Limited Randomness LT Codes

Chris Harrelson* Lawrence Ip† Wei Wang‡

{chrishtr,lip,wangwei}@eecs.berkeley.edu

Abstract

LT codes are asymptotically optimal rateless erasure codes with highly efficient encoding and decoding algorithms. In the original analysis of these codes, it was assumed that for each encoding symbol, the neighbors used to generate that encoding symbol are chosen uniformly at random.

Practical implementations of LT codes cannot afford this amount of randomness, because all random bits must be communicated to the decoding party. Instead, they use a linear congruential generator to reduce the randomness used per encoding symbol to a seed consisting of two random numbers. We show that such *limited randomness* LT codes perform almost as well as the fully random version. Thus even limited randomness LT codes are asymptotically optimal.

1 Introduction

When transmitting data over an IP network, recovery from dropped packets is usually achieved by using TCP, where the receiver sends acknowledgements or requests for retransmission for each packet. Such feedback-based reliability schemes do not scale well when transmitting over high latency or high loss networks [5]. These issues are of particular importance in one-to-many (multicast) and many-to-one data delivery, where to achieve scalability, we must limit the amount of feedback to the senders and the number of redundant packets sent to receivers. Erasure codes rely on the principle that with redundant encoding, lost packets can be recovered at the receiver without the need for retransmission. In what has become known as the *digital fountain approach*, efficient erasure codes have been proposed as a mechanism to achieve reliability without feedback and with minimal redundancy, so that the receiver can recover k data symbols by collecting any subset of $(1+\epsilon)k$ from a stream of encoding symbols [1, 2]. Examples of applications in use today are Transporter Fountain, the flagship product of Digital Fountain, Inc. [3], and multicast protocols such as WebRC [10].

Traditional erasure codes such as Reed-Solomon codes and Tornado codes [11] are typically block based with a fixed rate, that is n encoding symbols consisting of k input symbols and $n-k$ redundant symbols, with n determined a priori, for a rate of k/n. This rate must be estimated in advance, and significant overhead is incurred when the rate is too high or too low.

LT codes eliminate the need to obtain an accurate estimate of the rate [8]. They are rateless erasure codes, in the sense that a potentially unlimited number of of encoding symbols can be generated on the fly. As long as enough encoding symbols are received, regardless of which

*CS Division, UC Berkeley. Supported in part by a GAANN fellowship and NSF grant CCR-0105533.

†CS Division, UC Berkeley. Supported by DARPA Grant F30602-01-2-2054.

‡EECS Department, UC Berkeley. Supported by an NSF Graduate Fellowship and a Lucent GRPW Grant.

symbols are received, the receiver will be able to recover the original data with high probability. Thus only as many encoding symbols as required need be generated. We note that two other rateless codes have recently been designed which may have advantages over LT codes in practice [12, 13].

The original analysis of LT codes showed that k input symbols can be recovered with probability $1-\delta$ from a set of $k+O(\sqrt{k}\log^2 k/\delta)$ fully randomly generated encoding symbols. The analysis assumed that the input symbol neighbors of each encoding symbol are chosen uniformly at random.

In practice, implementations of LT codes use a linear congruential generator to reduce the amount of randomness used to generate the neighbors of each encoding symbol (from $O(k\log k)$ bits[1] per encoding symbol to $O(\log k)$ bits per symbol) [6, 7]. Since these bits need to be sent to the decoder, this dramatically reduces the transmission overhead. Although simulations have shown that the redundancy of the encoding process with a linear congruential randomness generator is virtually indistinguishable from that of the encoding process with full independence [9], there have been no analytical results on the performance of these limited randomness LT codes.

Our contribution In this paper we prove analytically that with the linear congruential generator encoding process, $k+O(k^{5/6}\text{polylog}(k,1/\delta))$ encoding symbols are sufficient to recover k input symbols with probability $1-\delta$. The number of symbol operations to encode and decode remains unchanged, $O(k\log k/\delta)$. This result provides a theoretical justification for the use of LT codes with only limited randomness.

