Erasure Codes with a Hierarchical Bundle Structure

Jeff Edmonds *† Michael Luby ‡

November 14, 2017

Abstract

This paper presents a proof of the existence of computationally fast probabilistic erasure codes at distance ϵ from being MDS, namely the decoding algorithm is able with high probability to reconstruct the n letter message from any set of $(1+\epsilon)n$ letters. It can either be fixed rate or a rateless LT code [10] in that any number of code letters can be produced and each is produced independently of the others. We also decrease the minimum packet size from many to one letter. The key ingredient is a scheme *Hierarchical Bundle/Bin (HB)* which splits the message into a hierarchy of disjoint bundles and produces coded packets about each bundle. We show a correspondence of this to a particular game having to do with randomly throwing balls into a hierarchy of bins. The "information" that does not over flow from a smaller bin, contributes to the next larger bin that it is contained in. We prove matching upper and lower bounds on the cost of this game and provide the implementation details. This analysis is somewhat analogous to the evolution of the "ripple" in the LT decoding analysis [10]. The bundle size corresponds to the degree of the packet, therefore, smaller bundles tend to reduce encoding/decoding complexity, but packets coming from larger bundles ensure the approximately-MDS constraint, by ensuring more coverage. Our HB scheme with largest block size b requires encoding and decoding time $\mathcal{O}(\epsilon b^2)$ rather than the $\mathcal{O}(b^2)$ needed for Reed-Solomon codes. This scheme HB (together with Spielman's expanders) gives a probabilistic code with running time $\mathcal{O}(\epsilon^{-1}\ln(\epsilon^{-1})n)$. Alon and Luby [5, 6] simultaneously developed a deterministic version but their running time is $\mathcal{O}(\epsilon^{-4}n)$. Both our and Alon's results have since been completely subsumed by the latest generation of Shokrollahi and Luby's Raptor codes [13, 14, 11].

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. Erasure codes allow decoding the message even when many of packets have been lost. Such codes are important for applications such as multicasting real-time high-volume video information over lossy packet based networks [2, 3, 9] and other high volume real-time applications [12].

A common scheme for encoding an n letter message is to break it into bundles of size b and separately encoding each bundle. Reed-Solomon codes required time $\mathcal{O}(b^2)$ per bundle for a total of $\mathcal{O}(b^2) \times \frac{n}{b} = \mathcal{O}(bn)$ time[1]. Suppose all of the code letters are permuted randomly before placing

*Computer Science Department, York University, York, Canada. Email: jeff@cse.yorku.ca.

†This paper was presented at 36th *FOCS*, pp. 512-519, 1995.

‡Qualcomm Technologies, Inc. Email: luby@qti.qualcomm.com. Research done while both were at the International Computer Science Institute.

[1]If we encode the b letters into cb, we ignore the linear dependence on c because it tends to be small.

them in packets and we receive $(1+\epsilon)n$ packets[2]. Then the number received about a given bundle is binomial with mean $(1+\epsilon)b$. Being a *maximal distance separable* (MDS) code, a bundle is recovered iff this number is at least b. The trade off in choosing b is between the running time and the probability of being able to successfully decode each bundle.

Our first scheme *Hierarchical Bundle/Bin (HB)* decreases time for a single bundle from $\mathcal{O}(b^2)$ to $\mathcal{O}(\epsilon b^2)$ by having a hierarchy of bundles. The bundle of size b is broken further into smaller bundles, which are broken again and so on. The scheme will produce packets *about* each level of bundle. The smaller the bundle, the less time will be used to produce a packet about it. The larger the bundle, the smaller the probability of there not being enough packets received *contributing* to this bundle. This can easily be modeled in a game having to do with randomly throwing balls into a hierarchy of bins. We prove matching upper and lower bounds on this stated trade off.

A common way to view an encoding scheme is to consider the bipartite graph with an edge between a packet and a letter of the message if the first depends on the second [4, 10, 11, 13, 14, 15, 16]. Often, the nodes of these graphs are of fixed degree. In contrast, in LT codes [10] and here the degree b_i of a packet is randomly chosen from a degree distribution. In both [10] and here, the design and analysis of a good degree distribution is a primary focus of the paper. These degrees have an exponential range in order to get as said both the time advantage of the smaller degree packets and the coverage advantage of the larger degree ones helping to ensure the approximately-MDS constraint. Another difference is that generally, the packet/letter bipartite graph is an *expander graph* so that each new packet received gives the maximum amount of new information about the message. In contrast, our graph is very structured (see Figure 5) so that packets about a small bundle of the message can also contribute to a larger bundle that it is contained in. We ensure that we do not receive more information than is useful about any particular bundle of the message. This structure is also designed to be retained during the decoding process in order to make it more computationally efficient.

LT codes introduced by Luby [10] are rateless, i.e., the number of encoding letters that can be generated from the data is potentially limitless. Furthermore, encoding symbols can be generated on the fly, as few or as many as needed. The key is that the decoder can recover the entire message with high probability from any set of the generated encoded letters that is only slightly longer in length than the data. Thus, no matter what the loss model is on the erasure channel, encoding letters can be generated as needed and sent over the erasure channel until a sufficient number have arrived at the decoder in order to recover the data. Our HB scheme can either be such a code or can encode each b letter bundle into cb letters for a fixed rate c. In the first paradigm, independently for each packet, the encoder randomly chooses which message bundle of which size b_i it should be about and then like in [10] randomly chooses a binary vector of length b_i corresponding to the row of the encoding matrix $V \times M = E$. The contents of the packet will simply be the GF$[2^\ell]$ dot product of this vector with the message ℓ bit letters in the bundle. The disadvantage of this is that this b_i bit encoding vector must be included in the header of the packet so that the decoder knows it. To amortize the size of this header, the letter size ℓ of the packet should be at least the same number $\ell = b$ of bits. This would mean that the entire message bundle would contain b letters of b bits each for a total of b^2 bits. In contrast, if the rate is fixed to some c, then these encoding vectors can be fixed ahead of time and known by both the encoder and the decoder, and hence not need to be included in the header. An inadvertent disadvantage of this is that though we prove that a

[2]The network is assumed to drop packets independent of their contents. We also assume that each packet contains a unique identifier.

random $\left(\left(\frac{w_i}{1+\epsilon}cb_i\right) \times b_i\right)$ binary matrix V_i for each bin/bundle size $i \in [1..s]$ works for this purpose with high probability, it is unknown how to construct such matrices. (After fixing the matrices V_i, the scheme is deterministic except for randomly permuting the packets.)

Another advantage of a fixed rate scheme is that then it can be what is called *systematic* in the literature, which means that the message itself is part of the encoding and more over the time to decode the message from the encoding depends primarily only on the amount of the message that is missing. This property is desirable especially in the case when only a small number of the packets are lost.

Another reason our scheme is fast is that the basic operation is bit-wise XORs as opposed to operations over a finite field. Throughout, our basic unit of length is a *letter*, which is a bit string of length ℓ. Each packet receives the one letter which is a linear combination of the letters in the message bundle that it is *about*. Encode and decode the message requires such $\mathcal{O}(\epsilon b^2)$ letter operations. For the algorithm, the letter size ℓ could be a single bit. One reason for having it larger is to allow for header information. Another reason is to amortize the number $\mathcal{O}(\ln(1/\epsilon^2)\epsilon^3 b^3)$ of additional bit operations required to decode. If we set ℓ to be $\mathcal{O}(\ln(1/\epsilon)\epsilon^2 b)$, then the number of "letter" operations to decode is $\mathcal{O}(\epsilon b^2)$ as required. In order to emphasize this, we expressed in Figure 1 the total decoding time as being $\mathcal{O}(\epsilon b^2) \times (1+\mathcal{O}(\epsilon^2 \log(\epsilon^{-1})b/\ell))$.

Returning now to encoding a message consisting of $\frac{n}{b}$ bundles of length b. If we do not want the decoding of any of the bundles to fail, when the failure probability for each bundle is $e^{-\Omega(\epsilon^2 b)}$, then we need to set the bundle size to $b = \mathcal{O}(\ln(n))$. The total time $\mathcal{O}(\epsilon bn)$ then is super linear. If, however, we only require a $1-\beta$ fraction of the bundles to be decoded, then we are able to decrease the bundle size to only $b = \ln(\beta^{-1})\epsilon^{-2}$, and hence the running time to only $\mathcal{O}(\epsilon bn) = \mathcal{O}(\ln(\beta^{-1})\epsilon^{-1}n)$, while keeping the failure probability exponentially small. This we call scheme *HB*$'$. See Figure 1.

