correcting a small number $t=\alpha n$ of Hamming errors. The encoding/decoding complexity is of order $O(n)$ and $\alpha\leq 5\times 10^{-7}$. These codes were used as a basis of recursion by Alon and Luby [2], who suggested codes over a very large alphabet correcting $t$ usual erasures. The number of check symbols of this construction is $t(1+\epsilon)$. The complexity of encoding/decoding is $O(n/\epsilon^4)$. Here $\epsilon$ must be very small since the construction relies on error-correcting codes from [14]. In principle, a similar technique can be used in our problem, but the parameters of the construction are sharply restricted by the condition on $\alpha$.

Dumer [7] introduced polynomial-time codes that use $t+o(t)$ checks to correct $t$ defects. The complexity of encoding is $O(n\log^2 n/\epsilon^3)$ and of decoding $O(n\log n/\epsilon)$. In our work we rely on these constructions to construct codes correcting localized erasures. This enables us to achieve linear encoding/decoding complexity for localized erasures. The redundancy of codes is $t(1+\epsilon)$, which is not possible for usual erasures with deterministic encoding/decoding algorithms.

## II. Preliminaries

Let us define formally our problem. Let $\boldsymbol{Z}_2^n$ be the set of all binary words of length $n$. A binary *code* $X$ of length $n$ is any subset of $\boldsymbol{Z}_2^n$. The quantity $k=\log|X|$ is called the *number of message bits* (all log's in the correspondence are base 2). Let $N=\{1,2,\cdots,n\}$. Suppose $x=(x_1,x_2,\cdots,x_n)$ is a codeword submitted to the transmission channel. We say that erasures are localized at $U\subset N$ if the output $y$ of the channel is given by

$$y_i=\begin{cases}x_i, & i\notin U\\ x_i \text{ or } *, & i\in U\end{cases} \tag{1}$$

where $*$ is the erasure symbol. In other words, symbols outside $U$ are never erased in the channel and symbols in $U$ may or may not be erased. Suppose $U$ can be any $t$-subset of $N$, which becomes known to the encoder (but not to the decoder) before transmission and can vary from one block to the other. Then we speak of transmitting over the channel with $t$ localized erasures.

Let $M$, $|M|=2^k$, be the set of messages. By encoding we mean assigning a codeword used to transmit a given message $m\in M$ over the channel with erasures localized at $U$. The encoding mapping is defined by

$$\phi(m,U)\colon M\times\binom{N}{t}\to\boldsymbol{Z}_2^n$$

where $\binom{N}{t}$ is the set of all $t$-subsets of $N$. Then

$$X=\bigcup\left(\bigcup\phi(m,U)\right).$$

Let $\psi$ be a decoding mapping which sends any codeword with at most $t$ bits missing to $M$. By definition, the code $X$ corrects $t$ localized erasures if $\psi(y) = m$ for any string $y$ with

$$y_i = \begin{cases} \phi(m, U), & i \notin U \\ \phi(m, U) \text{ or } *, & i \in U \end{cases}$$

for any $U \in \binom{N}{t}$.

We consider the problem of constructing codes correcting $t$ localized erasures. Below we assume that $n \to \infty$ and $t = \alpha n$, $0 < \alpha < 1$.

*Proposition 2.1:* Let $X$ be a binary code of length $n$ used to transmit $2^k$ messages from a set $M$ over the channel with $t$ localized erasures. Then $n - k \geq t$.

*Proof:* Let $E(\cdot)$ be the erasure operator which maps a codeword $x$ to a sequence $y$ that satisfies (1). A necessary and sufficient condition for $X$ to correct $t$ localized erasures is

$$E(\phi(m', U')) \neq E(\phi(m'', U'')) \tag{2}$$

for any $m' \neq m'' \in M$ and any $U', U'' \in \binom{N}{t}$. Let $U$ be a fixed $t$-subset of $N$. The outcome of the transmission can be any set of erasures on $U$. Assume that $E_U(\phi(m, U))$ erases all $t$ bits in $U$. The binary vector in the remaining $n-t$ bits should satisfy (2); thus $|M| \leq 2^{n-t}$. □

Below we adapt to our problem certain codes correcting defects. The channel with defects can be defined as follows. Let $U \in \binom{N}{t}$ and suppose $x$ is a codeword submitted to a channel with defects. Suppose $U = U_0 \cup U_1$. Then the output of the channel is a word $y$ with

$$y_i = \begin{cases} x_i, & i \notin U \\ 0, & i \in U_0 \\ 1, & i \in U_1. \end{cases} \tag{3}$$

Again we assume that $U$ becomes known to the encoder before the transmission and can vary from block to block.

