Linear Time Erasure Codes With Nearly Optimal Recovery (Extended Abstract)

Noga Alon [*] Jeff Edmonds [†] Michael Luby [‡]

Abstract

An (n, c, ℓ, r)-erasure code consists of an encoding algorithm and a decoding algorithm with the following properties. The encoding algorithm produces a set of ℓ-bit packets of total length cn from an n-bit message. The decoding algorithm is able to recover the message from any set of packets whose total length is r, i.e., from any set of r/ℓ packets. We describe erasure codes where both the encoding and decoding algorithms run in linear time and where r is only slightly larger than n.

1 Introduction

Most existing and proposed networks are packet based, where a packet is fixed length indivisible unit of information that either arrives intact upon transmission or is completely lost. This model accurately reflects properties of Internet and ATM-based networks, where local error correcting codes can be used (and often are used) on individual packets to protect against possible errors as the packet traverses the network. However, the timely arrival of individual packets sent long distances over a variety of heterogeneous networks is a global property that seems to be harder to control on a local basis. Thus, it makes sense to protect real-time traffic sent through such networks against losses by adding a moderate level of redundancy using erasure codes.

Algorithms based on this approach have been developed for applications such as multicasting real-time high-volume video information over lossy packet based networks [3,2,8] and other high volume real-time applications [11]. The two most important properties of erasure codes in these applications are the running times of the encoding and decoding algorithms and the amount of encoding sufficient to recover the message. An erasure code where any portion of the encoding equal to the length of the message is sufficient to recover the message is called a *maximal distance separable* (MDS) code in the literature. An ideal erasure code would be a linear time MDS code, but so far no such code is known.

Standard Reed-Solomon codes can be used to implement quadratic time MDS codes. These methods have been customized to run in real-time for medium quality video transmission on existing workstations [3,2], i.e., at the rate of a few megabits per second, but high quality video sent at the rate of hundreds of megabits per second will require either better algorithms or custom designed hardware. Theoretically more efficient (but not linear time) MDS codes can be constructed based on evaluating and interpolating polynomials over specially chosen finite fields using Discrete Fourier Transform, but these methods are not competitive in practice with the simpler quadratic methods except for extremely large messages. Thus, the design of highly efficient algorithms for implementing erasure codes is interesting theoretically and important for practical applications.

All of our schemes have the property that the code can be constructed for any message length n and redundancy factor $c > 1$. We call such a code an (n, c)-code. In applications, c is relatively small (it typically varies between 1 and 2, but can be as large as 5). Thus, when stating running times, we ignore the dependence on c.

A natural relaxation of an MDS code is to allow slightly more of the encoding than the optimal amount in order to recover the message: We say a (n, c)-code is $(1+\epsilon)$-MDS if the message can be recovered from any $(1+\epsilon)n$ of the encoding. For example, if $\epsilon = .05$ then only 5% more of the encoding than the optimal amount is sufficient to recover the message. A further relaxation of an MDS code is to allow both the encoding and decoding algorithms to be probabilistic (using the same set of random bits for both encoding and decoding): We say a (n, c)-code is probabilistic $(1+\epsilon)$-MDS if the message can be recovered from any $(1+\epsilon)n$ of the encoding with high probability. For probabilistic codes, the network is assumed to drop packets independent of their contents.

Of secondary importance, but still important, is the length ℓ of the packets. Ideally, the length of the packets should be as short as possible, e.g., $\ell = 1$, because an erasure code using longer packets can always be

[*]Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel and Institute for Advanced Study, Princeton, NJ 08540, USA. Email: noga@math.tau.ac.il. Research supported in part by the Fund for Basic Research administered by the Israel Academy of Sciences and by a USA-Israeli BSF grant.

[†]Computer Science Department, York University, York, Canada. Email: jeff@cs.yorku.ca. Research done while at the International Computer Science Institute supported by an NSF postdoctoral grant and an NSERC fellowship.

[‡]International Computer Science Institute, Berkeley, California, 94704, and Computer Science Division, UC Berkeley. Berkeley, California, 94720. Email: luby@icsi.berkeley.edu. Research supported in part by National Science Foundation operating grant CCR-9304722 and NCR-9416101, United States-Israel Binational Science Foundation grant No. 92-00226, and ESPRIT BR Grant EC-US 030.

constructed by concatenating several packets from an erasure code using shorter packets, but the reverse is not necessarily true. Internet packets for sending video are typically moderate sized, e.g., 1000 bytes, but ATM cells are rather short, i.e., 48 bytes, and here the payload size is more of a constraint. We try to minimize the value of ℓ as much as possible, but in general ignore this parameter when stating results.

We assume that each packet contains a unique index. ¿From this index, the portion of the encoding carried by each received packet can be determined. We do not count the space for the index as part of the packet size. In practice, the space for this index is small compared to the size of the payload of the packet, and packet based protocols typically include a unique index for each packet within the packet in any case.

This paper describes two new erasure code schemes, the first deterministic and the second probabilistic. The first scheme has the property that, on inputs n, c, and ϵ, the run time of the (n, c)-code is $\mathcal{O}(n/\epsilon^4)$, it is $(1+\epsilon)$-MDS, and the packet size is $\mathcal{O}(1/\epsilon^4 \log(1/\epsilon))$. Note that for constant ϵ this scheme runs in linear time. Although this is an interesting theoretical result, it is not clear if it can be made practical for values of ϵ that are reasonable, e.g., $\epsilon = .10$, because the $(1/\epsilon^4)$ factor is rather large both in terms of the running time and the packet size.

The second scheme has the property that, on inputs n, c, and ϵ, the run time of the (n, c)-code is $\mathcal{O}(n \log(1/\epsilon)/\epsilon))$, it is probabilistic $(1+\epsilon)$-MDS, and the packet size can be as small as $\mathcal{O}(\log(1/\epsilon))$. Partial implementations of variants of this scheme show it has promise of being practical.

Both of the schemes we describe are what is called *systematic* in the literature, which means that the message itself is part of the encoding. This property is good especially in the case when only a small number of the packets are lost, because the time to decode the message from the encoding is proportional only to the amount of the message that is missing.

