# Adaptive rateless coding under partial information

Sachin Agarwal
Deutsche Telekom A.G., Laboratories
Ernst-Reuter-Platz 7
10587 Berlin, Germany
Email: sachin.agarwal@telekom.de

Andrew Hagedorn
Department of ECE, Boston University
8 St. Mary's Street
Boston, MA, USA - 02215
Email: {achag, trachten}@bu.edu

Ari Trachtenberg

***Abstract*— We present novel rateless codes that generalize and outperform LT codes (with respect to overall communication and computation complexity) when some input symbols are already available at the decoding host. This case can occur in data synchronization scenarios, or where feedback is provided or can be inferred from transmission channel models. We provide analysis and experimental evidence of this improvement, and demonstrate the efficiency of the new code through implementation on highly constrained sensor devices.**

## I. Introduction

Rateless codes transmit information (*i.e.,* input symbols) as encoded symbols over lossy communication channels, typically an erasure channel. As their name implies, these codes have the property of being *rateless*, meaning that they have no fixed amount of redundancy: the encoder can produce a practically unending stream of encoding symbols, any fixed number (in expectation) of which can be used to decode a transmission. Rateless codes sometimes impose a small communication overhead in that the number of encoded symbols required at the decoder is slightly more than the number of input symbols. In addition, there may be a small overhead of transmitting meta-information used for decoding.

Random linear codes are one well-studied class of rateless codes known to have a very low overhead; in fact they can be implemented with a constant communication overhead, although their encoding and decoding steps are computationally expensive, and hence unsuitable for encoding long transmissions. Recently, codes proposed by Luby [1] have become popular because of their low encoding and decoding complexity, at the cost of a slightly larger communication overhead than random linear codes. Plain LT codes tend to work well only in asymptotic regimes, and for this reason Raptor codes [2] were introduced, which wrap LT codes within an outer code to improve decoding performance. Finally, rateless codes have been designed and implemented for content distribution in networks [3].

All the codes mentioned above are designed for the case when the decoding host has no *a priori* information about the content being transmitted. In this paper, we consider rateless encoding for situations where the decoding host starts with an (unknown) subset of input symbols. This could occur, for example, in the context of data synchronization [4], where each host has modifications of some common database. It could also occur in a multi-round communication scheme where feedback or side-channel information provides some knowledge of input symbols at the decoder.

## II. Preliminaries

### A. Problem formalization

For consistency with the existing literature, we assume an encoding host has $k$, $b$-bit input symbols comprising set $\mathbf{K}$, $n \leq k$ of which (comprising a set $\mathbf{N}$) are already known at the decoding host. We assume that the encoding host knows the cardinality $n$ of $\mathbf{N}$, without knowing the individual input symbols that comprise the set. The goal is for the decoding host to efficiently determine its missing input symbols (*i.e.,* the set difference $\mathbf{K}$-$\mathbf{N}$), where efficiency is measured with respect to both communication and computational complexity.

### B. Straightforward lower bounds

The communication aspect of this problem is simply a special case of the set reconciliation problem [5], for which a simple lower bound is given by

$$\hat{C_\infty} \geq (k-n)b - \frac{\lg(k-n)}{b}, \tag{1}$$

under the reasonable assumption that $2 < 2^{b-1}$. In other words, at a minimum, the encoder would have to send only a little less than the exact contents of the missing input packets to the decoder.

The computational aspect of this problem is a straightforward adaptation of the classical balls and bins argument initially presented in [1]. The analysis assumes that

computational cost is tied to the number of exclusive or (xor) operations on $b$-bit vectors needed for decoding, with each xor requiring unit time. In addition, input symbols are assumed to be included xor computations in a uniformly random fashion, as with common rateless codes.