Let $M$, $|M| = 2^k$, be a set of messages. The code is defined as

$$X = \bigcup_{m \in M} \left( \bigcup_{U \in \binom{N}{t} \times 2^t} \phi(m, U) \right)$$

where $\phi$ is an encoding mapping. Codes for which the encoding mapping is invertible are said to correct $t$ defects.

## III. The Constructions

### A. Overview

We shall present two constructions of binary codes correcting localized erasures. As we have pointed out in the Section I, the problem of correcting localized erasures is much related to the problem of correcting defects [6], [7]. An advantage of locality is that in our coding scheme the encoder knows exactly what the decoder will receive. To achieve this, the encoder forces all the bits that can possibly be erased to be 0's in its encoding, i.e., for any given $m \in M$ and $U \in \binom{N}{t}$ the encoder forces all bits of the codeword $\phi(m, U)$ contained in $U$ to be 0. The decoder replaces all erased bits in the received sequence by zeros. It turns out that in this way the encoder can convey certain useful information (such as the number of possible erasures in a segment) to the decoder in a fairly straightforward manner, which is something difficult to achieve when dealing with usual erasures.

In order to achieve linear time complexity, we propose to break a block of linear size into segments of constant size and apply the complexity of the optimal code is super-linear, it only causes a constant factor increase in the overall linear complexity. The optimal code we shall use is due to Dumer [6]. The original purpose of the code was to deal with asymmetric defects, i.e., defects with $U = U_0$ in (3). It turns out that this construction can be easily reformulated to correct localized erasures. We give the following theorem with proof since it is important for understanding our constructions.

*Theorem 3.1 [6]:* For all $n, t (n > t)$, there exists an easily constructible binary nonlinear code $\mathcal{D}[n, t]$ of length $n$ that corrects $t$ localized erasures. The redundancy of the code is $t+1$. The encoding time of the code is $O(t^3 + tn)$ and the decoding time is $O(tn)$. Moreover, in any codeword, all the bits corresponding to the possible erasures are equal to 0.

*Proof:* Let $U$ be the subset of positions of possible erasures. Let $m$ be the binary message of $n - t - 1$ bits that we want to send. Suppose that $n$ is divisible by $t + 1$, say $n = \nu(t + 1)$ (the opposite case will be treated later). Partition $m$ into segments $m_u$ of length $t + 1$. The encoding is performed by viewing the vectors $m_u$ as elements of $\mathrm{GF}(2^{t+1})$ and multiplying each of them by one and the same element $z \in \mathrm{GF}(2^{t+1})$. Each product $zm_u$ is expanded into a binary vector $c_u$, $1 \leq u \leq \nu - 1$; the codeword of length $n$ is formed by concatenating all these vectors and the binary expansion of $z$ as follows:

$$(c_1, c_2, \cdots, c_{\nu-1}, (z_1, \cdots, z_{t+1})).$$

Element $z$ is chosen so that all the bits of this word with numbers in $U$ be 0. Each product $zm_u$ is a linear form of the variables $z_1, z_2, \cdots, z_{t+1}$. In total, $t$ binary coordinates in these products must be 0. Each of these conditions defines a linear equation with unknowns $z_1, z_2, \cdots, z_{t+1}$. A nontrivial system of $t$ homogeneous equations with respect to $t + 1$ unknowns always has a nontrivial (not all-zero) solution.

Then the message $m$ can be encoded as follows. It is partitioned into segments as described. Then the encoder computes $z$, finds the products $zm_1, \cdots, zm_{\nu-1}$, and their binary expansions $c_1, \cdots, c_{\nu-1}$. This defines the codeword.

Note that $z$ itself is a part of the sequence transmitted and is also subject to erasures. If the set $U$ of possible erasures includes some of the last $t + 1$ coordinates of the transmitted sequence, then the corresponding coordinates of $z$ are simply set to 0.

