Rateless Coding with Feedback

Andrew Hagedorn, Sachin Agarwal, David Starobinksi, and Ari Trachtenberg

Abstract

The erasure resilience of rateless codes, such as Luby-Transform (LT) codes, makes them particularly suitable to a wide variety of loss-prone wireless and sensor network applications, ranging from digital video broadcast to software updates. Yet, traditional rateless codes usually make no use of a feedback communication channel, a feature available in many wireless settings. As such, we generalize LT codes to situations where receiver(s) provide feedback to the broadcaster. Our approach, referred to as Shifted LT (SLT) code, modifies the robust soliton distribution of LT codes at the broadcaster, based on the number of input symbols already decoded at the receivers. While implementing this modification entails little change to the LT encoder and decoder, we show both analytically and through real experiments, that it achieves significant savings in communication complexity, memory usage, and overall energy consumption. Furthermore, we show that significant savings can be even achieved with a low number of feedback messages (on the order of the square root of the total number of input symbols) transmitted at a uniform rate. The practical benefits of Shifted LT codes are demonstrated through the implementation of a real over-the-air programming application for sensor networks, based on the Deluge protocol.

A version of this paper appeared as:

- A. Hagedorn, S. Agarwal, S. Starobinski, and A. Trachtenberg, “Rateless Coding with Feedback”, *IEEE INFOCOM 2009*.

I. Introduction

Point-to-multipoint wireless data communication, *i.e.,* from a broadcaster to multiple downstream receivers, is gaining popularity with emerging wireless broadcast channels like digital video broadcast and cellular data broadcast [1] that support digital data broadcasting to multiple receivers. Broadcast scenarios also appear naturally in wireless sensor networks, most notably during software updates. Unlike analog broadcast, digital broadcast may also allow a back channel for receivers to communicate with the broadcaster, enabling interactive applications and protocols such as (n)ack-based data dissemination.

Point-to-multipoint wireless communication poses several unique challenges. First, wireless channels are prone to lost packets (packet erasures) due to interference, occlusion, multi-path, etc.; as a result, different receivers may, and often do, receive different subsets of the transmitted data packets. Second, energy constraints often require receivers to be off during various (often differing) time periods during a given broadcast, again leading to the reception of different subsets of broadcast packets at each receiver. The same energy constraints also typically limit computation and memory on receiver units, thus providing a natural limit on the complexity of error coding for the communication channel. Finally, receivers are usually heterogeneous, with the least capable device dictating, or at least heavily influencing, the broadcast protocols.

Erasure codes, which have the *rateless* property of being applicable to any channel loss probability, give application architects many options in choosing appropriate codes to address the problems listed above. For example, 3GPP [1] broadcast uses the Raptor rateless code [2] to implement robust video and data dissemination over cellular wireless channels. Yet, most existing rateless codes do not harness the back channel (from the receiver to the broadcaster) for feedback. The major contribution of this paper is to show that a small amount of feedback, whereby receivers periodically inform the broadcasting sources about the number of successfully decoded input packets, can lead to major communication, memory, and energy usage gains through a judicious modification of the encoding procedure.

The authors are with Boston University department of Electrical and Computer Engineering, except the second author who is with Deutsche Telekom A.G., Laboratories.

In this paper, we propose a new class of rateless codes exploiting feedback, called Shifted LT (SLT) codes. These codes operate similarly to LT codes, except that they shift the original LT probability distribution (the so-called "robust soliton distribution") used to generate the degree of encoded symbols, based on the feedback provided by receivers. We show that these codes keep enjoying the same theoretical properties as the original LT codes, but also achieve significant communication gain as they exploit the feedback information to better distribute the degree of encoded symbols. Our codes employ the the same encoding and decoding algorithms as LT codes, and in addition avoid the redundant encoded symbols when LT codes are used throughout the transmission.

Next, we develop and analytically justify a number of heuristics to limit communication complexity on the feedback channel. In particular, we devise approaches that limit the number of feedback message to the order of the square root of the total number of input symbols, similar to the Real Time (RT) oblivious codes [3] reviewed in the sequel.

We compare the performance of SLT codes to LT and RT codes, both though simulation and through implementation of a real over-the-air programming (OAP) protocol for sensor networks, based on Deluge [4] (to the best of our knowledge, this is the first reported implementation of RT codes and LT codes for sensor networks reprogramming). We demonstrate that SLT codes significantly outperform both of these codes in terms of communication complexity by reducing the number of encoded symbols required for decoding at each receiver. Moreover, unlike RT codes, SLT codes do not congest the feedback channel toward the end of the decoding procedure. Our simulations and experiments also provide insight into the computation, memory, and energy usage of the three types of codes. We also compare these coding-based approaches to standard uncoded Deluge. We use the terms "symbols" and "packets" interchangeably in this work.

A. Organization

We focus our literature search on recent advances in rateless codes in Section II. Some of these coding schemes have similar encoding and decoding algorithms to ours, but the codes themselves are designed differently. Then, in Section III we present a formal problem setup for describing the subsequent analysis of our proposed Shifted LT codes. This is followed by a brief introduction to LT and RT codes, two other rateless codes similar to ours, in Sections III-B and III-C respectively. In Section IV we introduce Shifted LT codes and discuss their properties. In Section V we develop several heuristics to make our codes more practical. We experimentally compare the performance of LT, RT, and Shifted LT codes in Section VI.The application of these codes to the Deluge sensor software updating system is discussed in Section VII. Conclusions and future work are provided in Section VIII.