Our deterministic scheme is based on the properties of *expanders* which are explicit graphs with pseudo-random properties. The relevance of these graphs to error correcting codes has been observed in [4], and indeed we apply some of the ideas of that paper. Our probabilistic scheme is based on grouping the message into a hierarchical set of unequal length blocks and then placing in each redundant packet the value of a random linear equation evaluated on a randomly chosen block of the message.

Erasure codes are related to error correcting codes, and are typically easier to design. For example, an error correcting code with encoding length cn that can correct up to bn bit flips can be used as an erasure code that can recover the message from any $(c-b)n$ of the encoding: For packets not received, set the missing bits to zero and then use the error correcting code to recover the message. This can be improved to $(c-2b)n$ by setting the missing bits randomly, noting that on average half of them will be correctly set.

The recent breakthrough result of Spielman [13] on error correcting codes is directly relevant to both our schemes. Spielman applies the techniques in [12] and [7], and constructs linear time error correcting codes with linear rate and linear minimum distance. This error correcting code stretches an n-bit message to a cn-bit message and can recover the message when up to bn of the encoding bits are flipped. Here, $b << 1$ and $c \approx 4$ are absolute constants. A direct application of [13] to the design of an erasure code yields a linear time code that at best is $(4-2b)$-MDS. Thus, [13] cannot be used directly to yield an $(1+\epsilon)$-MDS code for an arbitrary value of ϵ. Nevertheless, [13] is a crucial ingredient in both of our constructions.

2 Recovering All from Almost All

We assume that $\epsilon < 1$. Let $\gamma = \epsilon/3$, $\gamma' = \gamma/4$, $\beta = \gamma'^2/8$, and let $q = 4\log(1/\epsilon) + \log(c) + 16$. Throughout, our basic unit of length is a *letter*, which is a bit string of length q. We assume operations on letters, such as the XOR or AND of two letters, can be performed in constant time. The value of q has been chosen large enough so that all of the MDS codes we use in our constructions can be implemented over the finite field GF$[2^q]$.

Let $M_1, \ldots, M_n$ be the message consisting of n letters. The first step of both schemes constructs an encoding $M_1, \ldots, M_n, S_1, \ldots, S_{\gamma n}$ with the property that the message can be recovered from any fraction $1-\beta$ of this encoding.

The first step proceeds in two stages. The first stage uses expander graphs to construct $S_1, \ldots, S_{\gamma' n}$ from the message. The property of the first stage is that the entire message can be recovered from any $n - \beta n$ portion of the message given all of $S_1, \ldots, S_{\gamma' n}$. The second stage directly uses the constructions of Spielman [13] to stretch $S_1, \ldots, S_{\gamma' n}$ to $S_1, \ldots, S_{\gamma n}$. This stage has the property that all of $S_1, \ldots, S_{\gamma' n}$ can be recovered from any $\gamma n - \beta n$ portion of $S_1, \ldots, S_{\gamma n}$. Thus, the overall property of the first two stages is that the message can be recovered when up to a βn portion of $M_1, \ldots, M_n, S_1, \ldots, S_{\gamma n}$ is missing.

2.1 Stage 1: Restricted erasures

The main result of this subsection is the following.

Lemma 1 *There is a scheme for generating, for any given message $M_1, \ldots, M_n$ of n letters, a sequence of $\gamma' n$ additional letters $S_1, \ldots, S_{\gamma' n}$ with the following properties.*

(i) *The encoding time is $\mathcal{O}(n/\epsilon)$.*

(ii) *If $S_1, \ldots, S_{\gamma' n}$ are known, and at most a fraction β of $M_1, \ldots, M_n$ are missing, then all of $M_1, \ldots, M_n$ can be recovered in time $\mathcal{O}(n)$.*

The construction used in the proof of the above is similar to the one in [12], and is based on properties of expanders. Let us call an infinite increasing sequence of integers *dense* if the ratio between consecutive elements of the sequence tends to 1. The known constructions of expanders supply, for every admissible degree of regularity, infinite families of graphs on sets of nodes whose cardinalities form a dense sequence.

To simplify the presentation we assume here that there are sufficiently many expanders in these families whose number of nodes is divisible by any desired constant. It is not difficult to show that this assumption can be omitted.

Definition (Expanders): A graph is called a (d, λ)-*expander* if it is d-regular and the absolute value of each of its nontrivial eigenvalues is at most λ.

By [9], [10] the sequence of integers m for which there is a $(d, 2\sqrt{d-1})$-expander on m nodes is a dense sequence. We need the following from [5].

Proposition 1 *[5] The number of edges induced by any set of x nodes in a (d, λ)-graph on m nodes does not exceed*

$$\frac{1}{2}x(d\frac{x}{m} + \lambda(1 - \frac{x}{m})).$$

Proof of Lemma 1: Fix an integer d, where $\frac{64}{\gamma'^2} < d \leq \frac{128}{\gamma'^2}$, and let $\lambda = 2\sqrt{d-1}$. Let $G = (V, E)$ be a (d, λ)-expander on $m = 2n/d$ nodes. Given a sequence $\{M_e : e \in E\}$ of n letters we define the corresponding sequence $S_1, \ldots, S_{\gamma' n}$ by assigning each node v of G a set of $\gamma' d/2$ letters $S_{v,1}, \ldots, S_{v,\gamma' d/2}$ as described below. Note that the total number of letters $S_{v,j}$ is $m\gamma' d/2 = \gamma' n$, as needed. Let v be a node of G and let $e_1, \ldots, e_d$ be the edges incident with it. Use a quadratic time MDS code to map the message $M_{e_1}, \ldots, M_{e_d}$ to an encoding $M_{e_1}, \ldots, M_{e_d}, S_{v,1}, \ldots, S_{v,\gamma' d/2}$. Such a code can be implemented so that the encoding time is proportional to $d \cdot \gamma' d/2 = \mathcal{O}(d/\epsilon)$ and the decoding time is proportional to d^2, plus an additive $\mathcal{O}(d)$ for each missing message letter. Note that the letter length q is sufficient to implement such a code.