The decoder sets the erased coordinates to 0. Now the received word exactly equals the codeword. Therefore, the decoder splits it into segments of length $t + 1$ and finds $z \neq 0$. The message is found by computing $z^{-1}m_u$, $1 \leq u \leq \nu - 1$, and taking the binary expansions of these products.

Finally, if $n$ is not divisible by $t + 1$, say $n = \nu(t + 1) + \mu$, then we partition the first part of the message into $\nu - 2$ segments of length $t + 1$ and take the last segment of length $t' = t + 1 + \mu$. All segments except the last are treated exactly as above. The last segment is considered as an element of $\mathrm{GF}(2^{t'})$ and multiplied by $z' = (z, \cdots, z_{t+1}, 0^{\mu}) \in \mathrm{GF}(2^{t'})$. It is easy to see that the construction described above remains valid, and the number of check bits is still $t + 1$. This completes the proof. □

In what follows, we use the term $\mathcal{D}[n, t]$ *encoding/decoding* when we talk about the encoding/decoding of the code.

*Remark:* Note that the $\mathcal{D}[n, t]$ decoding is defined on the whole space $\boldsymbol{Z}_2^n$ except for the subspace $\mathcal{Y}$ spanned by the first $n - t - 1$ coordinates (i.e., subspace of vectors with last $t + 1$ zeros). In

In our constructions we need to extend the definition of $\mathcal{D}[n, t]$ decoding to the entire space $\{0,1\}^n$. Assume by definition that $\mathcal{D}[n, t]$ decoding of any $y \in \mathcal{Y}$ outputs a "message" of $n-t-1$ zeros.

In the first construction that we shall present, the coding scheme is implemented on-line. That is, the codeword is broken into segments each of constant length and the encoding/decoding is applied independently on each segment. The second construction is an extension of the work in [7]. It is a more complicated recursive construction and cannot be implemented on-line, but it has asymptotically better performance in terms of the encoding/decoding complexity.

### B. On-Line Construction

In this section, we present a code construction and prove the following.

*Theorem 3.2:* Given integer $n$ and $\alpha, \epsilon \in (0,1)$ such that $1/\epsilon^2\alpha \geq 2$, there exists an easily constructible binary code of length $n$ that corrects $t = \alpha n$ localized erasures and has redundancy $(1+\epsilon)t$. The encoding complexity of the code is $O(n \cdot \log^2(1/\epsilon\alpha)/\epsilon^4\alpha^2)$ and the decoding complexity is $O(n \cdot \log(1/\epsilon\alpha)/\epsilon^2\alpha)$. Moreover, the coding scheme can be implemented on-line.

*Proof:* We will describe the construction of the code, describe encoding and decoding, compute its redundancy, and estimate its complexity.

We begin with a message of $n - t(1+\epsilon)$ bits. By assumption, the set $U$ is known to the encoder. The set $N = \{1, \cdots, n\}$ of code coordinates is divided into subblocks of a smaller length $d = O(1)$ called segments (we assume without loss of generality that $n$ is a multiple of $d$). The encoding procedure is applied to each of these segments independently. The same holds for the decoding, so the construction introduces only a constant delay and can be implemented on-line.

Thus we need to describe what happens within a given segment $B$ of length $d$, where $d$ is taken equal to

$$d = 2^8 \log(1/\epsilon^2\alpha)/\epsilon^2\alpha.$$

(We remark that $2^8$ is not the minimal possible value of the constant.) The construction relies on two more parameters

$$\rho = \frac{2+\epsilon}{2(1+\epsilon)} \qquad l = \frac{2\log d + 1}{1-\rho}. \tag{4}$$

First, if the number of possible erasures in $B$ exceeds $\rho d$, the entire segment $B$ is filled with zeros and is not used to transmit information. Otherwise, since $B$ is of constant length, we could apply the optimal code of Theorem 3.1 on it. This would solve our problem, were the number of possible erasures in $B$ known to the decoder. To communicate this number, we use a part (subsegment) of segment $B$ with relatively few possible erasures. Such a subsegment of length $l$ exists since $B$ itself has at most $\rho d$ erasures. The final problem is to communicate to the decoder the location of this subsegment.