We associate bins with xor (encoded) symbols and balls with input symbols. Throwing a ball into a bin thus corresponds to using an input symbol when encoding the corresponding xor symbol. At a minimum, each input symbol must appear in the computation of at least one xor symbol for the decoder to be able to recover all inputs. Let $X_i$ denote the random variable representing the number of balls thrown between the time when $i-1$ distinct bins contain a ball and $i$ distinct bins contain a ball. Then the total number of balls thrown before all bins are non-empty is (assuming $n$ balls start in bins corresponding to the $n$ input symbols known at the decoder):

$$X = X_{n+1} + X_{n+2} + , \ldots , + X_{k-1} + X_k \qquad (2)$$

Clearly $X_i$ follows a geometric distribution with $p = (1 - \frac{i-1}{k})$, so its expectation is $E(X_i) = \frac{1}{1-\frac{i-1}{k}}$. If $n$ input symbols are known at the decoder, we have, by linearity of expectation,

$$E(X) = \sum_{i=k}^{n} \frac{1}{1 - \frac{i-1}{i}} = kln(k-n) + O(k). \qquad (3)$$

Our designed codes asymptotically approach the minimum computation complexity of (3).

## III. LT Codes

We next describe Luby's LT codes [1] and their encoding and decoding methods. In Section III-C we provide motivation for our modification of LT codes for our specific problem.

### A. Code construction

LT codes were first proposed in [1], based on a model similar to their Low-Density Parity-Check matrix cousins [6]. The main contributions of Luby involved demonstrating the utility of ratelessness through codes based on the *robust soliton* probability distribution. Codes generated through this distribution have low encoding and decoding complexity ($O(k\ ln\ \frac{k}{\delta})$ exclusive or operations for reconstruction probability $1-\delta$), as well as a low overhead of ($k + O(\sqrt{k}ln^2(k))$ expected number of encoded symbols needed to decode $k$ input symbols at the decoding host [1].

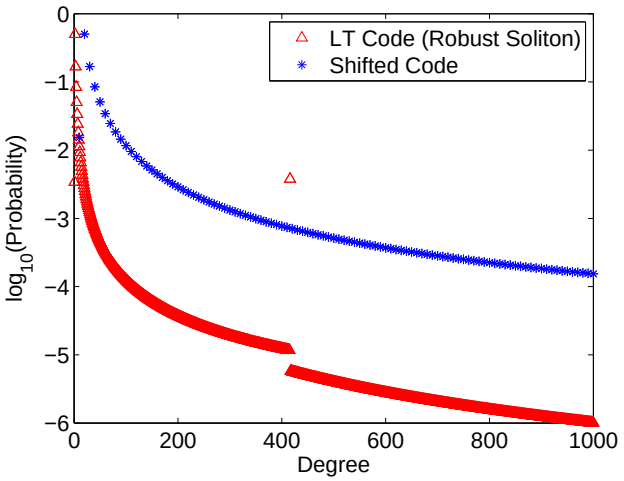


Fig. 1. LT code distribution and proposed Shifted code distribution, with parameters $k = 1000, c = 0.01, \delta = 0.5$. The number of known input symbols at the decoding host is set to $n = 900$ for the Shifted code distribution. The probabilities of the occurrence of encoded symbols of some degrees is 0 with the shifted distribution.

The robust soliton distribution is based on two distributions also proposed in [1]: the *ideal soliton* distribution $\rho(\cdot)$

$$\begin{aligned} \rho(1) &= \frac{1}{k} \\ \rho(i) &= \frac{1}{i(i-1)} \quad \forall i = 2 \ldots k \end{aligned}$$

and the distribution

$$\begin{aligned} \tau(i) &= \frac{R}{ik} \quad \text{for } i = 1, \ldots, \frac{k}{R} - 1 \\ \tau(i) &= R\ ln\left(\frac{R}{k}\right)/k \quad \text{for } i = \frac{k}{R} \\ \tau(i) &= 0 \quad \text{for } i = \frac{k}{R} + 1, \ldots, k, \end{aligned}$$

where $R = c \cdot ln(\frac{k}{\delta})\sqrt{k}$, $c > 0$ is a "suitable constant", and $\delta$ is the maximum probability of decoding failure.

Adding the ideal soliton distribution $\rho(\cdot)$ to $\tau(\cdot)$ and normalizing, Luby obtained the robust soliton distribution $\mu_k(\cdot)$ plotted in Figure 1 for parameters $k = 1000$ input symbols, $c = 0.01$, and maximum reconstruction failure probability $\delta = 0.5$. Note that the robust soliton distribution has a characteristic spike at $i = \frac{k}{R}$, based on the contribution of $\tau(\cdot)$.