Finally, scheme *HB*$''$ is able to recover all n-bits of the message because it is made up of two encoding/decoding phases. In the first, the n-bit message M undergoes an expansion by a factor of $1+\mathcal{O}(\epsilon)$ forming the intermediate message/code M'. It is this M' that is broken into bundles that are separately encoded in the second phase. Only a $1-\beta = 1-\epsilon^2$ fraction of these bundles need to be decoded in the second phase because the first phase only requires this fraction of M' to recover the entire message M. Plugging in $\beta = \epsilon^2$, gives a running time of $\mathcal{O}(\epsilon^{-1}\log(\epsilon^{-1})n)$. In contrast, Alon and Luby [5, 6] simultaneously developed a deterministic version but their running time is $\mathcal{O}(\epsilon^{-4}n)$. The time needed for Luby's LT code [10] is $\mathcal{O}(\log(n)n)$ with $\epsilon = \mathcal{O}(\frac{1}{\sqrt{n}})$ but with polynomial instead of exponential error probability. Shokrollahi [13] improves this time with *Raptor codes* which require time $\mathcal{O}(\log(\epsilon^{-1})n)$.

The encoding scheme for the first phase of HB$''$ was developed in [5, 6] and is based on a breakthrough result of Spielman [16] on error correcting codes. It uses *expanders* which are explicit graphs with pseudo-random properties.

Section 2 defines the hierarchical bin game and proves the matching upper and lower bounds for it. Section 3 describes how this ball game translates into scheme HB. Section 4 constructs scheme HB$'$ from HB and constructs HB$''$ from Expanders and HB$'$. The description of scheme Expanders is not included here, but is found in [5, 6].

Scheme	MDS	HB	HB$'$	HB$''$	Raptor	Expanders
Message Leng	b		n			
Code Length	cb		cn			$(1+\epsilon)n$
Code Needed	b	$(1+\epsilon)b$	$(1+\epsilon)n$			$(1-\beta)(1+\epsilon)n$
Mess. Acquired	all		$(1-\beta)n$	all		
Failure Prob	zero	$e^{-\Omega(\epsilon^2 b)}$	$e^{-\Omega(\beta\epsilon^2 n)}$	$e^{-\Omega(\epsilon^4 n)}$	$n^{-\Omega(1)}$	zero
Time	$\mathcal{O}(b^2)$	$\mathcal{O}(\epsilon b^2)\times$ $\left(1+\tilde{\mathcal{O}}(\epsilon^2 b/\ell)\right)$	$\mathcal{O}(\ln(\beta^{-1})\epsilon^{-1}n)\times$ $\left(1+\mathcal{O}(\ln(\beta^{-1})\ln(1/\epsilon)/\ell)\right)$	$\tilde{\mathcal{O}}(\epsilon^{-1}n)\times$ $\left(1+\mathcal{O}(\ln^2(1/\epsilon)/\ell)\right)$	$\tilde{\mathcal{O}}(n)$	$\mathcal{O}(\epsilon^{-1}n)$
Packet Size ℓ	1 f.e.	1				1 f.e.

Figure 1: The six encoding schemes discussed in this paper are compared in this table. The scheme *MDS* is the standard quadratic time Reed-Solomon *maximal distance separable* (MDS) code. The scheme *HB* is the our hierarchy of bundles scheme that encodes each bundle of size b. The scheme *HB$'$* applies *HB* to each bundle. The scheme *HB$''$* first applies Spielman's Expanders and then applies HB$'$. This result has since been completely subsumed by the latest generation of Shokrollahi and Luby's Raptor codes. In all six schemes, a message of the stated size is encoded into a code of the stated size partitioned into packets of the stated size. Given any subset of the packets with the stated total size, the message (or most of it) can be decoded. If it is probabilistic, the failure probability is given. The encoding and decoding times are also given. All sizes are measured in number of letters and the running time in number of letter operations. For some the packet size is a single field element (f.e.) needing $\mathcal{O}(\ln(\epsilon^{-1}))$ bits. For others a packet could contain as few as $\ell = 1$ bit. However, if we set ℓ to be $\tilde{\mathcal{O}}(\epsilon^2 b)$ or $\mathcal{O}(\ln^2(1/\epsilon))$, then this amortizes the number of bit operations required to invert $\widehat{V}$. Here $\tilde{\mathcal{O}}$ means an extra factor of $\ln(\epsilon^{-1})$. All schemes are systematic meaning the code contains the message.

2 A Hierarchy of Bin/Bundle Sizes

Varying the bundle size b yields a tradeoff between the computation time and the probability of success. We can optimize both metrics by using a hierarchy of bundle sizes. This is modeled by the following hierarchical bin game.

Given b and ϵ, the task is to set the parameters $b_1 < b_2 < \ldots < b_s = b$ and $0 < w_1, w_2, \ldots, w_s$, where $\sum_i w_i = 1+\epsilon$ in the way to minimize the *cost* of the game while maintaining a sufficiently high probability of the game *winning*. Given all the parameters, the game is as follows. There is one bin of size b_s. Poised over this bin are b_s/b_{s-1} bins of size b_{s-1}. Similarly, poised over each bin of size b_i are b_i/b_{i-1} bins of size b_{i-1}. Let $B_{\langle i,j\rangle}$ be the j^{th} bin of size b_i. We use the notation

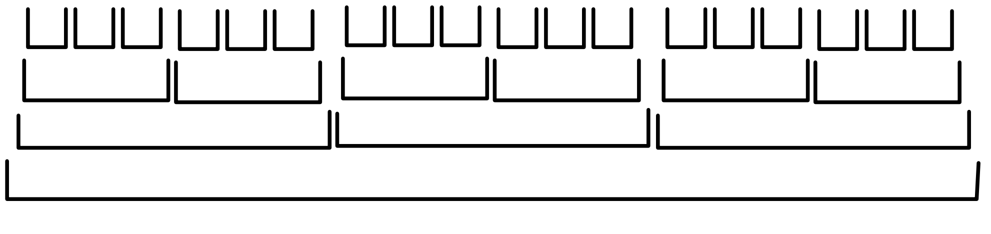

Figure 2: A Hierarchy of Bins

$B_{\langle i-1,j'\rangle} \subseteq B_{\langle i,j\rangle}$ to indicate that bin $B_{\langle i-1,j'\rangle}$ is poised over bin $B_{\langle i,j\rangle}$.

The game throws some number of balls into each bin and independently each ball fails to land with some probability so that the expected number to land in a given bin of size b_i is $w_i b_i$. Note that there are b/b_i bins of this size. Hence, the total number of balls expected to land is $\sum_i (w_i b_i)(b/b_i) = (\sum_i w_i)b = (1+\epsilon)b$. (Clearly a ball landing in a bin of size b_i corresponds to receiving a packet about a bundle of this size).

The game then proceeds by pouring the contents of each bin into the bin of the next largest size below it starting with smallest. Hence, a ball that is thrown into a bin of the smallest size will progress through the hierarchy until it reaches the single bin of the largest size. However, if a bin ever contains more balls then its size then the excess balls *overflow* from the bin and are lost. More formally, balls that land directly into a bin are said to be *about* the bin, balls that land or are poured into a bin are said to *contribute* to the bin, and the number of these balls that do not overflow out of the bin are said to be *useful* to the bin. We will denote the number of such balls by $a_{\langle i,j\rangle}$, $c_{\langle i,j\rangle}$, and $u_{\langle i,j\rangle}$, respectively. Then $Exp(a_{\langle i,j\rangle}) = w_i b_i$, $c_{\langle i,j\rangle} = a_{\langle i,j\rangle} + \sum_{(B_{\langle i-1,j'\rangle} \subseteq B_{\langle i,j\rangle})} u_{\langle i-1,j'\rangle}$ and $u_{\langle i,j\rangle} = \min(c_{\langle i,j\rangle}, b_i)$. We say a bin is *full* if $u_{\langle i,j\rangle} = b_i$. The game *wins* if the single bin of the largest size b is full. The *cost* of throwing a ball into a bin is the size b_i of the bin. The *cost of the game* is the expected sum of the costs of the balls that land[3], which is $\sum_i (w_i b_i)(b/b_i)b_i = (\sum_i w_i b_i)b$. As said, the task given b and ϵ is to set the parameters b_i and w_i in the way that minimize the *cost* of the game while maintaining a sufficiently high probability of the game *winning*. For example, suppose that all the balls are thrown into the bins of some fixed size b_i, i.e., $w_i = (1+\epsilon)$ and for $i' \neq i$, $w_{i'} = 0$. Then the single bin of the size b is full if and only if all of these bins of size b_i are full, because the sum of their sizes is b.