2 Description of LT Codes

2.1 Erasure channel model

LT codes are designed to work over an erasure channel. In a typical application the length ℓ of input and encoding symbols is chosen to be just less than the size of the packet payload (we need to reserve some room to transmit overhead information like the random bits used to generate each encoding symbol). If N is the input data length, the number of input symbols is $\lceil k = N/\ell\rceil$.

We assume that packets may be lost or reordered in transit but not corrupted. In addition, we assume that the losses and reorderings are *oblivious* to the random bits used in the encoding algorithm. (We assume that corrupted packets are handled by another network layer, which performs for example appropriate CRC checks.)

2.2 Encoding process

Full-randomness version In the *original* construction of LT codes, each encoding symbol is generated independently of all other encoding symbols, by the following process:

1. Randomly choose the *degree* d of the encoding symbol according to a degree distribution (see Section 2.6).

2. Choose a random set of d distinct input symbols as *neighbors* of the encoding symbol.

3. Set the *value* of the encoding symbol to be the XOR of the values of the d neighbors.

[1]This is a worst case bound; on average $O(\log k\log k/\delta)$ bits are required.

One can view this process as the construction of a bipartite graph. We have k vertices on one side of the graph, each of which represents one input symbol. On the other side we have one vertex for each encoding symbol. There is an edge from each encoding symbol to its d input symbol neighbors.

Each encoding symbol is then put into a *packet* that contains its value, its degree and its neighbor list. These packets are then sent to the decoder. Note that each packet contains $\lceil d \log k \rceil$ bits for the neighbor list.

Limited randomness version In practice, $\lceil d \log k \rceil$ bits per packet is an unreasonably large number (as the average degree of an encoding symbol turns out to be $\Omega(\log k/\delta)$). Instead, the d neighbors of each encoding symbol are chosen as follows:

1. Choose two integers $a \in \{1, \ldots, k-1\}$ and $b \in \{0, \ldots, k-1\}$ uniformly at random.

2. The ith neighbor of the encoding symbol is then the $(ai + b \bmod k)$th input symbol.

Here we have assumed without loss of generality that k is prime, so that we cover all input symbols evenly. This allows us to send only $\lceil 2 \log k \rceil$ bits per symbol for neighbor information, since the decoder can reconstruct the neighbors easily given the two random seeds a and b.

2.3 Decoding process

The decoding process is the same for limited randomness LT codes as in the original exposition. The decoder first receives K encoding symbols and the bipartite graph, from which it will try to *recover* input symbols, the neighbors of the encoding symbols in the graph. Let the set of received encoding symbols which have not been fully processed be C (initially C contains all K packets). We repeatedly choose an encoding symbol c from C and do one of the following, until none of them can be done, or all input symbols are recovered:

1. If c no longer has any unrecovered neighbors, it is useless to us, so we remove it from C.

2. If c is adjacent to exactly one input symbol we say that that input symbol is *covered* by c. We then *recover* that input symbol, since its value is the same as the value of c. Now c has finished being processed, and so we remove it from C.

3. Otherwise consider each neighbor of c in turn. If it has already been recovered, then XOR its value into the value of the c and remove the edge connecting that neighbor to c. Since c still has unrecovered neighbors, keep c in C.

We say that the decoding process *succeeds* if it recovers all of the input symbols from at most K encoding packets. Otherwise, we say that it *fails*.

2.4 Released encoding symbols and the ripple

At the point when an encoding symbol has exactly one remaining unrecovered input symbol neighbor, we say that the encoding symbol is *released*. If an encoding symbol is ever discarded in step 1 of the decoding process, we say that it is *wasted*. The set of wasted encoding symbols is exactly the set which does not end up helping us recover any input symbols. In particular, if the decoding process is successful there will be exactly $K - k$ wasted encoding symbols.

The set of as-of-yet unrecovered input symbols which are covered by a released encoding symbol is called the *ripple*. Note that when an encoding symbol releases, it may or may not increase the size of the ripple because the input symbol it covers may already be in the ripple.