### B. Encoding and decoding

Unlike their analysis, LT codes have remarkably simple encoding and decoding algorithms.

In order to create an encoded symbol an encoding

host first chooses a degree $d$ based on the robust soliton distribution, and then, uniformly at random, selects $d$ *distinct* input symbols (which we call *limbs*) from among the $k$ input symbols being encoded. The sum of these input symbols over a suitable finite field (typically $\mathbf{F}_2$) comprises the value of the encoded symbol, which is transmitted to the decoder. The indices of the input symbols selected must also be made available to the decoder, either in the form of a shared seed for a pseudorandom function, or through explicit communication.

For its part, the decoding host uses a simple greedy specialization of the belief propagation algorithm [7], which is typically faster than Gaussian elimination in practice. Specifically, the decoder begins by identifying encoded symbols of degree 1, meaning that each is an exact copies of one input symbol; this initial *ripple* thus yields the value of some input symbols

The known input symbols represent one known parameter in all encoding symbols that use them. As such, encoding symbols of degree 2, which utilize one of these input symbols, can now be utilized to decode some other input symbols (i.e. given a $x$ and $x \oplus y$, one can deduce $y$), resulting in the second ripple. In fact, each ripple involves utilizing known input symbols to reduce the number of unknown parameters in all encoding symbols that utilize these symbols. The robust soliton distribution is designed to determine input symbols at a rate that maintains a nearly constant ripple size, allowing all input symbols to be retrieved eventually with high probability.

### C. Inefficiency under partial information

The design of LT codes presumes no input information on the decoding host. In our case, the decoder already has decoded $n$ input symbols, meaning that any addends of an encoded symbol from the known input set $N$ are redundant. For example, if input symbols $i_1$ and $i_2$ have been decoded, then it is computationally redundant to compute an encoded symbol $x_1 = i_1 \oplus i_2$, as it provides no new information about input symbols.

The number of these *redundant* encoded symbols grows with the ratio of input symbols known at the decoder to input symbols total (*i.e.,* $\frac{n}{k}$). More precisely, a given collection of $d$ distinct limbs of an encoded symbol is a subset of $N$, the input symbols known at the decoder, with probability

$$\prod_{i=0}^{d} \frac{n-i}{k-i}.$$

As such, if $n$ input symbols are known at the decoder, then an additional LT-encoded symbol will provide no new information to the decoder with probability

$$\sum_{d=1}^{k} \mu_k(d) \left( \prod_{i=0}^{d} \frac{n-i}{k-i} \right), \tag{4}$$

which quickly approaches 1 as $n \to k$. It is thus not surprising that the LT codes become less efficient as the decoder learns more input symbols.

## IV. Shifted Codes

### A. Intuition

Intuitively, the robust soliton distribution is too sparse to accommodate known input symbols on the decoder end. The $n$ known input symbols serve the function of degree 1 encoded symbols, disproportionately skewing the degree distribution for LT encoding. We thus propose to *shift* the robust soliton distribution to compensate for the additional functionally degree 1 symbols.

### B. Construction

Formally, the $n$ input symbols at the decoder end have the effect of degree 1 encoded symbols, which can be immediately decoded. As such, they reduce the degree of each encoding symbols by an expected $\left(1 - \frac{n}{k}\right)$ fraction. Our goal is to redistribute the encoding degrees to the original robust soliton distribution.

**Definition** The shifted robust soliton distribution is given by

$$\gamma_{k,n}(j) = 0 + \mu_{(k-n)}(i) \text{ for round} \left( \frac{i}{1 - \frac{n}{k}} \right) = j,$$

where $k$ represents the number of input symbols in the system, $n$ represents the number of input symbols already known at the decoder, and round$(\cdot)$ rounds to the nearest integer.