Lemma 1 *When the total number of balls expected to land is* $(1+\epsilon)b$*, the optimal setting of the parameters* b_i *and* w_i *subject to maintaining a success probability of at least* $1 - \mathrm{e}^{-\Omega(\epsilon^2 b)}$ *yields a cost of* $(\sum_i w_i b_i)b = \Theta(\epsilon b^2)$.

Proof of Lemma 1 (upper bound): For $s = \log(\frac{1}{\epsilon^2})$ and $i \in [1..s]$, set $b_i = b/2^{s-i}$ so that each bin has two bins of half its size poised over it. The smallest bin is of size $b_1 = b/2^{s-1} = 2\epsilon^2 b$ and the largest of size $b_s = b$. We will set w_i so that with high probability no balls overflow from bins except possibly from the single bin of size b. If even a single ball does overflow from even one bin, then we will assume the entire game fails. We will define q_i and set $w_1 = 1 - q_1$, for $i \in [2..s-1]$, $w_i = q_{i-1} - q_i$, and $w_s = q_{s-1} + \epsilon$. Note that $\sum_i w_i = 1 + \epsilon$ as required. Later we set $q_i \approx \epsilon\sqrt{b/b_i}$, ranging from $q_1 = \epsilon\sqrt{\frac{b}{2\epsilon^2 b}} = 0.71$ and $q_s = \epsilon\sqrt{\frac{b}{b}} = \epsilon$. This gives $w_1 \approx 0.29$, $w_i \approx 0.41\epsilon\sqrt{b/b_i} = 0.41 \cdot 2^{-i/2}$, and $w_s \approx 2.41\epsilon$.

Because no bin overflows, the number of balls contributing to the bin $B_{\langle i,j\rangle}$ is the sum landing into it or any bin above it, namely $\mathrm{Exp}(c_{\langle i,j\rangle}) \leq \sum_{i' \in [1..i]} \frac{b_i}{b_{i'}} w_{i'} b_{i'} = \left(\sum_{i' \in [1..i]} w_{i'}\right) b_i = (1 - q_i)b_i$. Chernoff then bounds the probability that there is overflow from this bin to be $\Pr\left[c_{\langle i,j\rangle} > b_i\right]$ $\leq \mathrm{e}^{-\frac{1}{2}\frac{(q_i b_i)^2}{(1-q_i)b_i}} \leq \mathrm{e}^{-\frac{q_i^2 b_i}{2}}$. (Note that for larger b_i, it is less likely that $c_{\langle i,j\rangle}$ will deviate far from its expectation, causing overflow. Hence, q_i can safely be smaller.) To obtain our desired probability of failure, we set $q_i = \sqrt{\frac{\epsilon^2 b + 4(s-i)}{b_i}}$ $(\approx \epsilon\sqrt{b/b_i})$. This gives $\Pr\left[c_{\langle i,j\rangle} > b_i\right] \leq \mathrm{e}^{-2(s-i)} \times \mathrm{e}^{-\frac{1}{2}\epsilon^2 b}$. There are $b/b_i = 2^{s-i}$ bins of size b_i, giving that the probability of overflow in some bin excluding the largest is at most $\sum_{i \in [1..s-1]} 2^{s-i} \times \mathrm{e}^{-2(s-i)} \times \mathrm{e}^{-\frac{1}{2}\epsilon^2 b} \leq \mathrm{e}^{-\frac{1}{2}\epsilon^2 b}$.

[3]The cost of the scheme is the number thrown which is a factor of $\frac{c}{1+\epsilon}$ more which because c is assumed small is ignored.

If there is no overflow from bins except possibly the largest one, then every ball thrown contributes to the largest bin. The total number of balls thrown is Binomial with mean $(1+\epsilon)b$. Hence, the probability that fewer than b are thrown is at most $\mathrm{e}^{-\Omega(\epsilon^2 b)}$. This completes bound on the probability of the game failing.

What remains is to bound the cost $(\sum_i w_i b_i)b$ of the game with these parameters. Recall $w_s \times b_s = [q_{s-1} + \epsilon] \times b = \left[\sqrt{\frac{\epsilon^2 b + 4(s-(s-1))}{b_{s-1}}} + \epsilon\right] \times b = \mathcal{O}(\epsilon b)$. The term $w_i \times b_i$, when calculated similarly, decreases geometrically as i decreases. Namely, $[w_{i+1} \times b_{i+1}]/[w_i \times b_i] = [(q_i - q_{i+1}) \times b_{i+1}]/[(q_{i-1} - q_i) \times b_i]$. Recall that $b_{i+1} = 2b_i$ and factor out b_i. Let $a = \epsilon^2 b + 4(s-i)$. This gives a ratio of $[(\sqrt{a} - \sqrt{\frac{a-4}{2}}) \times 2]/[(\sqrt{\frac{a+4}{1/2}} - \sqrt{a}) \times 1] \geq 0.9\sqrt{2}$ (seen by plotting). Thus the cost of the game is $(\sum_i w_i b_i)b = \mathcal{O}(\epsilon b)b$. ■

Proof of Lemma 1 (lower bound): Consider a setting of the parameters $b_1 < b_2 < \ldots < b_s$ and $w_1, w_2, \ldots, w_s$ for which the cost of the game $(\sum_{i\in[1..s]} w_i b_i)b$ is $o(\epsilon b^2)$. We prove that the probability the game fails is at least $\mathrm{e}^{-o(\epsilon^2 b)}$.

Figure 3: We consider the underflow μ on the first m bins of medium size and all the larger bins they pour into.

We will carefully select one index $\hat{i} \in [1..s]$ and call the bin size $b_{\hat{i}}$ *medium.* We will fix $m \in [1..b_{\hat{i}+1}/b_{\hat{i}}]$ and consider the first m bins $B_{\langle\hat{i},1\rangle}$, $B_{\langle\hat{i},2\rangle}$, $\ldots$,$B_{\langle\hat{i},m\rangle}$ of this size. See Figure 3. Recall that $a_{\langle\hat{i},j\rangle}$ is the number of balls *about* bin $B_{\langle\hat{i},j\rangle}$ because they land directly into it, $c_{\langle i,j\rangle}$ is the number that *contribute* to it because they land or are poured into it, and $u_{\langle i,j\rangle}$ is the number that are *useful* because they do not over flow but are poured into the next bin size. We will assume that no smaller bin over flows and hence the expected number of balls contributing to a medium sized bin is $\text{Exp}(c_{\langle\hat{i},j\rangle}) = \sum_{i\in[1..\hat{i}]}(w_i b_i)(b_{\hat{i}}/b_i) = \left(\sum_{i\in[1..\hat{i}]} w_i\right) b_{\hat{i}} = (1+q)b_{\hat{i}}$. Here q is set so that $qb_{\hat{i}}$ is the expected amount to over flow from such a bin. By the game $\sum_{i\in[1..s]} w_i = 1+\epsilon$ and hence $q \leq \epsilon$. Though we expect excess balls in this bin, we may be unlucky and have *underflow* in it. Let $\mu_{\langle\hat{i},j\rangle} = \max(b_{\hat{i}} - c_{\langle\hat{i},j\rangle}, 0)$ be the amount of this under flow and let $\mu = \sum_{j\in[1..m]} \mu_{\langle\hat{i},j\rangle}$ be the total underflow in these bins we are considering. If we are going to succeed to get b balls in the bottom bin, then this μ underflow will need to be made up with balls these medium bins are poised over. Because $m \leq b_{\hat{i}+1}/b_{\hat{i}}$, these m medium bins of size $b_{\hat{i}}$ are all poised over the same first bin $B_{\langle\hat{i}+1,1\rangle}$ of the next larger size $b_{\hat{i}+1}$. This in turn is poised over the sequence $B_{\langle\hat{i}+2,1\rangle}$, $B_{\langle\hat{i}+3,1\rangle}, \ldots$, $B_{\langle s,1\rangle}$ of first bins of larger size. Let a be the random variable indicating the number of balls landing in these larger first bins, namely $a = \sum_{i\in[\hat{i}+1,s]} a_{\langle i,1\rangle}$. The proof proceeds by proving that if $\mu > a$, then the game will fail and proving that $\Pr(\mu > a) \geq \mathrm{e}^{-o(\epsilon^2 b)}$. Thus, as needed the game fails with at least this probability.