The decoding process processes input symbols in the ripple one-by-one. It is able to continue if and only if the ripple has nonzero size. If we can ensure that the ripple always has nonzero size with high probability, then we have shown that the decoding process has succeeded with high probability.

2.5 General notation

Let k be the number of input symbols; we assume for simplicity and without loss of generality that k is prime. Let $K = \beta k$ be the number of encoding symbols required to decode with probability $1-\delta$ under the assumption of completely random neighbors. We will require a slightly larger number of symbols in our analysis (see Lemma 5).

At each iteration of the decoding process, let L be the number of unrecovered input symbols (note that as the decoding process proceeds, L *decreases* in value) and S be the set of recovered input symbols.

Let $\alpha = \max(20, 2(8-(\log\delta/6))/\log k)$, $f = c_f k^{-1/6}(\log k/\delta)^{1/2}(\log k)^{1/2}\alpha^{1/6}$, $\lambda = \frac{1}{6}\log k$, and $R = c_R k^{5/6}(\log k)^{1/2}(\log k/\delta)^{-1/2}\alpha^{1/6}$. Their role in the analysis will be explained later.

2.6 Degree Distribution

Choosing a good degree distribution is the key to making LT codes work. The distribution determines the number K of encoding symbols needed to ensure that the ripple always has nonzero size with high probability (see Section 2.4). A large ripple size increases the probability of successful decoding but also increases the number of wasted encoding symbols. So we need a distribution that keeps the ripple size just large enough for successful decoding. One such degree distribution is the Robust Soliton distribution:

Definition 1 *(Robust Soliton distribution [8])*[2] *Let*

$$\rho(i) = \begin{cases} 1/k & : \quad i = 1 \\ 1/i(i-1) & : \quad i = 2, \dots, k \end{cases}$$

$$\tau(i) = \begin{cases} R/(ik) & : \quad i = 1, \dots, k/R - 1 \\ 2e^2(R\log\frac{R}{\delta})/k & : \quad i = k/R \\ 0 & : \quad i = k/R + 1, \dots, k \end{cases}$$

$$\beta = \sum_{i=1}^{k} (\rho(i) + \tau(i))$$

The Robust Soliton distribution $\mu(i)$ for $i = 1, \dots, k$ is $\mu(i) = (\rho(i) + \tau(i))/\beta$.

3 Proof outline

Some intuition The loss of independence amongst the neighbors of each encoding symbol makes it more difficult to analyze the decoding process. This is because it introduces dependencies between the set of recovered input symbols S and the input symbol a released encoding symbol covers.

[2]The original LT codes analysis required that $R = \Omega(\log\frac{k}{\delta}\sqrt{k})$ and minimized the number of symbols required by setting $R = c\log\frac{k}{\delta}\sqrt{k}$ for some constant c. For our analysis of LT codes using a linear congruential generator we need a larger value of R (see the definition in Section 2.5).

We analyze the decoding process by using *pessimistic filtering* to selectively discard symbols to remove the dependencies. The idea behind pessimistic filtering is that discarding symbols can only decrease the probability of successful decoding. Once we have removed the dependencies, the original analysis of LT codes with full randomness will apply (with some modifications).

Intuitively, the earlier a symbol is released, the less uniform the distribution of the released edge. When a degree d symbol is released, we know that exactly $d-1$ of its neighbors lie in the set of processed input symbols, and the smaller this set is, the more constraints we have on the parameters a and b in the linear congruential generator and thus the more biased the distribution of the remaining edge.

To counteract this effect, we pessimistically filter the $O(k^{-1/6}\text{polylog}(k, 1/\delta))$ fraction of symbols that release too early. Symbols that release later than this lose at most about $\log k$ bits of entropy and since a, b contain $2\log k$ bits of entropy, we will be left with about $\log k$ bits of entropy in the distribution of the released edge. The released edge now has a near uniform distribution. We can then use pessimistic filtering to force the distribution to be *exactly* uniform.