Before describing the coding scheme in detail, we note one inequality, useful for later analysis, which can be easily verified by using our definitions of $d$ and $\rho$ and the assumption of the theorem

$$\frac{2\log d + 1}{1-\rho} + 1 \leq \frac{\epsilon\alpha}{2} d. \tag{5}$$

*Encoding:* More to the point, we shall partition $B$ into $d/l$ consecutive subsegments $B_1, B_2, \cdots, B_{d/l}$, each of length $l$. Note there is an index, say $k$, such that $|B_k \cap U| \leq \rho l$. As said above, $B_k$ does not carry any part of the message. Let $r = |(B \setminus B_k) \cap U|$ be the number of possible erasures in the remaining part of the segment. We use the code $\mathcal{D}[d-l, r]$ to encode $d-l-r-1$ message bits and transmit them on $B \setminus B_k$.

It remains to specify the use of $B_k$ to accomplish the goals described above. A codeword of $\mathcal{D}[l, \rho l]$, written on this subsegment, is calculated from the codeword of $\mathcal{D}[d-l, r]$ written on $B \setminus B_k$, the number $r$, and the starting bit position of $B_k$ in $B$. Let $x \in \mathbf{Z}_2^{2\log d}$ be a binary vector, the first $\log d$ bits of which carry this position and the last $\log d$ bits represent $r$.

The codeword written on $B_k$ is calculated as follows. First, we apply $\mathcal{D}[l, \rho l]$ decoding to each subsegment $B_i$, $i \neq k$, of $B$. This is possible by the remark after Theorem 3.1. Denote the result of the $i$th decoding by $x_i$. It is clear that each such decoding gives a bit sequence of length $l - \rho l - 1 = 2\log d$. Next, compute

$$y = x + \sum_{i \neq k} x_i \pmod 2. \tag{6}$$

Finally, we encode $y$ with $\mathcal{D}[l, \rho l]$ and write the result on $B_k$.

*Decoding:* The decoding begins upon receipt of the first $d$-bit segment $B$. If after replacing the erased bits by zeros, the decoder observes an all-zero word, the segment is assumed not to carry any information and discarded. By the remark after Theorem 3.1, the only case when this can occur is when $|B \cap U| \geq \rho d$, i.e., this decision coincides with the intentions of the encoder.

Otherwise, we decompose $B$ into $d/l$ subsegments $B_1, B_2, \cdots, B_{d/l}$, each of length $l$. Then we apply the $\mathcal{D}[l, \rho l]$ decoding on each $B_i$. Let $y_i \in \mathbf{Z}_2^{2\log d}$ be the output of the decoding on $B_i$. Next compute $x = \sum_{i=1}^{d/l} y_i \pmod 2$. By (6), the first $\log d$ bits of this vector represent the (binary) number of the starting position of $B_k$ and the last $\log d$ bits contain the number $r$ of erasures in $B \setminus B_k$. So what remains is to isolate the segment $B \setminus B_k$ and apply the $\mathcal{D}[d-l, r]$ decoding on it. This gives the transmitted message.

Let us estimate the overall redundancy of the code. Let $W$ be the collection of segments $B$ with the number of possible erasures $t_B := |B \cap U| \geq \rho d$. The following parts of the $n$-word do not carry information: all the segments in $W$, the subsegments $B_k$ of the segments not in $W$, and the redundant bits of codewords of $\mathcal{D}[d-l, t_B]$ on these segments.

Let $w = |W|$. Clearly, $0 \leq w \leq \alpha n/\rho d$. The total number of segments is $n/d$ and the number of erasures in all the segments $B \notin W$ is at most $t - w\rho d$. Therefore, for the overall redundancy $K$ we get

$$\begin{aligned} K &\leq wd + \sum_{B \notin W} (l + t_B + 1) \\ &\leq wd + \left(\frac{n}{d} - w\right)\left(\frac{2\log d + 1}{1-\rho} + 1\right) + (t - w\rho d) \\ &\leq wd + \left(\frac{n}{d} - w\right)\frac{\epsilon\alpha d}{2} + (t - w\rho d) \end{aligned} \tag{7}$$

where the last inequality follows from (5). We need to show that $K \leq (1+\epsilon)\alpha n$ for $0 \leq w \leq \alpha n/\rho d$. Since (7) is linear in $w$, it suffices to show that the last expression in (7) is at most $(1+\epsilon)\alpha n$ for both $w = 0$ and $w = \alpha n/\rho d$.