To help understand why the game will fail if $\mu > a$, briefly recall the information theory, message encoding interpretation of the game. The bins $B_{\langle \hat{i},1\rangle}$, …,$B_{\langle \hat{i},m\rangle}$ corresponds to bundles of the message consisting of $mb_{\hat{i}}$ letters. About bundles of this size or smaller there are only $mb_{\hat{i}} - \mu$ code letters giving information about these. There are only a code letters about larger bundles also containing these same message letters. Hence if $\mu > a$ then there are fewer $mb_{\hat{i}}$ code letters about these $mb_{\hat{i}}$ message letters. Hence by information theory, the message letters cannot be reconstructed and the game fails. Let us reprove this based on the counting of balls. Suppose that each of the not yet considered $b_{\hat{i}+1}/b_{\hat{i}} - m$ medium sized bins $B_{\langle \hat{i},m+1\rangle}, \ldots, B_{\langle \hat{i},b_{\hat{i}+1}/b_{\hat{i}}\rangle}$ poised over $B_{\langle \hat{i}+1,1\rangle}$ receive more than $b_{\hat{i}}$ balls each. All but $b_{\hat{i}}$ of these balls over flow and hence at most $b_{\hat{i}}$ are poured into $B_{\langle \hat{i}+1,1\rangle}$. Because the first m medium bins have a total under flow of μ, the total number of balls poured into $B_{\langle \hat{i}+1,1\rangle}$ is $b_{\hat{i}} \times b_{\hat{i}+1}/b_{\hat{i}} - \mu = b_{\hat{i}+1} - \mu$. This bin has $a_{\langle \hat{i}+1,1\rangle}$ balls land directly into it giving $b_{\hat{i}+1} - \mu + a_{\langle \hat{i}+1,1\rangle}$ useful to it. Because $b_{\hat{i}+1}$ is the size of this bin, it has underflow of $\mu - a_{\langle \hat{i}+1,1\rangle}$. We carry this same idea down one bin size at a time for the first bin of the size. In the end, $b - \mu + \sum_{i\in[\hat{i}+1,s]} a_{\langle i,1\rangle} = b - \mu + a$ balls contribute to the single bin of largest size b. Hence if $\mu > a$, this bin will still have underflow and the game will fail.

We now show that $\Pr(\mu > a) \geq \mathrm{e}^{-o(\epsilon^2 b)}$. Recall that the expected number of balls contributing to a medium sized bin is $\mathrm{Exp}(c_{\langle \hat{i},j\rangle}) = (1+q)b_{\hat{i}}$. Getting underflow in one of these bins first requires the unlucky event that the $qb_{\hat{i}}$ excess balls fails to arrive. Once we have paid for the probability of this, the probability is not change much by considering the event that $2qb_{\hat{i}}$ fail to arrive giving an underflow of $\mu_{\langle \hat{i},j\rangle} = qb_{\hat{i}}$. Using Chernoff bounds, the probability of receiving $2qb_{\hat{i}}$ below $c_{\langle \hat{i},j\rangle}$'s expected value $(1+q)b_{\hat{i}}$ is at least $exp[-\frac{(2qb_{\hat{i}})^2}{2\cdot(1+q)b_{\hat{i}}}] \geq \mathrm{e}^{-\mathcal{O}(q^2 b_{\hat{i}})}$. If this happens in each of the first m medium sized bins then the total underflow is $\mu = mqb_{\hat{i}}$. Recall that $a = \sum_{i\in[\hat{i}+1,s]} a_{\langle i,1\rangle}$ is the number of balls about the first bin of each larger size. With probability at least $\frac{1}{e}$, it is at most its expected value $\overline{a} = Exp[a] = \sum_{i\in[(\hat{i}+1)..s]} w_i b_i$. We want $\mu > \overline{a}$ and hence set $m = \overline{a}/(qb_{\hat{i}})$ so that $mqb_{\hat{i}} = \overline{a}$. We can then compute $\Pr[$ the game fails $] \geq \Pr[\mu > a]$ $\geq \Pr[\ a < \overline{a}$ and $\forall j \in [1..m]\ c_{\langle \hat{i},j\rangle} \leq b_{\hat{i}} - qb_{\hat{i}}\ ] \geq \Pr[a < \overline{a}] \times \Pr[c_{\langle \hat{i},j\rangle} \leq b_{\hat{i}} - qb_{\hat{i}}]^m \geq \frac{1}{e} \times \mathrm{e}^{-\mathcal{O}(q^2 b_{\hat{i}})m}$ $= \mathrm{e}^{-\mathcal{O}(q\overline{a})}$. Recall $q \leq \epsilon$. The advantage of considering only one bin of each larger size is that $\overline{a} = \sum_{i\in[(\hat{i}+1)..s]} w_i b_i$ is at most the cost $\sum_{i\in[1..s]} w_i b_i$ of the game, which by assumption is $o(\epsilon b)$. In conclusion, the game fails with probability $\mathrm{e}^{-\mathcal{O}(q\overline{a})} = \mathrm{e}^{-o(\epsilon^2 b)}$.

What remains is to carefully select the one index $\hat{i} \in [1..s]$. The only requirement used is that $m = \overline{a}/(qb_{\hat{i}})$ is a number of medium size bins that are all poised over the first next size bin. This requires that $1 \leq \overline{a}/(qb_{\hat{i}}) \leq b_{\hat{i}+1}/b_{\hat{i}}$ or that $qb_{\hat{i}} \leq \overline{a} \leq qb_{\hat{i}+1}$. To help choose $\hat{i}$, it would be considerably easier if $\hat{i}$ was almost continuous in some range, namely can be increased in infinitesimal increments, instead of being an integer. Towards this goal, we modify our bin structure in the following way that has no effect on its cost or the success probability of its game. Instead of having a single row of bins of size b_i, each expecting to have $w_i b_i$ balls land in it, split it into w_i/δ separate rows of bins of this same size b_i, each expecting to have δb_i balls land it. Reindex all the rows. Note that if such an intermediate bin size $\hat{i}$ is chosen, then there is only one bin of this size poised over the next bin size and hence $m \in [1..b_{\hat{i}+1}/b_{\hat{i}}]$ is restricted to being one. However, this poses no problem. Note that as $\hat{i}$ is increased from zero almost continuously, we have: $q = (\sum_{i\in[1..\hat{i}]} w_i) - 1$ also increases almost continuously from -1 to ϵ; $qb_{\hat{i}}$ increase piecewise linearly from something negative to something positive; and $\overline{a} = \sum_{i\in[(\hat{i}+1)..s]} w_i b_i$ decreases from something positive almost continuously to zero. See Figure 4. Because $qb_{\hat{i}}$ starts below $\overline{a}$ and ends above it, by the intermediate value theorem, they must cross at some value. Set $\hat{i}$ to be this value. If they cross at a discontinuity of $qb_{\hat{i}}$, then $b_{\hat{i}}$ and and $b_{\hat{i}+1}$ will be two different original bin sizes and we will have as needed that $qb_{\hat{i}} \leq \overline{a} \leq qb_{\hat{i}+1}$. If the functions cross at a continuous place in $qb_{\hat{i}}$, then we do consider this bin size $b_{\hat{i}}$ to be split in two at this point and we have that $qb_{\hat{i}} = \overline{a} = qb_{\hat{i}+1}$. ■

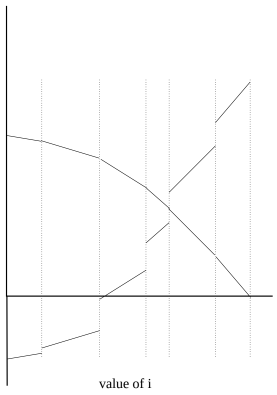

Figure 4: A plot of $qb_{\hat{i}}$ and $\overline{a}$ as a function of $\hat{\imath}$. Here they cross at a discontinuity of $qb_{\hat{i}}$.