Proof outline Our primary task is to prove that the distribution of the set S of already-recovered input symbols remains *uniformly distributed* throughout the execution of the recovery process. If this is the case, from the perspective of the already-recovered input symbols, the neighbors of a new encoding symbol will *appear* to be independent, even though they have been chosen using a linear congruential generator.

It is important to keep in mind the distinction between *recovering input symbols* and *releasing encoding symbols*. When an input symbol is recovered, the corresponding encoding symbol may have released much earlier in the process, even though it has only just now been processed. A released encoding symbol may not even contribute to the recovery of an input symbol, as there could be several released encoding symbols which cover the same input symbol. The first part of our proof (which is concerned with preserving uniformity) only deals with the *recovery* time of *input* symbols. The second part (which ensures that the ripple does not disappear) deals with the *release* time of *encoding* symbols.

In Section 4, we define what we mean for an encoding symbol to release too early. If it releases too early, we are unable to prove that the distribution of the covered input symbol looks uniform. We use pessimistic filtering to discard such misbehaving encoding symbols.

Given these restricted release times, we show in Section 5 that when a new encoding symbol covers an input symbol, the distribution of the input symbol covered is *almost uniform*, *almost all of the time*. This brings us most of the way to the goal of showing that S (after pessimistic filtering) is *exactly* uniform. The first "almost", that the newly recovered input symbol is not exactly uniformly distributed, will be taken care of by pessimistic filtering. We pessimistically filter away the peaks in the distribution to obtain a uniform distribution, at the cost of a small increase in the number of encoding symbols required to decode.

In order to remove the second "almost", we introduce the concept of a *good* set. The idea is that if the distribution is what we want almost all of the time, it must be true for *almost all* sets S. We call these sets *good*.[3] Over the course of the decoding process, the probability that we encounter a *bad* set is very small. So we simply absorb this probability into our overall probability of failure, and then assume for the rest of the proof that there are no bad sets.

In Section 6, we show that even though we discard symbols that release too early, because symbols with different degrees release at different points, the expected number of symbols released at each step is only slightly less than in the full randomness case.

In Section 7, we show that after all of the above steps, we can successfully decode with high

[3] A good set can be interpreted as one for which linear congruential sequences look random.

probability. As in the original LT codes paper, we divide the analysis into two parts. We use a random walk to model the size of the ripple until about R symbols remain to be processed and then use a coupon collector argument to show that we can decode the remaining R symbols. Our random walk argument is more complicated than in the original analysis because the events involved are not actually independent.

In Section 8, we calculate the overhead incurred due to pessimistic filtering and then optimize the parameters R and f to minimize the number of symbols required.

4 Restricting degree release times

For $L \geq R$, we say that a degree d symbol releases too early unless

$$\epsilon_{d,L} = \sqrt{\frac{\alpha d^2 \log k}{\left(\frac{k-L}{k}\right)^{d-1} \frac{L}{k}(k-1)}} \leq f \tag{1}$$

(see Section 2.5 for the definition of the values f, λ and R). In our analysis, for $L \geq R$ we will only allow a degree d encoding symbol to release if it has not released too early. Symbols that release too early are pessimistically filtered away. For the values that we assign f and R, and for $L \geq R$, condition (1) implies $d \leq \lambda k/R$.

We now define a more restrictive condition that is easier to work with and is a sufficient condition for condition (1) to hold. Since $d \leq \lambda k/R$ and $L \geq R$, (1) is true if

$$\left(\frac{\alpha k^2 \lambda^2 \log k}{R^3 f^2}\right) \leq \left(1 - \frac{L}{k}\right)^{d-1}. \tag{2}$$

5 Near-uniform Distribution of Released Edge

We analyze what happens when an input symbol is covered. As observed in Section 3, this is *not* the time when the corresponding encoding symbol is released. We only need to consider the distribution of the covered input symbol, given that the encoding symbol has *already been released and has not released too early.* We only consider symbols of degree 3 and higher, because the neighbors of an encoding symbol with degree less than 3 are already independent.