We claim that this scheme satisfies the two properties required in the proposition. The validity of (i) is clear, as the total encoding time is $\mathcal{O}(md/\epsilon) = \mathcal{O}(n/\epsilon)$.

Since we assume we are missing at most βn message letters, and since the time for recover of each missing message letter is $\mathcal{O}(d^2)$, and since $\beta d^2 = \mathcal{O}(1)$, it follows that the decoding time is at most $\mathcal{O}(n)$.

We now prove that the entire message can be recovered if we are given all the letters associated with the nodes and at most $\beta n = \gamma'^2 n/8$ of the original message letters are missing. The decoding algorithm works as follows. If there is some node v in the graph where at most $\gamma' d/2$ of the message letters associated with the edges incident with v are missing, then, because $S_{v,1}, \ldots, S_{v,\gamma' d/2}$ are also known, we know a total of at least d letters of the MDS code associated with v, and thus from the properties of the MDS code we can recover all the missing message letters associated with edges incident to v. Repeating this process as long as there are such nodes v, we either recover the entire message, or we are left with a nonempty set of edges corresponding to the missing message letters that form a subgraph of minimum degree greater than $\gamma' d/2$ in G.

We now show that Proposition 1 implies that if the subgraph is non-empty then it must contain more than βn edges, and from the assumption that we started with at most βn such edges it will follow that the subgraph must be empty, i.e., the entire message is recovered. Let x denote the number of nodes incident with edges of this subgraph. Then, since each such node has degree at least $\gamma' d/2$ in the subgraph, the total number of edges of the subgraph exceeds $x\gamma' d/4$. Thus, by Proposition 1,

$$x\gamma' d/4 \leq \frac{1}{2}x(dx/m + \lambda(1 - x/m)).$$

Therefore,

$$\frac{\gamma' d}{2} \leq \frac{dx}{m} + \lambda(1 - \frac{x}{m}) \leq \frac{dx}{m} + \lambda.$$

Since $\lambda = 2\sqrt{d-1} < 2\sqrt{d}$, and $d > 64/\gamma'^2$, the inequality $\gamma' d/2 - \lambda \geq \gamma' d/4$ holds and hence

$$dx/m \geq \gamma' d/2 - \lambda \geq \gamma' d/4,$$

implying that the number of edges corresponding to missing letters exceeds

$$x\gamma' d/4 \geq \frac{\gamma'^2 md}{16} = \frac{\gamma'^2}{8}n = \beta n,$$

contradicting the assumption. This completes the proof. ■

2.2 Stage 2: Spielman-like Construction

We need the following result, which is an easy consequence of the main result of Spielman in [13].

Proposition 2 *[13] There is an absolute positive constant b so that for all m there is an explicit construction that maps messages of m letters into $4m$ letters so that:*

(i) *The encoding time is $\mathcal{O}(m)$.*

(ii) *If at most bm letters are missing then the original m letters can be recovered in time $\mathcal{O}(m)$.*

We note that since we are interested here only in erasure codes, whereas the construction of Spielman supplies error correcting ones, it is possible to improve the constant b that follows from his construction considerably, but since we are not optimizing the constants here we do not include the details. For simplicity hereafter, we assume $b \geq \gamma'/8$, which implies that $b\gamma' \geq \beta$.

The second stage of the construction uses the construction of Proposition 2 to stretch $S_1, \ldots, S_{\gamma' n}$ to $S_1, \ldots, S_{\gamma n}$. The next lemma follows directly from Lemma 1 and Proposition 2.

Lemma 2

(i) *The encoding $M_1, \ldots, M_n, S_1, \ldots, S_{\gamma n}$ can be computed in $\mathcal{O}(n/\epsilon)$ time from the message $M_1, \ldots, M_n$.*

(ii) *The message can be decoded in time $\mathcal{O}(n)$ when at most βn letters of the encoding are missing.*

3 The Deterministic Scheme

Let $c' = c/(1+\gamma)$, $N = (1+\gamma)n$, and $\ell = 8/(\gamma^2\beta)$. The final goal is to stretch the encoding produced by the first step by a factor of c'. The second step partitions $M_1, \ldots, M_n, S_1, \ldots, S_{\gamma n}$ produced from the first step into *blocks* $B_1, \ldots, B_{N/\ell}$ of ℓ letters each and then uses a standard MDS erasure codes to produce, for each $i \in \{1, \ldots, N/\ell\}$, an encoding E_i based on B_i consisting of $c'\ell$ letters. Note that the letter length q is sufficient to implement such a code. The properties of the second step are the following.

Lemma 3

(i) *If, for at least a fraction $1-\beta$ of the N/ℓ encodings $E_1, \ldots, E_{N/\ell}$, at least ℓ letters of the encoding are recovered, then the entire message $M_1, \ldots, M_n$ can be recovered.*

(ii) *Both the encoding and decoding times are $\mathcal{O}(n\ell)$.*

Proof of Lemma 3: By properties of MDS codes, for each i where at least ℓ letters of E_i are recovered, the corresponding block of B_i can be completely recovered. From Lemma 2 and the conditions of the lemma, it follows that the message can be completely recovered. The time for both encoding and decoding using a quadratic time MDS code is $(N/\ell) \cdot \ell^2 = N\ell$. ∎

The third step of the scheme is to use a $c'\ell$-regular expander graph with N/ℓ nodes to deterministically simulate a "random mapping" of the letters of the encodings $E_1, \ldots, E_{N/\ell}$ into N/ℓ packets $P_1, \ldots, P_{N/\ell}$ containing $c'\ell$ letters each.