**Lemma IV.1** *For any $n < k$, $\gamma_{k,n}(j), j = 1..k$ is a probability distribution.*

*Proof:* The proof hinges on the observation that $\frac{1}{1-\frac{n}{k}} \geq 1$ when $n < k$. As such, $\frac{i}{1-\frac{n}{k}}$ and $\frac{i'}{1-\frac{n}{k}}$ will differ by at least 1 for any different integers $i$ and $i'$, meaning that $\sum_j \gamma_{k,n}(j) = \sum_j \mu_{(k-n)}(j) = 1$. ■

We construct a shifted code exactly as an LT-code, but based on the shifted robust soliton distribution $\gamma$. More precisely, given $n$, the number of input symbols already decoded, we pick a degree $d$ with probability $\gamma_{k,n}(d)$ and then xor a corresponding number of the $k$ input symbols, chosen distinctly and uniformly at random.

Any encoding node chosen through the $\mu_k$ distribution to have degree $d$, will have degree roughly $\frac{d}{1-\frac{n}{k}}$ in the new distribution, meaning that we can expect $\bar{d}$ of the input symbols used in the encoding to not be from the $n$ known to the decoder.

### C. Analysis

We can straightforwardly apply the results of [1] to the shifted distribution.

**Lemma IV.2** *A decoder that knows $n$ of $k$ input symbols needs*

$$m = (k-n) + O\left(\sqrt{k-n}\ ln^2\left(\frac{k-n}{\delta}\right)\right) \quad (5)$$

*encoding symbols under the shifted distribution to decode all $k$ input symbols with probability at least $1-\delta$.*

Note that Lemma IV.2 represents a roughly $1-\frac{n}{k}$ fraction of the encoding nodes needed for standard LT codes, which is particularly effective as $n$ approaches $k$. The downside of this shift is that encoding nodes have relatively higher expected degrees.

**Lemma IV.3** *The average degree of an encoding node under the $\gamma$ distribution is given by $O\left(\frac{k}{k-n}\ ln\ (k-n)\right)$.*

*Proof:* The proof follows from the definitions, since a node with degree $d$ in the $\mu_k$ distribution will correspond to a node with degree roughly $\frac{d}{1-\frac{n}{k}}$ in the shifted distribution. Thus, the average degree is:

$$\begin{aligned}
\bar{d_k} &= \sum_{j=1}^{k} j\gamma_{k,n}(j) \\
&= \sum_{i=1}^{k-n} \text{round}\left(\frac{i}{1-\frac{n}{k}}\right)\mu_{(k-n)}(i) \\
&= O\left(\frac{k}{n-k}\right)\sum_{i=1}^{k-n}\mu_{(k-n)}(i),
\end{aligned}$$

and the result follows from the average degree of $\mu$ given in [1]. ∎

The first step in decoding a shifted code involves removing all $n$ known input symbols, and their incident edges, from the decoding graph. Our next lemma establishes the computational complexity of this removal process, after which the resulting graph looks like the decoding graph of a standard LT code under the robust soliton distribution.

**Lemma IV.4** *For a fixed $\delta$, the expected number of edges $R$ removed from the decoding graph upon knowledge of $n$ input symbols at the decoding host is given by*

$$R = O\left(n\ ln(k-n)\right) \quad (6)$$

*Proof:* The total degree of encoding nodes must equal the total degree of input symbols in the decoding graph, meaning, by Lemma IV.3, that:

$$m\bar{d_k} = k\bar{d'}_k,$$

where $\bar{d'}_k$ is the average degree of an input symbol. As such, the expected total degree of $n$ input symbols is given by

$$n\frac{m}{k}\bar{d}_k.$$

Rewriting, with the aid of Lemma IV.2 and recalling that $\delta$ is presumed constant:

$$\begin{aligned}
R &\leq n\frac{(k-n)+O\left(\sqrt{k-n}\ ln^2(k-n)\right)}{k} \\
&\quad O\left(\frac{k}{k-n}\ ln\ (k-n)\right) \\
&= O\left(\left(1+O(\frac{ln^2(k-n)}{\sqrt{k-n}})\right)\right) n\ln(k-n) \\
&= O(n\ln(k-n)).
\end{aligned}$$

∎

We now assemble the various lemmae to determine the computational complexity of shifted decoding, which is asymptotically equivalent to the lower bound derived in Section II-B.