*Case 1:* $w = 0$

$$(7) = \frac{n}{d}\frac{\epsilon\alpha d}{2} + \alpha n$$

*Case 2:* $w = \alpha n/\rho d$

$$(7) = \frac{\alpha n}{\rho} + \left(\frac{n}{d} - \frac{\alpha n}{\rho d}\right)\frac{\epsilon\alpha d}{2} = \left(\frac{\alpha}{\rho} + \frac{(\rho-\alpha)\epsilon\alpha}{2\rho}\right)n$$
$$\leq \left(\frac{\alpha}{\rho} + \frac{\epsilon\alpha}{2\rho}\right)n$$
$$= (1+\epsilon)\alpha n$$

where the last step makes use of (4).

To complete the proof it remains to estimate the complexity of encoding and decoding. By Theorem 3.1, it is easy to see that the time needed for encoding each segment $B$ is $O(d^3)$ and for decoding is $O(d^2)$. Therefore, the overall encoding time (on $n/d$ segments) is $O(d^2 n) = O(n \cdot \log^2(1/\epsilon\alpha)/\epsilon^4\alpha^2)$ and the overall decoding time is $O(dn) = O(n \cdot \log(1/\epsilon\alpha)/\epsilon^2\alpha)$. □

### C. Recursive Construction

As we mentioned in Section I, Dumer [7] gave a nearly optimal construction of binary codes of length $n$ correcting $t$ defects with $(1+o(1))t$ redundancy, $O(n \log^2 n)$ encoding time, and $O(n \log n)$ decoding time. This construction can be easily modified to give binary codes correcting localized erasures with the same amount of redundancy and the same encoding/decoding complexity. In this section, we outline an extension of that construction and derive a coding scheme that satisfies the following properties.

*Theorem 3.3:* Given any $\alpha, \epsilon, \delta \in (0, 1)$ such that $(1+\epsilon)\alpha \leq 1-\delta$, there is an $n_0(\alpha, \epsilon, \delta)$ such that for any integer $n \geq n_0$, there exists an easily constructible binary code of length $n$ that corrects $t = \alpha n$ localized erasures and has redundancy $(1+\epsilon)t$. The encoding complexity of the code is $O(n \log^2(1/\alpha\epsilon\delta)/(\alpha\epsilon\delta)^2)$ and the decoding complexity is $O(n \log(1/\alpha\epsilon\delta)/\alpha\epsilon\delta)$.

Here we point out that this construction is recursive and the coding scheme cannot be implemented on-line.

*Proof:* We assume the existence of an easily constructible binary code of any length $n' < n$ correcting any $t' < t$ localized erasures that satisfies the properties required in the theorem. The construction proceeds recursively and relies on this assumption as an induction hypothesis.

As in Theorem 3.2, we present a constructive definition of the code by specifying the encoding procedure. Then we describe the decoding, compute the code redundancy, and estimate the necessary complexities. The general encoding scheme is based on isolating a large (linear-size) segment $R \subset N$ in which the fraction of possible erasures is at most the same as in the entire block. This segment does not carry message bits; instead, we encode on it the number of possible erasures in the remaining part of $N$. Encoding and decoding on $R$ are performed with a code whose existence is guaranteed by the induction hypothesis. The rest of the block length is used to transmit message bits with codes of Theorem 3.1 and the information about the location of $R$. This is done similarly to Theorem 3.2.

Let us introduce notation for the partition described, specify the parameters, and give a formal description of encoding and decoding. Let $|R| = \beta n$, where $\beta = \frac{\epsilon\alpha}{2}$. In the remaining part we isolate an auxiliary segment $I$ of length $l = (\log(1/\beta)+1)/(1-\frac{\alpha}{1-\beta})$, which is used to transmit the location of $R$. Finally, the set $N \setminus (I \cup R)$ is partitioned into subsets $B_1, B_2, \cdots, B_q$ each of length $d$, called blocks. Here

$$d = 4 \log(1/\beta\delta)/\beta\delta = O(\log(1/\alpha\epsilon\delta)/\alpha\epsilon\delta). \quad (8)$$

All the parameters are known both to the encoder and decoder and

*Encoding:* We are given a set of possible erasures $U$ and a message of $n-(1+\epsilon)t$ bits. Let $R \subset N$, $|R| = \beta n$, be a segment such that $|R \cap U| \leq \alpha\beta n$. Then we partition the subset $N \setminus R$ into consecutive segments of length $l$ each and fix one such segment $I$ with $|I \cap U| \leq \frac{\alpha l}{(1-\beta)}$. (This can be done even if $U \subset N \setminus R$.) Finally, we partition the remaining part of $N$ into $q = ((1-\beta)n - l)/d$ consecutive blocks $B_1, B_2, \cdots, B_q$ each of length $d$.