Lemma 4 *There is a scheme for mapping the letters of $E_1, \ldots, E_{N/\ell}$ into packets $P_1, \ldots, P_{N/\ell}$ containing $c'\ell$ letters each such that:*

(i) *The time for both the mapping and the inverse mapping is $\mathcal{O}(n)$.*

(ii) *Every set $\mathcal{I} \subseteq \{1, \ldots, N/\ell\}$ with $|\mathcal{I}| \geq \frac{(1+\epsilon)n}{c'\ell}$ has the following property: For at least a fraction $1-\beta$ of $i \in \{1, \ldots, N/\ell\}$, at least ℓ letters of E_i are contained in the set of packets indexed by $\mathcal{I}$.*

Proof of Lemma 4: Let $\lambda = 2\sqrt{c'\ell - 1}$. Let $G = (V, A)$ be a $(c'\ell, \lambda)$-expander with $V = \{1, \ldots, N/\ell\}$. The mapping of the letters of $E_1, \ldots, E_{N/\ell}$ into packets $P_1, \ldots, P_{N/\ell}$ is defined as follows: For each $i \in V$, let $(i, w_1), \ldots, (i, w_{c'\ell})$ be the edges incident to i in G. Then, the j^{th} letter of the encoding E_i is placed into packet P_{w_j}.

Let $\mathcal{I}$ be any subset of V with $|\mathcal{I}| = \frac{(1+\epsilon)n}{c'\ell}$. For each $i \in V$, let d_i denote the number of letters of E_i that are in the packets indexed by $\mathcal{I}$. By a lemma in [5] (see also [6], page 122),

$$\sum_{i \in V}(d_i - |\mathcal{I}|c'\ell^2/N)^2 \leq \lambda^2|\mathcal{I}|(1-|\mathcal{I}|\ell/N) \leq 4c'\ell|\mathcal{I}| \leq 8n. \tag{1}$$

Note that

$$\frac{|\mathcal{I}|c'\ell^2}{N} = \frac{(1+\epsilon)n\ell}{N} = \frac{(1+\epsilon)\ell}{1+\gamma} \geq (1+\gamma)\ell. \tag{2}$$

Let M be the set of $i \in V$ for which the packets indexed by $\mathcal{I}$ contain less than ℓ letter of E_i. ¿From Equation (2) it follows that, for each $i \in M$,

$$(d_i - |\mathcal{I}|c'\ell^2/N)^2 \geq (\gamma\ell)^2,$$

and thus the left-hand side of Inequality (1) is at least $|M| \cdot (\gamma\ell)^2$. This and Inequality (1) implies that $|M| \leq \frac{8n}{(\gamma\ell)^2}$. Recalling that $\ell = \frac{8}{\gamma^2\beta}$, this implies that $|M| \leq \beta n/\ell \leq \beta N/\ell$ as desired. ∎

We now state and prove the main theorem.

Theorem 1 *There is a scheme that, on input n, c and ϵ, has the following properties:*

(i) *A message of n letters is encoded into packets containing a total of cn letters, where each packet contains $\mathcal{O}(1/\epsilon^4)$ letters.*

(ii) *The message can be decoded from any set of packets containing in total at least $(1+\epsilon)n$ letters.*

(iii) *The run time for both the encode and decode algorithms is $\mathcal{O}(n/\epsilon^4)$.*

Proof of Theorem 1: The encoding consists of applying the constructions described in steps 1, 2, and 3, in sequence. The decoding guarantee and the run time follow from combining Lemma 4 with Lemma 3. ∎

4 The Probabilistic Scheme

In this section, we relax the requirements on the erasure code to a probabilistic guarantee and thereby improve the running time from $\mathcal{O}(n/\epsilon^4)$ to $\mathcal{O}(n\log(1/\epsilon)/\epsilon)$ and the packet size from $\mathcal{O}(1/\epsilon^4\log(1/\epsilon))$ to $\mathcal{O}(\log^2(1/\epsilon))$. We will first present a simple way of using randomness in the above deterministic scheme that allows the block size ℓ to be $\mathcal{O}(\log(1/\epsilon)/\epsilon^2)$ instead of $\mathcal{O}(1/\epsilon^4\log(1/\epsilon))$. Automatically, this deceases running time $\mathcal{O}(n\ell)$ to $\mathcal{O}(n\log(1/\epsilon)/\epsilon^2)$.

Starting from Lemma 3, we are given the N/ℓ encodings $E_1, \ldots, E_{N/\ell}$ consisting of $c'\ell$ letters each. Recall, the deterministic scheme used an expander graph to simulate "randomly mapping" these letters. Instead, the probabilistic scheme simply randomly permutes the $c'N = cn$ letters and puts one into each of the packets (as opposed to $c'\ell$ letters per packet). When one receives any $(1+\epsilon')N = (1+\epsilon)n$ of the packets, one receives a random subset of $(1+\epsilon')N$ of the letters. Note the expected number received about a particular block is $(1+\epsilon')\ell$. What remains is to prove that with high probability for at least a fraction $1-\beta$ of the N/ℓ encodings $E_1, \ldots, E_{N/\ell}$, at least ℓ letters are received. Lemma 3 then states that the entire message can be recovered. (For the rest of this section the ' will be dropped on the ϵ and the c.)

A key parameter to this block based encoding scheme is the block size ℓ. If it is too big, then the running time is too large. If it is too small, then the probability of failure is too large. For example, consider the extreme example when the blocks are of size one. In this case, each packet would contain the j^{th} message letter with probability $1/N$. Receiving $(1-\beta)N$ of the letters, becomes the classical coupon collector or occupancy problem. It is well known that $N\ln(1/\beta) >> (1+\epsilon)N$ packets would need to be received, before one could expect to receive $(1-\beta)N$ distinct message letters. The optimal block size is $\ell = \Theta(\ln(\frac{1}{\beta})/\epsilon^2)$.

Lemma 5 *When the blocks have size $\ell = \Omega(\ln(\frac{1}{\beta})/\epsilon^2)$, the probability of not being able to recover at least $(1-\beta)N$ of the message is at most* $\mathrm{e}^{-\Omega(\beta\epsilon^2 N)}$.