Fix L, the number of as-of-yet unrecovered input symbols, and d the degree of the encoding symbol. We model the set S of already-recovered symbols as follows. For each input symbol a, we independently place it in S with probability $p = \frac{k-L}{k}$. Let X_a be the indicator random variable of the event $a \in S$. This is actually a relaxation of the definition of S we really want, since the size of S is now binomially distributed with mean $k - L$, and *not* exactly $k - L$. Later we will condition on the size of S being exactly $k - L$. Consider the ratio[4]

$$r_m = \frac{\sum_l \sum_{a,b:\ al+b=m} X_b X_{a+b} \cdots X_{a(l-1)+b}(1 - X_{al+b}) X_{a(l+1)+b} \ldots X_{a(d-1)+b}}{\sum_l \sum_{a,b} X_b X_{a+b} \cdots X_{a(l-1)+b}(1 - X_{al+b}) X_{a(l+1)+b} \ldots X_{a(d-1)+b}}.$$

r_m is the conditional probability that, for some encoding symbol c of degree d which has released, the input symbol m is covered by c but not yet recovered. The numerator counts the number of (a, b) pairs that generate degree d symbols that have been released and cover m. The

[4]To simplify the notation, we assume that all additions and multiplications in the subscript values are computed mod k.

denominator counts the number of (a, b) pairs that generate degree d symbols that have been released.

Let $T^l_{a,b} = X_b X_{a+b} \cdots X_{a(l-1)+b} X_{a(l+1)+b} \cdots X_{a(d-1)+b}$ and $U^l_{a,b} = X_b X_{a+b} \cdots X_{a(l-1)+b}(1 - X_{al+b}) X_{a(l+1)+b} \ldots X_{a(d-1)+b}$, N be the value of the numerator, and D be the value of the denominator. We rewrite r_m as $\frac{N}{D} = \frac{(1-X_m)\sum_l \sum_{a,b:\ al+b=m} T^l_{a,b}}{\sum_l \sum_{a,b} U^l_{a,b}}$.

When $m \in S$, $X_m = 1$, so $r_m = 0$. For $m \notin S$ we lower bound r_m. Conditioned on $m \notin S$, $E[N] = p^{d-1}(k-1)d$ and $E[D] = p^{d-1}(1-p)k(k-1)d$, so $E[N]/E[D] = \frac{1}{(1-p)k} = 1/L$. We lower bound r_m by showing that the numerator and denominator are each close to their respective means. In the following analysis all probabilities and expectations will be conditioned on $m \notin S$.

The numerator Since $m \notin S$, $(1 - X_m) = 1$, we just need to bound $\sum_l \sum_{a,b:\ al+b=m} T^l_{a,b}$. We do this by applying an inequality of Janson. The inequality gives a Chernoff-like lower bound with the strength of the bound depending on the dependence between pairs of the random variables.

Proposition 1 *(Janson bound) [4] Let $\{J_q\}_{q\in Q}$ be a set of independent 0–1 random variables. Let $\{Q_i\}$ be a collection of subsets of Q. Let Y_i, $1 \le i \le n$, be the indicator random variable for the event that $J_q = 1$ for all $q \in Q_i$. Let $Y = \sum_i Y_i$. Define $\Delta = \sum_{Y_i \sim Y_j} \Pr[Y_i \wedge Y_j]$, where $Y_i \sim Y_j$ means that they are dependent (that is, when $Q_i \cap Q_j \ne \emptyset$). Then for any $0 < \epsilon < 1$, $\Pr[Y \le (1-\epsilon)E[Y]] \le \exp\left(-\frac{\epsilon^2 E[Y]}{2+\Delta/E[Y]}\right)$.*