3 A Linear Encoding Scheme over GF[2]

Lemma 2 *There exists a deterministic encoding scheme HB mapping a bundle B of b letters onto an encoding E with cb letters with properties given in Figure 1, namely (i) If a random subset of the letters of the encoding is received, with the number received binomial with mean* $(1+\epsilon)b$*, then the probability of not reconstructing all b letters of the message is at most* $e^{-\Omega(\epsilon^2 b)}$*. (ii) The encoding and the decoding times are* $\mathcal{O}(\epsilon b^2)$*. (iii) The letter size is* $\ell = \mathcal{O}(\ln(1/\epsilon)\epsilon^2 b)$ *bits (or fewer but then the decoding time is* $\mathcal{O}(\epsilon b^2) \times (1+\mathcal{O}(\ln(1/\epsilon)\epsilon^2 b/\ell))$*). (iv) It is systematic. Alternatively, the scheme can be a rateless LT code in that any number of code letters can be produced and each is produced independently of the others.*

This section is organized as follows. First the details of the encoding scheme are given. Then Lemma 2 (Prob) bounds the probability of being able to reconstruct the message. The encoding and decoding algorithms are simple matrix multiplication and Gaussian elimination with some extra care not to do work when there is known to be zeros in the matrix. The pheudo code is given for the decoding and its running time is bounded.

The Encoding Scheme: We now give the encoding scheme for Lemma 2. The scheme will be a *linear* erasure code as are most erasure codes, e.g., Reed-Solomon codes. However, this code will be over GF[2] instead of over some larger finite field. This, in itself, greatly speeds up the computation time, because performing an operation over the finite field GF[2^ℓ] like multiplication takes $\mathcal{O}(\ell^2)$ time for the straightforward algorithm and $\mathcal{O}(\ell\log(\ell))$ with FFT while taking the bitwise XORs of two ℓ bit letters can generally be done in 64 bits parallel in $\mathcal{O}(\frac{\ell}{64})$ time. The message M, consisting of b ℓ bit letters, is viewed as a $(b \times \ell)$ binary matrix and its encoding E as a $(cb \times \ell)$ matrix. The encoding algorithm is simply $V \times M = E$, where V is a fixed $(cb \times b)$ matrix. To decode, let $\widehat{V}$ and $\widehat{E}$ be the rows of the matrices V and E corresponding to those letters of the encoding received. The relation $\widehat{V} \times M = \widehat{E}$ still holds and the message M can be reconstructed as long as the matrix

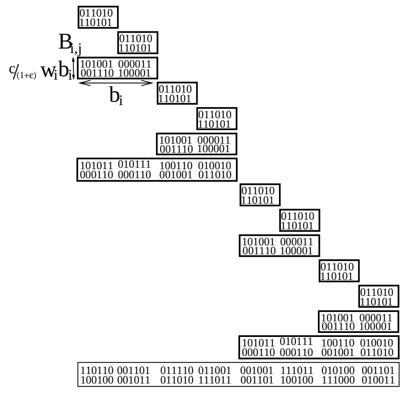

Figure 5: The layout of the encoding matrix V is shown. Each column corresponds to one letter of the message and each row to one letter of the encoding. The matrix V is completely zero outside of the indicated bundle structure. For the fix rate scheme, each rectangle of length b_i is a copy of the same fixed $\left(\left(\frac{w_i}{1+\epsilon}cb_i\right) \times b_i\right)$ binary matrix V_i.

$\widehat{V}$ has rank b. The matrix V will have a special form that allows the system $\widehat{V} \times M = \widehat{E}$ to be solved quickly. The algorithm is simple Gaussian elimination with some extra care not to do work when there is known to be zeros in the matrix.

The matrix V is implicitly defined as follows. See Figure 5. Its parameters are $1 = b_0 < b_1 < \ldots < b_s = b$ and $0 < w_0, w_1, \ldots, w_s \leq 1+\epsilon$, where $\sum_i w_i = 1+\epsilon$. (Recall Lemma 1 sets $s = \ln(1/\epsilon^2)$, has b_i double from $b_1 = 2\epsilon^2 b$ to $b_s = b$, $w_1 \approx 0.29$, $w_i \approx 0.41 \cdot 2^{-i/2}$, and $w_s \approx 2.41\epsilon$.) The message is broken into a hierarchy of bundles. The message M itself is a single bundle of size $b = b_s$. This bundle is broken into smaller sub-bundles of size b_{s-1}, which are broken further into sub-sub-bundles of size b_{s-2} and so on. Let $B_{\langle i,j\rangle} \subseteq [1..b]$ denote the index set of message letters in the j^{th} bundle of size b_i. Let $M_{\langle i,j\rangle}$ denote the portion of the message M indexed by bin $\langle i,j\rangle$. If the scheme is to be a rateless LT codes [10], then the probability of a packet being about $M_{\langle i,j\rangle}$ is $\frac{w_i}{1+\epsilon}\frac{b_i}{b}$, while if the scheme is to have rate c, then there will be $\frac{w_i}{1+\epsilon}cb_i$ such packets. Note that the total probability is $\sum_i \left(\frac{w_i}{1+\epsilon}\frac{b_i}{b}\right)\left(\frac{b}{b_i}\right) = 1$ and the total number of letters of the encoding is $\sum_i \left(\frac{w_i}{1+\epsilon} \times cb_i\right)\left(\frac{b}{b_i}\right) = cb$. Either way let $E_{\langle i,j\rangle}$ denote these encoded letters. Each such letter will simply be the GF$[2^\ell]$ dot product of some b_i bit vector with the message letters $M_{\langle i,j\rangle}$, i.e. bit-wise XORs of the ℓ bit letters. Let $V_{\langle i,j\rangle}$ denote the matrix with rows being these vectors. Then the letters of the encoding $E_{\langle i,j\rangle}$ "about" the bundle $B_{\langle i,j\rangle}$ is given by $V_{\langle i,j\rangle} \times M_{\langle i,j\rangle} = E_{\langle i,j\rangle}$. For LT codes, independently for each packet, the encoder randomly chooses this binary vector of length b_i and includes it in the header of the packet. For a fixed rate scheme, a fixed $\left(\left(\frac{w_i}{1+\epsilon}cb_i\right) \times b_i\right)$ binary matrix $V_i = V_{\langle i,j\rangle}$ is specified for each bin/bundle size $i \in [1..s]$. (It is sufficient to choose these matrices randomly and then fix them.)

To make the scheme *systematic*, the b letters of the message are contained unchanged in b of the code letters, by having the smallest bundle size be $b_0 = 1$, where the number of them packets about each be 1. (This change only increases the probability of success and decreases the cost. There being only one code letter, the bundle/bin cannot overflow.)

Probability: We next bound the probability of being able to reconstruct the message.

Proof of Lemma 2(Prob): To simplify the analysis of the probability of success, we will initially not use fixed matrices V_i. Instead, each letter of the encoding will be the linear combination of a subset of the letters from the appropriate bundle where each letter of the bundle is included with probability $\frac{1}{2}$ independently at random.

The hierarchy bin game ensures that w.h.p. no more than b_i code symbols are about a bundle of size b_i or its subbundles (except the full bundle).[4] Now, in addition to these information theoretic requirements, we require that the equations defining these code letters are linearly independent. To help this, tighten the requirement of the game from needing $c_{\langle i,j\rangle} \leq b_i$ to $c_{\langle i,j\rangle} \leq (1-q_i^2/2)b_i$ encoding letters to *contribute* to it. When randomly choosing the equation for the k^{th} such letter it is chosen from a space of dimension b_i. The previous $k-1$ equations span a sub-space of dimension at most $k-1$. Hence, the probability that the k^{th} is within this sub-space is at most $2^{-(b_i-k+1)}$. The probability that the $(1-q_i^2/2)b_i$ equations are dependent is at most $\sum_{k\in[1..(1-q_i^2/2)b_i]} 2^{-(b_i-k+1)}$ $\leq e^{-\frac{q_i^2 b_i}{2}}$, which is the same as the probability we had before. Hence we can conclude that with high probability all the letters that are about bundles of size smaller than b are linearly independent.