Proof of Lemma 5: A complication in the proof is that there is dependency between the number of letters of the encoding received about the different blocks. To simplify things, we will first consider a different distribution, in which the number received about each block is chosen independently according to the Poisson distribution with mean $(1+\epsilon)\ell$. Standard bounds on the Poisson distribution give the probability of not receiving at least ℓ letters about a particular block when you expect $(1+\epsilon)\ell$ is at most $\mathrm{e}^{-\Omega(\epsilon^2\ell)}$. Then the recovery of each block is an independent Bernoulli trial. Standard Chernoff bounds state that the probability of failing to recover at least a $1-\beta$ fraction of the N/ℓ blocks is at most $\mathrm{e}^{-\Omega(\beta(N/\ell)[\epsilon^2\ell-\ln(1/\beta)])} = \mathrm{e}^{-\Omega(\beta\epsilon^2 N)}$. (For small message sizes $N = \mathcal{O}(\ln(\frac{1}{\delta})/\beta\epsilon^2)$, a block size of $\ell = \mathcal{O}\left(\ln(\frac{m}{\delta\ell})/\epsilon^2\right)$ gives a failure probability of at most δ.)

What remains is to adjust for the change in distribution. The key observation is that the only difference between the distributions is that in the original distribution, the total number of letters received is definitely $(1+\epsilon)N$, while in the new one the number received is a random variable. Let K denote this number received. It happens that if one takes the new distribution with the added constraint that $K = (1+\epsilon)N$, then one (more or less) gets the original distribution, i.e. $\mathbf{Pr}_{old}[E] \le \mathbf{Pr}_{new}[E \mid K = (1+\epsilon)N]$, where E denotes the event that not enough of the message is recovered. Decreasing the number of letters of the encoding received only increases the probability of E. Therefore, $\mathbf{Pr}_{old}[E] \le \mathbf{Pr}_{new}[E \mid K \le (1+\epsilon)N]$. The variable K has the Poisson distribution with mean $(1+\epsilon)N$. Therefore, $\mathbf{Pr}_{new}[K \le (1+\epsilon)N] \ge \frac{1}{\mathrm{e}}$. We can conclude that $\mathbf{Pr}_{old}[E]\cdot\frac{1}{\mathrm{e}}. \le \mathbf{Pr}_{new}[E \mid K \le (1+\epsilon)N] \cdot\mathbf{Pr}_{new}[K \le (1+\epsilon)N] \le \mathbf{Pr}_{new}[E]$. ■

Both the deterministic and the above probabilistic scheme use a standard deterministic MDS erasure code to expand each of the N/b blocks B_j from ℓ letters to $c'\ell$ letters. This takes time $\mathcal{O}(\ell^2)$ for a total of $\mathcal{O}(N\ell)$ time. Hence, the running time of the probabilistic scheme can be improved to $\mathcal{O}(N\log(1/\epsilon)/\epsilon)$ simply by using instead a probabilistic erasure code with the following properties.

Lemma 6 *There is a deterministic encoding scheme mapping a block B of ℓ letters into an encoding E with $c\ell$ letters and the following properties.*

(i) *If a random subset of the letters of the encoding is received, where the number received is Poisson with mean $(1+\epsilon)\ell$, then the probability of not recovering all ℓ letters of the message is at most* $\mathrm{e}^{-\Omega(\epsilon^2\ell)}$.

(ii) *The encoding and the decoding times are $\mathcal{O}(\epsilon\ell^2)$.*

(iii) *It is systematic, meaning that the message itself is part of the encoding.*

The encoding scheme is as follows. As required, the first ℓ letters of the encoding consist of the message itself. The message then is broken into a hierarchy of blocks. The message itself is considered to be a block of size ℓ. (Recall in Lemmas 3 and 5, the N letter message is broken into blocks of size ℓ.) This block is then broken into two sub-blocks, which are further broken into two even smaller sub-blocks, and so on. (Recall that smaller blocks lead to a faster computation.) Let $f = \log(1/\epsilon^2) - \mathcal{O}(1)$ be the number of different block sizes and for $s \in \{0,\ldots,f\}$, let $B_{\langle s,i\rangle} \subseteq \{1,\ldots,\ell\}$ be the set of letters of the message in the i^{th} block of size $\ell_s = \ell/2^s$. Let $r_0 = 0$; for $s \in \{1,\ldots,f\}$, let $r_s = c\epsilon 2^{s/2}\ell$; and let $r_{f+1} = (c-1)\ell$. The encoding will consist of $(r_{s+1}-r_s)/2^s$ letters "about" the block $B_{\langle s,i\rangle}$, for $s \in \{0,\ldots,f\}$ and $i \in \{1,\ldots,2^s\}$. This gives $r_{s+1}-r_s$ letters about blocks of size ℓ_s, $(c-1)\ell$ letters about some block, and $c\ell$ letters in total.

The scheme is also specified by a fixed $((r_{s+1}-r_s)/2^s \times \ell_s)$ boolean matrix V_s for reach $s \in \{0,\ldots,f\}$. (It is sufficient to choose these matrices randomly and then to fix them.) Of the $(r_{s+1}-r_s)/2^s$ letters about the block $B_{\langle s,i\rangle}$, the j^{th} letter will be the linear combination (over GF[2]) of the letters in $B_{\langle s,i\rangle}$ specified by the j^{th} row of the matrix V_s. (It is sufficient for all the blocks of the same size to use the same matrix). The ℓ letters of the message can be recovered from the received letters of the encoding if the equations defining the received letters have full rank.

Proof of Lemma 6(i): To simplify the analysis, we will initially not use fixed matrices V_s. Instead, we will independently for each letter of the encoding randomly choose a subset of the letters from the appropriate block. This is done by including each letter of the block with probability $\frac{1}{2}$. The letter of the encoding is a linear combination of the chosen letters.