To apply the bound, we need to compute an upper bound on Δ. Consider the equations

$$\begin{gathered} a_1 i + b_1 = a_2 j + b_2, \quad a_1 i' + b_1 = m, \quad a_2 j' + b_2 = m, \\ 1 \le a_1, a_2 \le k-1, \quad 0 \le b_1, b_2 \le k-1, \quad 0 \le i, j, i', j' \le d-1, \end{gathered}$$

where all except d and m are variables. Solutions to these equations model dependencies between two terms $T^{l_1}_{a_1,b_1}$ and $T^{l_2}_{a_2,b_2}$. The number of solutions to this set of equations, at most kd^4, is an upper bound on the number of dependent pairs of terms. If $T^{l_1}_{a_1,b_1}$ and $T^{l_2}_{a_2,b_2}$ are dependent, $\Pr[T^{l_1}_{a_1,b_1} \wedge T^{l_2}_{a_2,b_2}] \le p^{d-1}$, since in the worst case they share all variables. Thus $\Delta \le p^{d-1} d^4 k$. We now apply Janson's bound, with $\epsilon = \epsilon_{d,L} = \sqrt{\frac{\alpha d^2 \log k}{(k-1)p^{d-1}(1-p)}}$. This gives

$$\Pr[N \le (1-\epsilon_{d,L})E[N]] \le \exp\left(-\frac{\epsilon^2_{d,L} E[N]}{2 + \frac{\Delta}{E[N]}}\right) = \exp\left(-\frac{\frac{\alpha d^2 \log k}{(k-1)p^{d-1}(1-p)}(p^{d-1}(k-1)d)}{2 + \frac{p^{d-1}d^4 k}{p^{d-1}(k-1)d}}\right) \le \frac{1}{k^{\alpha/2}},$$

where we upper bounded $1-p$ by 1 and $2 + d^3k/(k-1)$ by $2d^3$ because we are only considering encoding symbols of degree at least 3, and values of k much greater than 3. Then we have

Lemma 1 $N \ge (1-\epsilon_{d,L})E[N]$ *with probability at least* $1 - k^{-\alpha/2}$.

The denominator Next we bound the denominator. We need to bound it from above, so we cannot apply Janson's inequality. Instead, we use Chernoff bounds.

Proposition 2 *(Chernoff bound) Let Y_i, $1 \le i \le n$, be independent 0–1 random variables. Let $Y = \sum_i Y_i$. Then for $0 < \epsilon < 1$, $\Pr[Y \ge (1+\epsilon)E[Y]] \le \exp\left(-\frac{\epsilon^2 E[Y]}{3}\right)$.*

Since we have conditioned on $m \notin S$, we know that $X_m = 0$. To simplify our arguments, we first symmetrize the sum by removing the conditioning on X_m. Next, we bound the symmetrized sum and then show that setting $X_m = 0$ does not change the value of the sum by too much.

Consider the value of the sum without the conditioning on X_m. First we bound the inner sum $D_l = \sum_{a,b} U^l_{a,b}$ and then sum over l. Not all terms in the sum are independent, so a Chernoff bound is not directly applicable. To overcome the dependence, we split D_l into groups of independent terms. For $i \le d-1$, let $I^i_a = \{(a,b) : b = (d \cdot j + i) \cdot a$ for some $j, 0 \le j \le \lfloor k/d \rfloor\}$.

Lemma 2 *For any fixed $a \in [1:k-1], i \le d$, the terms $U^l_{a,b}$ are independent for each $(a,b) \in I^i_a$ such that $b \in [0:k-1]$, and $|I^i_a| \ge \lfloor k/d \rfloor$. Furthermore, for any d, L pair that satisfy (2), at most $\epsilon_{d,L} E[D_l]$ (a,b) pairs do not appear in any of the sets I^i_a.*