What remains is to consider the letters of the encoding that are about the bundle of size b. By the statement of the lemma, the number of letters of the encoding received is binomial$(cb, \frac{1+\epsilon}{c})$ with mean $(1+\epsilon)b$. Therefore, the probability that at least $(1+2\epsilon^2)b$ letters are received is at least $1 - \mathrm{e}^{-\Omega(\epsilon^2 b)}$. With the same argument given in the second paragraph of the proof of this lemma, the first $(1-\epsilon^2)b$ of these are linearly independent with probability at least $1 - \mathrm{e}^{-\Omega(\epsilon^2 b)}$. Now consider one of the remaining $3\epsilon^2 b$ 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 $3\epsilon^2 b$ equations can be thought of as $3\epsilon^2 b$ Bernoulli trials. The matrix has full rank if at least $\epsilon^2 b$ of the trials succeed. The expected number of successes is $1.5\epsilon^2 b$. The probability of getting fewer than $\epsilon^2 b$ is at most $\mathrm{e}^{-\Omega(\epsilon^2 b)}$.

The remaining step is to prove that it is sufficient to use a fixed matrix V_i for each size of bundle. Randomly choosing the same V_i for each bundle of size b_i adds dependence between the events considered but does not change their probability. Because we always use the sum of probabilities of bad events, the proof does not assume independence. Finally, if the overall probability of failure is $\mathrm{e}^{-\Omega(\epsilon^2 b)}$ when the V_i are chosen randomly, then there exists fixed ones that leads to a probability that is at least as good. (On the other hand, proving that you have such a matrix may be hard.) ■

Encoding and Decoding Algorithm: The encoding is done via binary matrix multiplication $V \times M = E$. The obvious savings in time comes from never looking at or even storing the entries of $\widehat{V}$ outside of the bundle structure. The time to produce one of the $\left(\frac{w_i}{1+\epsilon}cb_i\right)(b/b_i) = \mathcal{O}(w_i b)$ code letters about a bundle of size b_i is b_i, where XORing two ℓ bit letters is considered to be a single operation. Hence, the total time is cost $\sum_i \mathcal{O}(w_i b)b_i = \mathcal{O}(\epsilon b^2)$ of the hierarchical bin game from Lemma 1. 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. The difficulty is that even with the hierarchical bundle structure of our matrix $\widehat{V}$, its inverse is not sparse. In general this means that solving a $(b \times b)$ system with $\mathcal{O}(b)$ non-zero entries requires $\mathcal{O}(b^3)$ bit operations. However, in our

[4]Having the subbundles not overflow was for the benefit of the proof. If $(1+2\epsilon)b$ were received instead of $(1+\epsilon)b$ then enough packets would be received to be able to decode any bundle just from the bounds about it and its subbundles.

case, we are able to maintain the bundle structure of the matrix as we use Gaussian elimination to zero the bottom triangle of $\widehat{V}$. Once the matrix is upper triangular, the system can be solved quickly.

The main sub-task during Gaussian elimination of $\widehat{V}$ is that of taking the r^{th} row, which has a one on the "diagonal", and adding it to every row r' below it that contains a one in the corresponding column. In a general sparse matrix, the non-zero entries of these rows do not necessarily fall in the same places. Hence, the r'^{th} row will gain most of the non-zero entries of the r^{th} row. The effect is that the number grows exponentially with the number of row operations. This, however, does not happen here. We will assume that the rows are sorted in what we will call the *hierarchical partial order.* As each such $B_{\langle i_r,j_r\rangle}$ is either disjoint from or contained in a latter $B_{\langle i_{r'},j_{r'}\rangle}$. See Figure 5. In the first case, adding the r^{th} and the r'^{th} row would not cancel any entries, hence these rows are never added together. In the second case, adding the r^{th} row to the r'^{th} will change which entries in the bundle $B_{\langle i_{r'},j_{r'}\rangle}$ are one, but will not contribute ones outside of the bundle. Hence, as the matrix $\widehat{V}$ is zeroed below the diagonal, the bundle hierarchical structure is maintained.

Once $1..r{-}1$ columns have been zeroed below the diagonal, the next step is to do the same for an r^{th} column. If the $\langle r,r\rangle$ diagonal entry is zero, the standard thing to do is to swap the r^{th} row with a lower row that does have a non-zero in the r^{th} column. The problem with doing this is that it may mess up the required hierarchical partial order. Hence, we will instead imagine swapping the r^{th} column with a column to the right that does have a non-zero in the r^{th} row. Instead of actually swapping, we will say that matrix $\widehat{V}$ is *column permuted lower triangular* if for each row r, either the row is stated to be redundant, or associated with it is a column $c \equiv Diagonal(r)$ such that this entry $\widehat{V}_{\langle r,c\rangle}$ is one and the column part below it, namely $\widehat{V}_{\langle r',c\rangle}$ for $r' > r$, has been zeroed. If the matrix has full rank b, then each column $c \in [b]$ is the diagonal of some row r.

$$\begin{array}{cccccccc}
1 & * & * & * & & & & \\
0 & 1 & * & * & & & & \\
0 & 0 & * & 1 & & & & \\
 & & & & 1 & * & * & * \\
 & & & & 0 & * & 1 & * \\
0 & 0 & 1 & 0 & 0 & * & 0 & * \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & * \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

Figure 6: A column permuted lower triangular matrix is shown. The '1' are along the "diagonal", the '0' are below the diagonal, and the '*' representing an arbitrary character from $\{0,1\}$ are above. The blanks are outside the bundle structure and are zeros and hence either a '0' or a '*'. Note how there is exactly one '1' in each row and in each column. Note also that each column transitions from '*' to '1' to '0'. $Diagonal(3) = 4$. The third row has a pointer to the sixth.

In order to save time when zeroing column $c = Diagonal(r)$ using the r^{th} row, only rows r' for which $B_{\langle i_r,j_r\rangle} \subseteq B_{\langle i_{r'},j_{r'}\rangle}$ need to be considered. This can be done by associating with each row r, a pointer to the next row $next(r)$ for which $B_{\langle i_r,j_r\rangle} \subseteq B_{\langle i_{next(r)},j_{next(r)}\rangle}$. A property of the partial order is that following this linked list of pointers starting at any row r will reach every row r' for which $B_{\langle i_r,j_r\rangle} \subseteq B_{\langle i_{r'},j_{r'}\rangle}$. (The same would not be true in the reverse order.)

To save even more time, the time spent adding the r^{th} row to any one r'^{th} row should be the number of ones in the r^{th} row and not the size b_{i_r} of its bundle. There are two ways of achieving this. Either at the beginning of this sub-task a succinct list of the ones of the r^{th} row can be made or the bundle $B_{\langle i_r, j_r\rangle}$ in the r^{th} row can scanned and each time a one is found the above linked list giving all rows r' for which $B_{\langle i_r, j_r\rangle} \subseteq B_{\langle i_{r'}, j_{r'}\rangle}$ could be followed. The code below does the later.