We will say that a letter of the encoding contributes to a block $B_{\langle s,i\rangle}$ if it is about the block, about one of its sub-blocks, or is itself one of the letters of the block. Denote by $c_{\langle s,i\rangle}$ the number of letters received that are about the block $B_{\langle s,i\rangle}$. If $c_{\langle s,i\rangle}$ exceeds the size of the block ℓ_s, then the message block is over-determined and the excess encoding letters are necessarily useless. Consider each block $B_{\langle s,i\rangle}$ of size $\ell_s < \ell$ starting with the smallest blocks. The first step is to compute the expected value of $c_{\langle s,i\rangle}$. Being a block of size ℓ_s, the expected number of letters received that are themselves one of the letters of the block is $\frac{1+\epsilon}{c}\ell_s$. The expected number received about a block $B_{\langle s',i'\rangle}$ of size $\ell_{s'}$ is $\frac{1+\epsilon}{c}(r_{s+1}-r_s)/2^s$. The number of sub-blocks of $B_{\langle s,i\rangle}$ of size $\ell_{s'}$ is $\frac{\ell_s}{\ell_{s'}} = 2^{s'-s}$. Hence, $\mathbf{E}(c_{\langle s,i\rangle}) = \frac{1+\epsilon}{c}\ell_s + \sum_{s'\in[s..f]} \frac{\ell_s}{\ell_{s'}}[\frac{1+\epsilon}{c}(r_{s'+1}-r_{s'})/2^{s'}] = (1-\Omega(\epsilon 2^{s/2}))\ell_s$. Chernoff bounds give that $\mathbf{Pr}\left[c_{\langle s,i\rangle} > (1-\epsilon^2 2^s)\ell_s\right] \le \mathrm{e}^{-\Omega(\epsilon^2 2^s \ell_s)} = \mathrm{e}^{-\Omega(\epsilon^2\ell)}$. Now assume, that we receive fewer than $(1-\epsilon^2 2^s)\ell_s$ letters contributing to the block $B_{\langle s,i\rangle}$.

The next step is to bound the probability that the equations defining these $(1-\epsilon^2 2^s)\ell_s$ letters are linearly independent. Consider the k^{th} letter contributing to $B_{\langle s,i\rangle}$. If it is about a sub-block of $B_{\langle s,i\rangle}$, then we considered the possibility of the equation defining this letter being linearly dependent on the previous equations when we considered that sub-block. On the other hand, if the k^{th} letter is about $B_{\langle s,i\rangle}$, then the equation is randomly chosen from a space of dimension ℓ_s. The previous $k-1$ equations span a sub-space of dimension at most $k-1$. Hence, the probability that the k^{th} equation is within this sub-space is at most $2^{-(\ell_s-k+1)}$. The probability that the $(1-\epsilon^2 2^s)\ell_s$ equations are dependent is at most $\sum_{k\in\{1,\dots,(1-\epsilon^2 2^s)\ell_s\}} 2^{-(\ell_s-k+1)} \le 2^{-\Omega(\epsilon^2 2^s \ell_s)} = \mathrm{e}^{-\Omega(\epsilon^2\ell)}$.

Even after multiplying this probability by the number of blocks $B_{\langle s,i\rangle}$ of size $\ell_s < \ell$, the probability is still at most $\mathrm{e}^{-\Omega(\epsilon^2\ell)}$. Hence we can conclude that with high probability all the letters that are about blocks of size smaller than ℓ are linearly independent.

What remains is the consider the letters of the encoding that are about the block of size ℓ. By the statement of the lemma, the number of letters of the encoding received is Poisson with mean $(1+\epsilon)\ell$. Therefore, the probability that at least $(1+2\epsilon^2)\ell$ letters are received is at least $1-\mathrm{e}^{-\Omega(\epsilon^2\ell)}$. With the same argument just given, the first $(1-\epsilon^2)\ell$ of these are linearly dependent with probability at least $1-\mathrm{e}^{-\Omega(\epsilon^2\ell)}$. Now consider one of the remaining $3\epsilon^2\ell$ equations. If the equations before it do not have full rank, then the probability that it increases the rank is at least $\frac{1}{2}$. Hence, choosing these $3\epsilon^2\ell$ equations can be thought of as $3\epsilon^2\ell$ Bernoulli trials. The matrix has full rank if at least $\epsilon^2\ell$ of the trials succeed. The expected number of successes is $1.5\epsilon^2\ell$. The probability of getting fewer than $\epsilon^2\ell$ is at most $\mathrm{e}^{-\Omega(\epsilon^2\ell)}$.

The remaining step is to prove that it is sufficient to use a fixed matrix V_s for each size of block. Assume by way of induction that the probability of success is the same as proven above even if a fixed matrix $V_{s'}$ is used for each size of block $\ell_{s'}$ for $s' < s$, while the equations for the letters about larger blocks are still chosen independently at random. Now change the scheme so that the equations for the letters about blocks of size ℓ_s are chosen as follows. First choose the $((r_{s+1}-r_s)/2^s \times \ell_s)$ boolean matrix V_s by choosing each entry independently from $\{0,1\}$ with probability $\frac{1}{2}$. Then for each block of this size independently do the following. For each of the letters received about the block choose independently without replacement a row from the matrix V_s. For a particular block of size ℓ_s, the probability distribution has not changed at all. The only change is that using the same V_s for each block of size ℓ_s adds some dependence between the events for these blocks. This is not a problem for the following three reasons. First, the equations about different blocks are on different variables and hence will not be linearly dependent. Second, much of the randomness in choosing the equations comes from choosing $(1+\epsilon)/c$ of the rows of V_s. Finally, recall that in the above proof, the probability of failure for one block was multiplied by the number of blocks. Hence, the proof does not assume independence between these events. Therefore, the overall probability of failure remains unchanged. We can conclude that if the overall probability of failure is $\mathrm{e}^{-\Omega(\epsilon^2\ell)}$ when V_f is chosen randomly, then there exists a fixed matrix V_f that leads to a probability that is at least as good. Fix V_f to be such a matrix. This completes the induction step. ■

Proof of Lemma 6(ii): The encoding is done via boolean matrix multiplication $V \times M = E$, where V is the $(cn \times \ell)$ matrix described by the equations for each letter of the encoding and M and E are respectively are the $(\ell \times q)$ and the $(c\ell \times q)$ boolean matrixes representing the message and the encoding. (Recall that a message for this lemma is ℓ letters of q bits each.) As stated a single operation is considered to be the XOR of two q bit letters. This is done once for every non-zero value in V. The number of letters about the the block of the largest size ℓ is $r_1 - r_0 = \mathcal{O}(c\epsilon\ell)$ Each corresponding row of V has at most ℓ ones. Hence, the number of ones in these rows is at most $\mathcal{O}(c\epsilon\ell^2)$. The number of ones in the rows of V for all blocks of size ℓ_s decreases geometrically in s making the total number and the encoding time also $\mathcal{O}(c\epsilon\ell^2)$.