Proof: Omitted. ■

Let $(D_l)^i_a = \sum_{(a,b) \in I^i_a} U^l_{a,b}$. Each term within each sum is 1 with probability $p^{d-1}(1-p)$, so $E[(D_l)^i_a] = p^{d-1}(1-p)\lfloor \frac{k}{d} \rfloor$. For $\epsilon = \epsilon_{d,L}$, the Chernoff bound gives

$$\Pr[(D_l)^i_a \le (1-\epsilon_{d,L})E[(D_l)^i_a]] \quad \le \quad \exp\left(-\frac{\epsilon^2_{d,L} E[(D_l)^i_a]}{3}\right) = \exp\left(-\frac{\frac{\alpha d^2 \log k p^{d-1}(1-p)\lfloor \frac{k}{d} \rfloor}{p^{d-1}(1-p)(k-1)}}{3}\right) \le k^{-\alpha}.$$

By taking a union bound over all of the sets I^i_a and all l we obtain

$$\Pr[\sum_l \sum_{a,i} (D_l)^i_a \ge (1+\epsilon_{d,L})p^{d-1}\lfloor \frac{k}{d} \rfloor (k-1)d^2] \le \frac{(k-1)d^2}{k^\alpha} \le k^{3-\alpha}.$$

We bound the number of uncovered (a,b) pairs by $\epsilon_{d,L}E[D]$ by Lemma 2. For these, all we can say is that they sum up to at most $\epsilon_{d,L}E[D]$. If we now condition on $X_m = 0$, we affect at most kd terms in D. So conditioning on the value of X_m can change the value of D by at most $kd \le \epsilon_{d,L}E[D_l] \le \epsilon_{d,L}E[D]$. Thus

Lemma 3 *$D \le (1+3\epsilon_{d,L})E[D]$ with probability at least $1-k^{3-\alpha}$.*

The ratio We can now bound r_m.

Lemma 4 *For all $L > R$ and d that satisfy (2), at least a $1-2k^{7-\alpha/2}$ fraction of the sets S of size $k-L$ satisfy the following:*

Suppose that an encoding symbol of degree d recovers an input symbol when S is the current set of recovered input symbols. Then for all *$m \notin S$, the probability that the encoding symbol recovers m is at least $\frac{1}{L}(1-4f)$. We call such sets S* good.

Proof: (Sketch) Observing that $\Pr(|S| = k-L) > 1/k$, and taking a union bound over L, m and d combined with Lemma 1 and Lemma 3 gives $r_m \ge (1-4\epsilon_{d,L})\frac{1}{L} \ge (1-4f)\frac{1}{L}$ with probability at least $1-2k^{7-\alpha/2}$. Thus a $1-2k^{7-\alpha/2}$ fraction of the sets S are *good*. ■

Corollary 1 *By pessimistic filtering, for any good S we can force the distribution over newly recovered input symbols outside of S to be exactly uniform, while losing a $1/(1-4f)$ factor in the number of encoding symbols required. Furthermore, if we have several independently generated encoding symbols releasing at the time of a single good S, the distributions of their released edges are independent.*

Corollary 2 *Consider an evolution of recovered symbols which is completely uniform (*i.e.*, each set of symbols recovered so far is random). Then with probability at least $1-2k^{8-\alpha/2} > 1-\delta/3$, this recovery process never encounters a bad set.*

6 Number of symbols that release too early

In order to show that the decoding process succeeds with high probability, we first restate and prove our version of the robust uniform release probability proposition (Proposition 14 in [8]). The proposition shows that, with the Robust Soliton distribution, when a new, random input symbol is recovered among L remaining unrecovered input symbols, the expected number of encoding symbols released is lower bounded by $\frac{L}{L-\theta R}$, for a constant $\theta < 1$.

Proposition 3 *(robust uniform release probability): For all $L = k-1, \ldots, R$, the expected number of symbols that do not release too early is at least $\frac{L}{L-\theta R}\left(1 - \frac{\alpha\lambda^2 k^2 \log k}{R^3 f^2}\right)$, where $\theta = \frac{1}{16e}$.*

Note that we actually want to show that the release rate is sufficiently high for good sets (see Lemma 4). The result above was obtained by assuming we had a uniform distribution over all sets. We can lower bound the release rate by upper bounding the release rate for bad sets by $K = O(k)$. Since the bad sets form at most a $2k^{8-\alpha/2}$ fraction of all sets this means that we lose at most $K2k^{8-\alpha/2} = O(k^{9-\alpha/2})$ in our release rate. This results in a lower bound of at least $\frac{L}{L-\theta R}\left(1 - \frac{\alpha\lambda^2 k \log k}{R^2 f^2} - O(k^{9-\alpha/2})\right)$.