Pseudo Code: The following is pheudo code for solving $\widehat{V} \times M = \widehat{E}$ for M.

```
algorithm $Decode(\widehat{V}, \widehat{E})$

$\langle$pre−cond$\rangle$: Matrix $\widehat{V}$ and encoding $\widehat{E}$ are the rows of $V$ and $E$ corresponding to those letters
    of the encoding received. Matrix $\widehat{V}$ retains $V$'s hierarchical bundle structure. See Figure 5.
    Row $r$ is specified by giving its bundle $B_{\langle i_r, j_r\rangle} \subseteq [b]$ outside of which the row is zero and inside
    of which is given by the row $\rho_r$ of matrix $V_{i_r}$. The corresponding code word in $\widehat{E}_r$ is given by
    the linear combination $[V_{i_r}]_{\rho_r} \times M_{\langle i_r, j_r\rangle}$.

$\langle$post−cond$\rangle$: The message $M$ for which $\widehat{V} \times M = \widehat{E}$ is returned. This is a single message iff $\widehat{V}$
    has full rank $b$, otherwise all $2^{b-rank(\widehat{V})}$ such solutions $M$ are returned.

begin
    – Because the contents of the packets have been randomly permuted, we begin by sorting
      the rows of $\widehat{V}$ according to the hierarchical partial order, so that if $r < r'$ then either
      $B_{\langle i_r, j_r\rangle} \subseteq B_{\langle i_{r'}, j_{r'}\rangle}$ or these bundles are disjoint. Figure 5 lays out one such ordering. While
      sorting, store for each row $r$ of $\widehat{V}$, the next row $next(r)$ such that $B_{\langle i_r, j_r\rangle} \subseteq B_{\langle i_{next(r)}, j_{next(r)}\rangle}$.

    % $\widehat{V}$ and $\widehat{E}$ are transformed so that $\widehat{V}$ becomes a column permuted lower triangular matrix.
    loop $r = 1 \ldots |\widehat{E}|$ (i.e. number of rows = number of packets received = $(1+\epsilon)b$)
        $\langle$loop−invariant$\rangle$: The matrix $\widehat{V}$ and the vector $\widehat{E}$ have been transformed handling
            the first $r$–1 rows, i.e. for each row $r' < r$, either the row is stated to be redundant,
            or associated with it is a column $c' = Diagonal(r')$ such that this entry $\widehat{V}_{\langle r', c'\rangle}$ is
            one and the column below it, namely $\widehat{V}_{\langle r'', c'\rangle}$ for $r'' > r'$, has been zeroed. Moreover
            the set of $M$ for which $\widehat{V} \times M = \widehat{E}$ remains the same; $\widehat{V}$ retains $V$'s hierarchical
            bundle structure; and its rank remains same.

        % Establish the LI for the $r^{th}$ row.
        if( the $r^{th}$ row of $\widehat{V}$ is all zeros ) then
            if( $\widehat{E}_r \neq \vec{0}$ ) then return( "System is inconsistent. No solutions $M$" )
            else the $r^{th}$ row is redundant.
        else
            Let $c = Diagonal(r)$ be an index of a one in the $r^{th}$ row of $\widehat{V}$.
            % Zero the column below $\widehat{V}_{\langle r, c\rangle}$ by adding the $r^{th}$ row of $\widehat{V}$ and $\widehat{E}$ to every
              $r'^{th}$ row for which $\widehat{V}_{\langle r', c\rangle}$ is one.
            % Start with $\widehat{E}$. Follow the linked list $next(r')$ reaching
              every row $r' > r$ for which $B_{\langle i_r, j_r\rangle} \subseteq B_{\langle i_{r'}, j_{r'}\rangle}$.
            for( $r' = r$; $r' = next(r')$; $r' > |\widehat{E}|$ )
                if( $\widehat{V}_{\langle r', c\rangle} = 1$)
                    $toBeSummed(r') = yes$.
                    $\widehat{E}_{\langle r', c'\rangle} = \widehat{E}_{\langle r', c'\rangle} + \widehat{E}_{\langle r, c'\rangle}$ over GF[$2^{\ell}$].
                else
                    $toBeSummed(r') = no$.
                end if
            end loop
            % To save time, we do the adding one column at a time.
            for each column $c' \in B_{\langle i_r, j_r \rangle}$
                if ( $\widehat{V}_{\langle r,c' \rangle} = 1$) then
                    % add this one in row $r$ where needed in this column.
                    for( $r' = r$; $r' = next(r')$; $r' > |\widehat{E}|$ )
                        if( $toBeSummed(r') = yes$ )
                            $\widehat{V}_{\langle r',c' \rangle} = \widehat{V}_{\langle r',c' \rangle} + 1 \bmod 2$.
                        end if
                    end loop
                end if
            end for
        end if
    end loop

    % We now solve the column permuted lower triangular system $\widehat{V} \times M = \widehat{E}$.
    – If the matrix $\widehat{V}$ started with full rank $b$, then each column $c \in [b]$ is $c = Diagonal(r)$
      for some row $r$. Otherwise there are $b - rank(\widehat{V})$ columns $c$ with no such $r$. Give the
      corresponding variables $M_{c_k}$ the undefined value $x_k$. The $2^{b-rank(\widehat{V})}$ values of these indicate
      the $2^{b-rank(\widehat{V})}$ possible solutions.
    loop $r = |\widehat{E}| \ldots 1$
        ⟨loop−invariant⟩: For every column $c$, the corresponding value $M_c$ is known (possi-
            bly as a function of the $x_k$), except for those $c'$ for which there exists a $r' \leq r$ such
            that $c' = Diagonal(r')$.

        $c = Diagonal(r)$
        Solve the $r^{th}$ equation $\widehat{V}_r \times M = \widehat{E}_r$ giving the value of $M_c$ as a function of the
        already known values $M_{c'}$.
            % Note that this can be done because $\widehat{V}$ is a column permuted lower triangular
              matrix. Hence, $\widehat{V}_{\langle r,c \rangle}$ is one. Also by the loop invariant, for any other unknown
              value $M_{c'}$, there exists a $r' < r$ such that $c' = Diagonal(r')$. Hence, the column
              $\widehat{V}_{\langle r'',c' \rangle}$ for $r'' > r'$, has been zeroed. Specifically $\widehat{V}_{\langle r,c' \rangle}$ is zero.
    end loop
    return( $M$ )
end algorithm
```

Running Time: Now consider the computation time of the above algorithm. The number of letter operations within the matrix $\widehat{E}$ (i.e., XORing two ℓ bit letters together) when zeroing below the diagonal is the number of row operations in $\widehat{V}$. The r^{th} row is added to the r'^{th} at most once and only if the bin for r is poised over that for r', i.e. $B_{\langle i_r, j_r \rangle} \subseteq B_{\langle i_{r'}, j_{r'} \rangle}$. Recall that we ensured that the total number of packets that arrive *about* a bundle or a sub bundle is at most its size $b_{i_{r'}}$. Hence, the total number of such operations is at most $\sum_{r' \in [1..(1+\epsilon)b]} b_{i_{r'}}$. The number of rows r' of size $b_{i_{r'}}$ is $w_i b_i \cdot \frac{b}{b_i} = w_i b$. This bounds number of operations by $\sum_{i \in [1..s]} \mathcal{O}(w_i b) b_i$. This was defined to be the *cost* of the game which Lemma 1 proves is $\mathcal{O}(\epsilon b^2)$. When finding M, the number of such letter operations is the number of one's that are above the diagonal of the upper triangular matrix, but as said, the bundle structure does not change when zeroing below the diagonal. Hence, the number is this same $\mathcal{O}(\epsilon b^2)$.

Now let us bound the number of bit operations within $\widehat{V}$ in order to add some r^{th} row to some r'^{th} row. The expected number of rows r whose bundle $B_{\langle i_r, j_r \rangle}$ is of size b_i is $(w_i b_i)(b/b_i) = w_i b$. For each of these, the code loops over the number of rows r' for which $B_{\langle i_r, j_r \rangle} \subseteq B_{\langle i_{r'}, j_{r'} \rangle}$, namely the geometric sum $\sum_{i' \in [i..s]} w_{i'} b_{i'} = \sum_{i' \in [i..s-1]} \mathcal{O}((\epsilon\sqrt{b/b_{i'}}) b_{i'}) = \mathcal{O}(\epsilon b)$. Recall that in order to save time, we ensure that the time spent adding the r^{th} row to any one r'^{th} row should be the number of ones in the r^{th} row and not the size b_{i_r} of its bundle. The expected number of these is b_i minus the number of entries that have already been zeroed. The number zeroed is the number of rows r' before it for which $B_{\langle i_{r'}, j_{r'} \rangle} \subseteq B_{\langle i_r, j_r \rangle}$, namely $\sum_{i' \in [1..i]} (w_{i'} b_{i'})(b_i/b_{i'}) = (1-q_i) b_i$. Hence, the expected number of ones remaining is $q_i b_i$. We can conclude that the expected total number of bit operations in the Gaussian elimination is $\sum_{i \in [1..s]} w_i b \times \mathcal{O}(\epsilon b) \times q_i b_i = \sum_{i \in [1..s]} \mathcal{O}(\epsilon\sqrt{b/b_i} b) \times \mathcal{O}(\epsilon b) \times \mathcal{O}(\epsilon\sqrt{b/b_i} b_i)$ $= \mathcal{O}(s\epsilon^3 b^3) = \mathcal{O}(\ln(1/\epsilon^2)\epsilon^3 b^3)$.

The computation times stated in Lemma 2 are in terms of letter operations. This is reasonable given the input size is stated in terms of the number of letters. However, the time $\mathcal{O}(\ln(1/\epsilon^2)\epsilon^3 b^3)$ to perform Gaussian elimination on $\widehat{V}$ is measured in bit operations. A letter operation requires ℓ bit operations. Hence, it is reasonable to allow the algorithm to do ℓ bit operations within $\widehat{V}$ for every letter operation. Using this measure, the number of "letter" operations required to decode the message is $\mathcal{O}(\epsilon b^2) + \frac{1}{\ell}\mathcal{O}(\ln(1/\epsilon^2)\epsilon^3 b^3)$. For the algorithm, the letter size ℓ could be a single bit. The reason for having it larger is to amortize the number of bit operations required to invert $\widehat{V}$. If we set ℓ to be $\mathcal{O}(\ln(1/\epsilon)\epsilon^2 b)$, then the number of letter operations to decode is $\mathcal{O}(\epsilon b^2)$. In order to emphasize this, we expressed in Figure 1 the total decoding time as being $\mathcal{O}(\epsilon b^2) \times \left(1+\tilde{\mathcal{O}}(\epsilon^2 b/\ell)\right)$.