Decoding is a little harder. It requires solving the system $\widehat{V} \times M = \widehat{E}$, where $\widehat{V}$ and $\widehat{E}$ are the rows of the matrices V and E corresponding to those letters of the encoding that are received. Here again, the fact that $\widehat{V}$ is sparse should help. However, inverting a $(\ell \times \ell)$-boolean matrix with only $\mathcal{O}(\ell)$ non-zero entries seems to require more than $\mathcal{O}(\ell^2)$ bit operations. The main reason is that the inverse of a sparse matrix is not necessarily sparse. This is no improvement at all over the standard deterministic MDS erasure codes used above. To improve the computation time, we take advantage of the fact that V has a block hierarchical structure. A $((1+\epsilon)\ell \times \ell)$-boolean matrix $\widehat{V}$ is said to have its rows blocked by $B_1, \dots, B_{(1+\epsilon)\ell} \subseteq \{1,\dots,\ell\}$, if each row is zero outside of its block B_i, i.e. $j \notin B_i$ implies that $\widehat{V}_{\langle i,j\rangle} = 0$. These blocks are said to have a hierarchical structure if the rows are partially ordered with respect to containment of the blocks, i.e. for $i < j$, either $B_i \subseteq B_j$ or $B_i \cap B_j = \emptyset$.

This block hierarchical structure improves the computation time for the following reason. During Gaussian elimination, the i^{th} row may be added to the j^{th} row in order to remove one non-zero entry from the j^{th} row. In a general sparse matrix, the non-zero entries of these rows do not necessarily fall in the same places. Hence, the j^{th} row will gain most of the non-zero entries of the i^{th} row. The effect is that the number grows exponentially with the number of row operations. This provides intuition into why the inverse of a sparse matrix is not necessarily sparse. This exponential growth in the number of non-zero entries, however, does not occur when the matrix has the block hierarchical structure. By the definition of the partial order, B_i is either disjoint from or contained in B_j. In the first case, adding the i^{th} and the j^{th} row would not cancel any entries, hence these rows are never added together. In the second case, adding the i^{th} row to the j^{th} will change which entries in the block B_j are one, but will not contribute ones outside of the block. Hence, as the matrix $\widehat{V}$ is zeroed below the diagonal, the block hierarchical structure is maintained. On the other hand, when zeroing above the diagonal, this structure is destroyed, because the j^{th} row is added to the i^{th} row where $B_i \subset B_j$. In fact, the inverse of $\widehat{V}$ is not likely to be sparse. However, once the matrix is upper triangular, the system can be solved quickly.

The decoding time has two components. We will first consider the number of letter operations within the matrix $\widehat{E}$ and later consider the number of bit operations within $\widehat{V}$. One letter operation within $\widehat{E}$ (i.e. the XOR of two q bit letters) is required for every row operation in $\widehat{V}$. The number of row operations when zeroing below the diagonal is at most $\sum_{j \in \{1,\ldots,(1+\epsilon)m\}} |B_j|$, because the i^{th} row is added to the j^{th} at most once and only if $i \in B_j$. The number of row operations when zeroing above the diagonal is the number of one's that are above the diagonal of the upper triangular matrix. The block structure of the matrix does not change when zeroing below the diagonal. Hence, the number of ones is at most $\sum_{j \in \{1,\ldots,(1+\epsilon)\ell\}} |B_j|$. Note that this is the same as the encoding, except for the fact that there are $(1+\epsilon)\ell$ instead of $c\ell$ rows. Hence, Gaussian elimination of $\widehat{V}$ requires $\mathcal{O}(\epsilon\ell^2)$ row operations and the time is as required in Lemma 6(ii).

If we instead considered a single operation to be a single bit operation, then the time spent in $\widehat{E}$ blows up by a factor of q. On the other hand, measuring the length of the message in bits blows the length up by the same factor. Hence, the time is still linear with the same constant. For the algorithm, the letter size (also the packet size) q could be a single bit. The reason for having it larger is to "amortize" the number of bit operations required to invert $\widehat{V}$.

The main sub-task during Gaussian elimination of $\widehat{V}$ is that of taking the i^{th} row, which has a one on the diagonal, and adding it to every row below it that contains a one in the i^{th} column. Let us compute the number of bit operations required to do this. The first obvious savings in time comes from never looking at or even storing the entries of $\widehat{V}$ outside of the block structure. There are, however, two additional tricks that save a considerable amount of time.

The first trick is not to check every row $j > i$, but only those for which $B_i \subseteq B_j$. This can be done by associating with each row i, a pointer to the next row j for which $B_i \subseteq B_j$. A property of the partial order is that following this linked list of pointers starting at any row i will reach every row j for which $B_i \subseteq B_j$. (The same would not be true in reverse.) If B_i is of size ℓ_s, then the expected number of rows j for which $B_i \subseteq B_j$ is at most the geometric sum $\sum_{s' \in \{0,\ldots,s\}} \frac{1+\epsilon}{c}(r_{s'+1} - r_{s'})/2^{s'} = \mathcal{O}(\epsilon\ell)$.