7 Random Walk and Coupon Collector

We show that we can successfully decode by showing that the ripple size is always positive with high probability. The proof is in two parts. For $L = R, \ldots, k-1$, we model the ripple size as a random walk. For $L = 1, \ldots, R$ we use a coupon collector argument.

Let X be the LT decode process from an encoding with limited randomness. We lower bound the probability that X succeeds. Ideally, we would like to restrict ourselves to a recovery process that evolves randomly, that is, the next input symbol recovered is chosen uniformly from the set of unrecovered input symbols. We can achieve this with pessimistic filtering, Lemma 4 and its corollaries, and as a result, we can prove a theorem almost as strong as in the full-randomness case.

Theorem 1 *Excluding the contribution from $\tau(k/R)$, the ripple size is positive for all $L = k-1, \ldots, R$, with high probability $(1 - \delta/3)$.*

We show that if there are only R unprocessed input symbols remaining, then the contribution of $\tau(k/R)$ will be sufficient to complete the decoding process with high probability.

Proposition 4 *Using only the contribution of $\tau(k/R)$, the number of encoding symbols released for $L = R, \ldots, 2R$, $K \sum_{L=R}^{2R} r(L)$, is at least $R \log(R/\delta)$ with high probability $(1 - \delta/3)$.*

By earlier results and using pessimistic filtering we can ensure that each released symbol uniformly covers the remaining R unprocessed input symbols. By the results about the well known coupon collector's problem, all the R unprocessed input symbols are covered with probability at least $1 - \delta/3$.

8 Number of encoding symbols required

The total fractional overhead is $\frac{R \log k/\delta}{k} + \frac{\alpha\lambda^2 k^2 \log k}{R^3 f^2} + +k^{9-\alpha/2} + 4f$, where the first term comes from the coupon collector argument, the second term from the symbols that are discarded for releasing too early and the third term from the symbols that are discarded to make the distribution of the covered input symbol exactly uniform. Optimizing over f and R we find that:

Lemma 5 *The number of encoding symbols needed is at most* $k + O(k^{5/6} polylog(k, 1/\delta))$.

9 Acknowledgements

We thank Kenji Obata, Satish Rao, Hoeteck Wee and Jason Waddle for valuable discussions. Special thanks to Michael Luby for posing this problem, suggesting some of the intuition behind the proof, proposing the use of a Poisson random variable to complete the random walk analysis in the original LT codes paper, and pointing out errors in a previous draft of this paper.

References

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM*, pages 56–67, 1998.

[2] J.W. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads. In *Proc. IEEE INFOCOM*, pages 275–283, 1999.

[3] `http://www.digitalfountain.com`.

[4] S. Janson. Poisson approximation for large deviations. *Random Structures and Algorithms*, 1(2):221–230, 1990.

[5] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[6] M. Luby. Information additive code generator and decoder for communication systems. US Patent No. 6,307,487, October 2001.

[7] M. Luby. Information additive code generator and decoder for communication systems. US Patent No. 6,373,406, April 2002.

[8] M. Luby. LT codes. In *Proc. IEEE FOCS*, 2002.

[9] M. Luby. Personal communication, February 2003.

[10] M. Luby, V. K. Goyal, S. Skaria, and G. Horn. Wave and equation based rate control using multicast round-trip time. In *Proc. ACM SIGCOMM*, pages 191–204, 2002.

[11] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Practical loss-resilient codes. In *STOC*, 1997.

[12] P. Maymounkov. Online codes. Technical report, NYU, November 2002. TR2002-833.

[13] A. Shokrollahi. Raptor codes. Technical report, Digital Fountain, June 2003. DF2003-06-001.