4 One Level of Bundles and Two Phases of Expansion

Review the table in Figure 1.

Scheme	MDS	HB	HB$'$	HB$''$	Raptor	Expanders
Message Leng	b		n			
Code Length	cb		cn			$(1+\epsilon)n$
Code Needed	b	$(1+\epsilon)b$	$(1+\epsilon)n$			$(1-\beta)(1+\epsilon)n$
Mess. Acquired	all		$(1-\beta)n$	all		
Failure Prob	zero	$e^{-\Omega(\epsilon^2 b)}$	$e^{-\Omega(\beta\epsilon^2 n)}$	$e^{-\Omega(\epsilon^4 n)}$	$n^{-\Omega(1)}$	zero
Time	$\mathcal{O}(b^2)$	$\mathcal{O}(\epsilon b^2)\times \left(1+\tilde{\mathcal{O}}(\epsilon^2 b/\ell)\right)$	$\mathcal{O}(\ln(\beta^{-1})\epsilon^{-1} n)\times \left(1+\mathcal{O}(\ln(\beta^{-1})\ln(1/\epsilon)/\ell)\right)$	$\tilde{\mathcal{O}}(\epsilon^{-1} n)\times \left(1+\mathcal{O}(\ln^2(1/\epsilon)/\ell)\right)$	$\tilde{\mathcal{O}}(n)$	$\mathcal{O}(\epsilon^{-1} n)$
Packet Size ℓ	1 f.e.	1				1 f.e.

Lemma 3 *The scheme* HB$'$ *in Figure 1 is constructed by applying the scheme* HB *to each of the* $\frac{n}{b}$ *bundles of size* b.

Proof of Lemma 3: The scheme *HB$'$* with parameters $\langle n, c, \epsilon, \beta \rangle$ breaks its n letter message into $\frac{n}{b}$ bundles of size $b = \ln(\beta^{-1})\epsilon^{-2}$ and applies the scheme *HB* with parameters $\langle b, c, \epsilon' = \epsilon/2 \rangle$

to each producing $\frac{n}{b} \times cb = cn$ single letter packets. The number of letter operations is $\frac{n}{b} \times \mathcal{O}(\epsilon' b^2) = \epsilon bn = \mathcal{O}(\ln(\beta^{-1})\epsilon^{-1}n)$. The number of bit operations to invert $\widehat{V}$ is $\frac{n}{b} \times \mathcal{O}(\ln(1/\epsilon)\epsilon^3 b^3) = \mathcal{O}(\ln^2(\beta^{-1})\epsilon^{-1}\ln(1/\epsilon)n)$. Hence setting $\ell = \mathcal{O}(\ln(\beta^{-1})\ln(1/\epsilon))$ amortizes this time. We then receive a random subset of $(1+\epsilon)n$ of these code letters. A message bundle is w.h.p. recovered if $(1+\epsilon/2)b$ of the cb letters about it are received. It is sufficient to prove that this occurs for at least a $1-\beta$ fraction of the $\frac{n}{b}$ bundles with failure probability at most $\mathrm{e}^{-\Omega(\beta\epsilon^2 n)}$.

To simplify things, first assume that each of the cn code letters is received independently with probability $\frac{(1+\epsilon)}{c}$. Standard Chernoff bounds give the probability of not receiving at least $(1+\epsilon/2)b$ letters is at most $\mathrm{e}^{-\Omega(\epsilon^2 b)}$. Denote this probability by p. We then use Chernoff bounds a second time to bound the probability of failing to reconstruct at least a $1-\beta$ fraction of the n/b bundles. The Chernoff bound states that if Y is the sum of mutually independent Bernoulli variables where $\mathrm{Exp}(Y) = \mu$, then $\Pr[Y > q\mu] \le \mathrm{e}^{-q\ln(q/\mathrm{e})\mu}$. Here the probability that a bundle is lost is $p = \mathrm{e}^{-\Omega(\epsilon^2 b)}$ and the expected number lost is $\mu = p\frac{n}{b}$. The number of bundles that we can afford to lose is $\beta\frac{n}{b}$ and setting this to $q\mu$ gives $q = \frac{\beta}{p}$. Thus, the failure probability is at most $\mathrm{e}^{-q\ln(q/\mathrm{e})\mu} = \mathrm{e}^{-\Omega(\beta(n/b)[\epsilon^2 b - \ln(1/\beta)]))} = \mathrm{e}^{-\Omega(\beta\epsilon^2 n)}$ as desired.

We now compare the effect of having changed the distribution. The original distribution is the same as the assumed one with the added constraint that the total number of letters received K is $(1+\epsilon)n$. Decreasing K only increases the probability of failure E. Therefore, $\Pr_{old}[E] \le \Pr_{new}[E \mid K \le (1+\epsilon)n] \le \Pr_{new}[E]/\Pr_{new}[K \le (1+\epsilon)n] \le \Pr_{new}[E] \times \mathrm{e} = \mathrm{e}^{-\Omega(\beta\epsilon^2 n)}$. ■

Lemma 4 *The scheme* HB$''$ *described in Figure 1 is constructed from the schemes* Expanders *and* HB$'$.

Proof of Lemma 4: Suppose we are constructing the scheme HB'' with parameters $\langle n, c, \epsilon\rangle$. The first phase uses the scheme *Expanders* with parameters $\langle n, \beta = \epsilon^2, \epsilon' = \epsilon/3\rangle$ to encode the n letters M into $(1+\epsilon/3)n$ letters M'. The second phase uses the scheme HB' with parameters $\left\langle n' = (1+\epsilon/3)n, c' = c/(1+\epsilon/3), \beta = \epsilon^2, \epsilon'' = \frac{2}{3}\epsilon/(1+\epsilon/3)\right\rangle$ to encode the $(1+\epsilon/3)n$ letters M' into $c'n' = cn$ letters. These are randomly permuted before putting one into each packet. From any $(1+\epsilon'')n' = (1+\epsilon)n$ packets one gets a random subset of the code letters, sufficient to reconstruct with high probability some $(1-\beta)$ fraction of M' and from this the message M can be reconstructed. ■

Note that the first stage can be skipped by setting $\beta < 1/n$. But then the encoding time would be $\mathcal{O}\left(\frac{\ln(n)}{\epsilon}n\right)$.

5 Concluding remarks and open problems

The open problem is to obtain an even faster probabilistic or deterministic erasure code. An interesting aside about the required packet size is that for any deterministic erasure code with the parameters in Lemma 4 (without any assumption on the efficiency of its encoding and decoding procedures), the minimum possible packet size is at least $\Omega(\ln((c-1)/\epsilon))$ for all $\epsilon \ge 1/n$. This can be proved using the Plotkin bound. This scheme demonstrates that in probabilistic schemes the packet size can be as small as one bit.

References

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

[2] A. Albanese, J. Blömer, J. Edmonds, M. Luby, M. Sudan, "Priority Encoding Transmission", *Proceedings of* 35th *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, J. Edmonds, M. Luby, "Linear Time Erasure Codes with Nearly Optimal Recovery," 36th *FOCS*, pp. 512-519, 1995.

[6] N. Alon, M. Luby, "Linear Time Erasure Codes with Nearly Optimal Recovery," submitted for journal publication.

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

[8] 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.

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

[10] M. Luby, "LT Codes", *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271. 280, November 2002.

[11] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery". Internet Engineering Task Force (IETF), ISSN: 2070-1721, Aug 2011.

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

[13] A. Shokrollahi, "Raptor Codes", *IEEE Transactions on Information Theory*, vol. 52, no. 6, June 2006.

[14] A. Shokrollahi and M. Luby, "Raptor Codes. Foundations and Trends in Communications and Information Theory", *Foundations and Trends in Communications and Information Theory*, 6(3-4): 213-322, 2009.

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

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