**Theorem IV.5** *For a fixed probability of decoding failure $\delta$, the number of operations needed to decode using a shifted code is*

$$O\left(k\ ln(k-n)\right). \quad (7)$$

### D. Decoding distribution

Our shifted codes rely heavily on a correct knowledge of $n$, the number of input symbols decoded at the receiving host. An underestimate of $n$ at the encoder is wasteful of communication and computation (in fact LT codes are equivalent to estimating 0 decoded inputs), but an over-estimate can be catastrophic, in that it can starve the decoder of lower-degree encoding symbols and effectively stall the decoding.

For example, if $\text{round}\left(\frac{k}{k-n}\right) = r$, then there will be no encoded symbols generated of degree $< r$, which means that if $r$ is at least 2 greater than the number

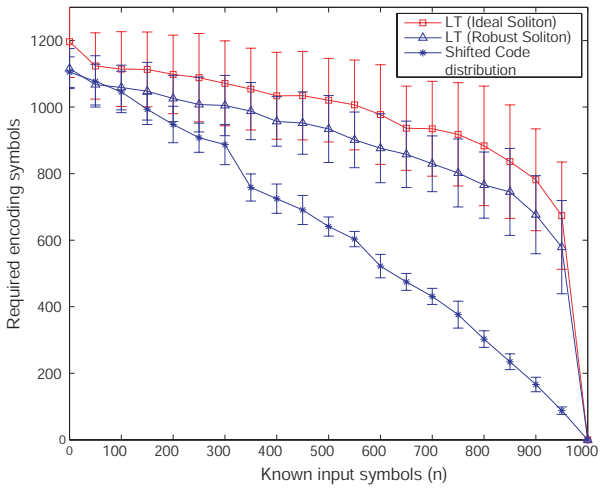


Fig. 2. Performance of LT code and shifted codes for $k = 1000$ over 100 randomized trials per point.

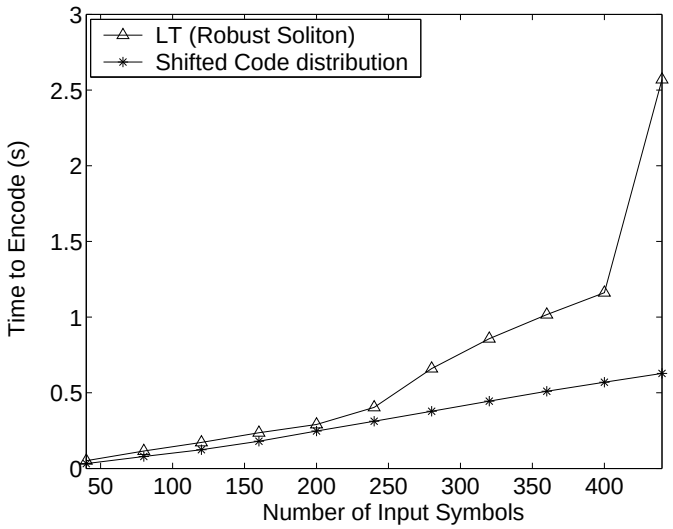


Fig. 3. Time to create enough encoded symbols to decode all unknown input symbols for LT code and proposed shifted code, $n = 0.8 \times k$ over 100 randomized trials.

of decoded input symbols, then decoding will stall and never complete.

In applications where the number decoded input symbols $n$ cannot be determined accurately at the encoder, a more flexible shift might be necessary. Specifically, suppose we know that the decoder will decode $i$ input symbols with probability $p(i)$, resulting in the known input distribution $\mathbf{p}$. Then the appropriate shift of the robust soliton distribution would be:

$$\theta_{\mathbf{p}}(j) = 0 + \sum_{\Delta=0}^{k-1} p(\Delta)\mu_{(k-\Delta)}(i) \quad \text{for round}\left(\frac{i}{(1-\frac{\Delta}{k})}\right) = j. \quad (8)$$

This distribution has the $\gamma$ as the special distribution when $p$ is the impulse distribution (*i.e.*, $p(n) = 1$, $p(i) = 0$ for $i \neq n$).