Let $x_i := |B_i \cap U|$. Block $B_i$ is used to transmit $d - x_i - 1$ message bits, protected against erasures with the code $D[d, x_i]$. We also need to transmit to the decoder the $q$ numbers $x_i$ and the starting position $y$ of the auxiliary segment $I$. In total this is $q \log d + \log n$ bits. These bits are encoded with the $[\beta n, \alpha n]$ code, whose existence constitutes the induction hypothesis. The encoding result is written in $R$. Note that

$$q \log d + \log n + (1+\epsilon)\alpha\beta n$$
$$= \left(\frac{(1-\beta)\log d}{d} + (1+\epsilon)\alpha\beta\right)n + \log n - \frac{l \log d}{d}$$
$$\overset{(a)}{\leq} ((1-\beta)\delta\beta + (1-\delta)\beta)n + \log n - \frac{l \log d}{d}$$
$$\overset{(b)}{\leq} \beta n$$

where (a) follows by (8) and the assumption for the theorem and (b) holds for $n$ sufficiently large. Thus this step of the encoding procedure is well-defined.

The only thing the encoder still needs to transmit to the decoder is the starting position of the segment $R$. Let us assume without loss of generality that both $n$ and $\beta n$ are multiples of $l$. We partition $N$ into segments $N_1, N_2, \cdots, N_{n/l}$ each of length $l$. Then by the construction, we have that $I = N_k$ for some $1 \leq k \leq n/l$. For $i \neq k$, let $x_i \in \boldsymbol{Z}_2^{\log(1/\beta)}$ be the result of the $\mathcal{D}[l, \alpha l/(1-\beta)]$ decoding on segment $N_i$. Since $R$ is one of the $1/\beta$ consecutive segments of $N$, to locate it we need $\log(1/\beta)$ bits. Let $x \in \boldsymbol{Z}_2^{\log(1/\beta)}$ be the binary representation of the segment number of $R$. Let $y = x + \sum_{i \neq k} x_i \pmod 2$ and let $I$ carry the codeword obtained by $\mathcal{D}[l, \alpha l/(1-\beta)]$ encoding of $y$.

*Decoding:* The decoding begins upon receipt of $n$ bits of the codeword. First we replace all the erased bits by 0. Then we decompose $N$ into $n/l$ segments $N_1, N_2, \cdots, N_{n/l}$ each of length $l$ and apply the $\mathcal{D}[l, \alpha l/(1-\beta)]$ decoding on each segment $N_i$. Each decoding yields a $\log(1/\beta)$-bit vector $y_i$. Let $x = \sum_{i=1}^{n/l} y_i \pmod 2$. By the above, this is the binary number of the segment $R$ in the partition $N$ into $1/\beta$ consecutive segments of length $\beta n$. Upon having located $R$, we apply the decoding recursively on $R$ to obtain the number $x_i$ of possible erasures in $B_i$ and the starting bit position $y$ of segment $I$. Now we can recover all the segments $B_i$, $1 \leq i \leq q$, $i \neq k$, which carry the message. Decoding each of them with the code $\mathcal{D}[d, x_i]$, we get the transmitted message bits. This completes the decoding.

Let us show that the amount of redundancy satisfies the requirement of the theorem. The redundancy of the code consists of segments $R$ and $I$ and the redundancy needed for each $\mathcal{D}[d, x_i]$ encoding on $B_i$. Therefore, the total is at most

$$\beta n + l + \left(\alpha n + \frac{n}{d}\right) = \left(\beta + \frac{\delta\beta}{4\log(1/\delta\beta)}\right)n + \alpha n + l$$
$$\overset{(c)}{\leq} \left(1 + \frac{\delta}{4}\right)\beta n + \alpha n + l$$
$$\overset{(d)}{\leq} (1+\epsilon)\alpha n.$$

Here in (c) we used the obvious inequality $\alpha\delta \leq 1$ which implies

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.