What now is the cost each time the i^{th} row is added to some row j? The second trick is that this cost should be the number of ones in the i^{th} row and not the size ℓ_s of B_i. There are two ways of achieving this. Either at the beginning of this sub-task a succinct list of the ones of the i^{th} row can be made or the block B_i in the i^{th} row can scanned and each time a one is found the above linked list giving all rows j for which $B_i \subseteq B_j$ could be followed. The expected number of ones in the i^{th} row is ℓ_s minus the number of entries that have already been zeroed. The number of entries that have been zeroed is the number of rows j before it for which $B_j \subseteq B_i$. This is close to the expected number of letters contributing to the block and was computed as being $(1 - \mathcal{O}(\epsilon 2^{s/2}))\ell_s$. Hence, the total number of ones remaining is at most $\mathcal{O}(\epsilon 2^{-s/2}\ell)$.

The expected number of rows i whose block B_i is of size ℓ_s is $\frac{1+\epsilon}{c}(r_{s+1} - r_s) = \mathcal{O}(\epsilon 2^{s/2}\ell)$. We can conclude that the total number of bit operations in the Gaussian elimination then is $\sum_{s \in \{0,\ldots,f\}} \mathcal{O}(\epsilon 2^{s/2}\ell) \times \mathcal{O}(\epsilon\ell) \times \mathcal{O}(\epsilon 2^{-s/2}\ell) = \mathcal{O}(\log(1/\epsilon^2)\epsilon^3\ell^3)$. If a single operation is considered to be $q = \mathcal{O}(\ln(1/\epsilon)\epsilon^2\ell)$ bit operations, then decoding only takes $\mathcal{O}(\epsilon\ell^2)$ operations as required. ■

In the full version of the paper, we prove the lower bound, that for no setting of the parameters (i.e. the sizes of the blocks and the number of letters about each letter) are the encoding and decoding times for this scheme more than a constant factor better than we have achieved. It is interesting that this goes against our initial intuition. Initially, we planned for the expected number of letters received contributing to a block of size ℓ_s to be larger than $(1 - \Omega(\epsilon 2^{s/2}))b_s$. In such a case, it is very possible that a particular block is over defined, i.e. $c_{\langle s,i\rangle} > \ell_s$. However, averaged over many such blocks, with high probability not too many letters of the encoding will be wasted in this way. The advantage of doing this is that having more small blocks decrease the computation time. It turns out, however, that a large fraction of the computation time is spent on the letters about the blocks of size ℓ and making this change means increasing the number of them and hence increases the overall time.

5 Concluding remarks and open problems

It would be interesting to obtain a construction similar to the one in Theorem 1 in which the packet size is smaller. It is not difficult to prove, using the Plotkin bound, that the minimum possible packet size in any erasure code with the parameters in the theorem (without any assumption on the efficiency of its encoding and decoding procedures) is at least $\Omega(\log((c-1)/\epsilon))$ for all $\epsilon \geq 1/n$, and the algebraic geometry codes show that this is essentially tight. Our construction supplies much bigger packet sizes, but has the advantage of linear encoding and decoding time.

It is probably possible to construct a scheme that has a theoretical run time that is polylogarithmic in $1/\epsilon$ and linear in n, using Discrete Fourier Transform methods in place of quadratic time methods for MDS codes, but details of this need to be checked. Even if this is the case, it is unlikely that using these methods in places where we use quadratic time MDS codes will be as efficient in practice.

The construction in Section 3 can be improved by using walks in expanders instead of edges, using the methods of [1]. The relevance of this method to the case of expander based error correcting codes has been observed in (cf. [12]), and a similar remark holds here also.

Combining our technique here with the methods of Spielman in [13] we can obtain explicit, linear time encodable and decodable error correcting codes over a large alphabet, whose rate and minimum distance in the range close to the MDS bound are close to optimal. We omit the details.

References

[1] M. Ajtai, J. Komlós, E. Szemerédi, "Deterministic Simulation in Logspace", *Proc. of the* 19^{th} *STOC*, 1987, pp. 132-140.

[2] A. Albanese, J. Blömer, J. Edmonds, M. Luby, M. Sudan, "Priority Encoding Transmission", *Proceedings of* 35^{th} *FOCS*, 1994.

[3] A. Albanese, J. Blömer, J. Edmonds, M. Luby, "Priority Encoding Transmission", *ICSI Technical Report No. TR-94-039*, August 1994.

[4] N. Alon, J. Bruck, J. Naor, M. Naor, R. Roth, "Construction of asymptotically good, low-rate error-correcting codes through pseudo-random graphs", *IEEE Transactions on Information Theory*, Vol. 38, 1992, pp. 509-516.

[5] N. Alon, F. R. K. Chung, "Explicit construction of linear sized tolerant networks", *Discrete Math.*, Vol. 72, 1988, pp. 15-19; (*Proc. of the First Japan Conference on Graph Theory and Applications*, Hakone, Japan, 1986.)

[6] N. Alon, J. H. Spencer, **The Probabilistic Method**, Wiley, 1991.

[7] L. A. Bassalygo, V. V. Zyablov, M. S. Pinsker, "Problems in complexity in the theory of correcting codes", *Problems of information transmission*, 13, Vol. 3, 1977, pp. 166-175.

[8] E. Biersack, "Performance evaluation of forward error correction in ATM networks", *Proceedings of SIGCOMM '92*, Baltimore, 1992.

[9] A. Lubotzky, R. Phillips, P. Sarnak, "Explicit expanders and the Ramanujan conjectures", *Proceedings of* 18^{th} *ACM STOC*, 1986, pp. 240-246; (See also: A. Lubotzky, R. Phillips, P. Sarnak, "Ramanujan graphs", *Combinatorica*, Vol. 8, 1988, pp. 261-277).

[10] G. A. Margulis, "Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators" *Problemy Peredachi Informatsii*, Vol. 24, 1988, pp. 51-60 (in Russian). (English translation in *Problems of Information Transmission*, Vol. 24, 1988, pp. 39-46).

[11] M. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", *J. ACM*, Vol. 36, No. 2, April 1989, pp. 335-348.

[12] M. Sipser and D. Spielman, "Expander codes", *FOCS* 1994.

[13] D. Spielman, "Linear-Time Encodable and Decodable Error-Correcting Codes" *STOC* 1995, ACM Press, 388-397.