## V. Comparison and Discussion

The results presented in Section IV-C are asymptotic in nature. In this section we provide experimental performance measures for shifted codes for realistic numbers of input symbols $k$, comparing our codes to standard LT codes. In both cases, we assume that $n$ input symbols have been determined at the decoder side, and decoding proceeds to determine the unknown input symbols.

Our experimental data were collected from two different encoder/decoder implementations of generalized LT codes. The first implementation was designed for standard personal computers with no significant limitations on memory or computation, in our case a 2.0GHz Intel Core 2 iMac with 2GB RAM,. The second implementation was designed for Tmote Sky wireless motes running the TinyOS operating system on an 8 MHz processor with 10KB RAM communicating in the 2.4GHz band at 250kbps.

We implemented a software encoder and decoder for the LT process that can accept any probability distribution to produce encoded symbols and decode them as discussed in Section III-B. The decoding algorithm includes an optimization to efficiently "remove" known input symbols from received encoded symbols in order to compare fairly with our shifted codes.

### A. Number of encoded symbols

Figure 2 plots the number of encoded symbols the number of additional encoding symbols that must be received for full decoding, assuming that $n$ input symbols have already been determined at the decoding host. The shifted code clearly outperforms the robust soliton LT code and is within 100 of the omniscient lower bound in (1). For example, for $n = 900$, the decoding host using the shifted code would only download 152 encoded symbols compared to 678 encoded symbols using the robust soliton LT code.

### B. Encoding and Decoding

Our first experiment examined the computational complexity of creating encoded symbols. In this experiment,

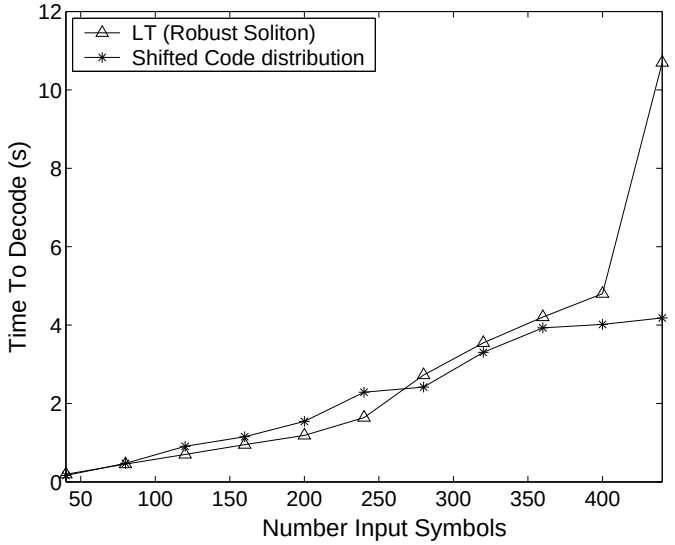


Fig. 4. Time to decode unknown input symbols for LT code and shifted codes for $n = 0.8 \times k$ over 100 randomized trials.

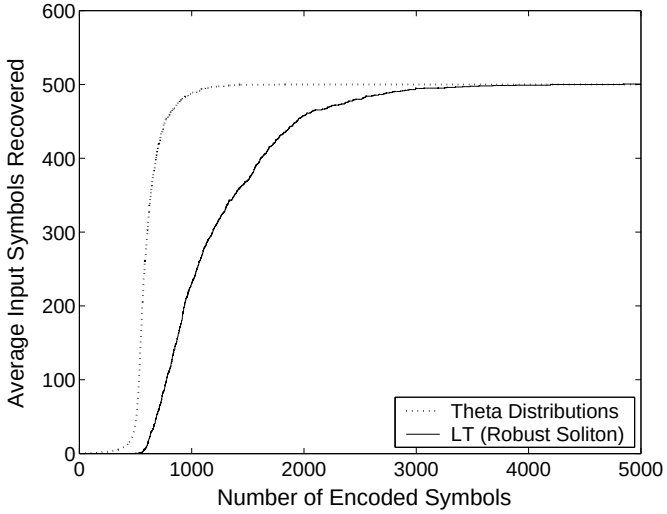


Fig. 5. Comparison of distribution shifting using the $\theta$ distribution (8) and LT decoding. The experiment was conducted over 1000 randomized trials using 500 input symbols.

the number of input symbols known at the decoder was $n = 0.8 \times k$, and the total number of input symbols $k$ was varied. As expected, the amount of computation on average per encoded symbol is larger for the shifted codes, given their higher average degree. For example, when $k = 360$, creating an encoding symbol takes $6.95$ ms for the shifted codes, on average, but only $5.84$ ms for the robust soliton LT code. However, as Figure 3 shows, the total time required at the encoding sensor to create adequate encoded symbols (for successfully decoding) is less for the shifted code than LT codes.

The second experiment examined the computational complexity of the decoding process. In this experiment $n = 0.8 \times k$ and the value of $k$ was varied. Figure 4 shows the amount of time required to recover all unknown input symbols. As $k$ grows, the increasing number of encoded symbols required by the robust soliton LT code becomes more significant and the shifted code performs better.

### C. Distribution shifting

Our final experiment demonstrated the benefit of distributional shifting by means of the $\theta$ distribution of (8).

In this experiment, we first empirically (off-line) determined the distribution $p_1$ of decoded input packets under robust soliton decoding. More precisely, $p_1(i)$ contained the average number of times that exactly $i$ input packets were decoded from the given number of LT-encoded packets. This probability distribution determined a shifted distribution $\theta_1$, as per (8), which we used to empirically derive a probability distribution $p_2$, representing the number of input packets decoded under our $\theta_1$ distribution. This, in turn, produced a $\theta_2$ distribution, etc., yielding a list $p_1, p_2, p_3, \ldots$ and corresponding degree distributions $\theta_1, \theta_2, \theta_3, \ldots$.

Figure 5 shows a comparison of these "Theta" distributions and LT codes. Clearly, the "Theta" distribution shifting decodes input symbols much more quickly than the standard LT codes. It would be of clear interest to replace our empirical evaluations with explicit combinatorial computations, of which we have only lower bounds thusfar, of the appropriate shifts after each decoding iteration. This could be combined with some side knowledge of a transmission channel model to produce channel-matched rateless coding.

## VI. Conclusion

We have proposed shifted codes that outperform LT codes when a fraction of the input symbols is already available to the decoding host, as may occur from preliminary feedback (whether explicit or inferred) or in broadcasting or synchronization environments. Our analysis and experimental results demonstrate the improvement of the shifted code over standard LT codes in terms of overall communication and computational complexity, and show promise for developing rateless codes for specific transmission channel models. Finally, we have demonstrated the feasibility of implementing both sets of rateless codes for extremely constrained sensor devices.

We believe that the proposed codes will be relevant for a wide variety of applications. On the theoretical

side, we believe that they can adapted to work with inner codes in order to improve upon popular rateless code designs, such as Raptor codes [2]. The proposed idea of carefully modifying the probability distribution of LT codes under partial information may also be applicable to other codes. On the applied side, shifted codes should be useful for applications that broadcast updates on wireless channels in a sensor network or a mobile wireless network, because multiple decoding hosts can use the same encoded symbols received over the broadcast channel, irrespective of the particular input symbols available at different decoding hosts. In cases where a feedback channel exists, the shifted code can be continuously adapted, depending on the number of input symbols successfully received at the decoding host. We also envision using shifted codes in storage applications where storage devices can be "repaired" to a master copy in cases of partial accidental erasure and updating outdated data.

## VII. Acknowledgements

This work was supported in part by NSF CAREER grant CCR-0133521 and NSF grants CNS-0435312 and CCF-0729158.

## References

[1] Michael Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.

[2] Amin Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *Proceedings of ACM SIGCOMM '98*, pp. 56–67, September 1998.

[4] S. Agarwal, A. Hagedorn, and A. Trachtenberg, "Near optimal update-broadcast of datasets," in *International Workshop on Data Intensive Sensor Networks 2007 (DISN'07), MDM'07*, 2007.

[5] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Trans. on Info. Theory*, vol. 49, no. 9, pp. 2213–2218, September 2003.

[6] R.G. Gallager, *Low Density Parity Check Codes*, Ph.D. thesis, Massachusetts Institute of Technology, 1963.

[7] F.R. Kschischang, B.J. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, February 2001.