II. Related Work

Fixed rate LDPC [5] and Turbo codes [6] are used for erasure correction to protect against data packet losses. The well known random linear rateless codes are efficient in communication but are deemed expensive in computation complexity for many practical applications. LT codes [7] and their subsequent derivative, Raptor codes [2], are usually regarded as the first practical rateless codes with efficient decoding algorithms to implement the fountain codes introduced in [8].

Recently, several new rateless codes have been proposed for specific applications and data content. Growth codes [9] have been proposed for data aggregation in lossy sensor networks (*i.e.,* aggregating data from multiple senders to one receiver). RT [3] codes support a low memory the receiver but need more encoded transmissions. The authors in [10] and [11] propose rateless codes for channel erasure-resistant software updates on sensors. However, the random linear codes used in [10] limit the applicability of their approach to large transmissions. The authors in [11] use a genetic algorithm to converge upon rateless codes without explicitly describing the mathematical structure of the code.

The rateless codes proposed in [12] differentiate input data based on their post-decoding importance in video playback, protecting key video frames more than others. This is not the case with our application of Deluge sensor software updates, where each transmitted byte is considered equally important and moreover, all transmitted bytes should be received in order for the transmission to be successful.

In contrast to the schemes listed above, we present a Shifted LT code that uses a feedback channel to improve the overall communication/computational complexity of LT codes. Though there has been work on coding with feedback as a form of hybrid ARQ, for example doped fountain code [13] where feedback is used to restart LT decoding when a ripple has stopped, our work focuses on situations where sender and receiver share some (undetermined) common data. We compare our approach to LT codes and RT codes through extensive experimental data on a simulator and on sensor motes. We note that SLT codes are also applicable in situations where a fraction of the transmitted symbols are already available at the receivers (for example, an outdated copy) [14].

III. Preliminaries

A. Setup

The broadcaster (encoder) has k input symbols that need to be transmitted to all the receivers over a shared wireless broadcast channel (hence, there is no dedicated communication link from the broadcaster to each receiver). Each input symbol may be relatively large (e.g. 10s of kilobytes), making it infeasible to provision a dedicated point-to-point channel for each receiver. In our setup we assume the availability of some additional information - the number of input symbols already decoded at the receivers - at the broadcaster. This information may be modeled as a number $n \leq k$ that is periodically sent from each receiver to the broadcaster via the feedback channel.

In the description of LT, RT and Shifted LT codes in Sections III-B, III-C, and IV respectively we limit the discussion to one broadcaster and one receiver in order to focus on the coding schemes. We briefly discuss multiple-receiver heuristics for applying our codes to broadcast scenarios in Section V-C and show the application of the coding schemes to the Deluge broadcast software update application with multiple receivers in Section VII.

B. LT Codes

We next describe Luby's LT codes [7] and their encoding and decoding methods. In Section III-B.3 we provide motivation for our modification of LT codes for our specific problem.

1) Code construction: LT codes were first proposed in [7], based on a model similar to their Low-Density Parity-Check matrix cousins [15]. 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 [7].

The robust soliton distribution is based on two distributions also proposed in [7]: the *ideal soliton* distribution $\rho(\cdot)$

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

and the distribution

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

where $R = c \cdot ln(\frac{k}{\delta})\sqrt{k}$, $c > 0$ is a "suitable constant", and δ 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 $\mu_k(\cdot)$. Note that the robust soliton distribution has a characteristic spike at $i = \frac{k}{R}$, based on the contribution of $\tau(\cdot)$.

2) Encoding and decoding: 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 pseudo-random function, or through explicit communication. The encoding process therefore creates a bipartite *decoding graph* with input symbols on the left connected to (possibly multiple) encoded symbols on the right.

For its part, the decoding host uses a simple greedy specialization of the belief propagation algorithm [16], 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. The encoding and decoding complexity of LT codes was shown to be $O(klog(k))$ in [7].

3) Inefficiency under feedback: 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

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

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

C. RT Oblivious Codes

Real Time oblivious codes [3] codes do not randomize the degree of encoded symbols as LT codes do. Instead, starting from degree 1 encoded symbols, an RT encoder transmits encoded symbols of increasing degree based upon feedback about the number of decoded symbols. Moreover, the RT oblivious decoder simplifies the memory requirement by discarding any undecoded symbols in real-time, instead of storing these for later decoding as an LT decoder does. The price for this simplified decoder is the increased number of encoded symbol transmissions, although the authors in [3] develop an optimal algorithm for choosing the degree distribution that minimizes the probability of having to discard an encoded symbol. RT codes require a feedback channel, although the authors show that $O(\sqrt{k})$ successful feedback transmissions suffice for their scheme.

The real time characteristic of the RT decoder yields input symbols at a near constant rate (hence the 'real time'), although the decoded input symbols are unordered and therefore, may not be immediately useful. Moreover, the feedback is non-uniform, with most of it occurring toward the end of decoding. This may lead to congestion of the feedback link and a data implosion problem at the broadcaster if it gets overwhelmed by feedback messages. As we shall see in Section VI, our Shifted LT codes yield excellent performance even when feedback is sent at uniformly (in number of decoded input symbols) through the decoding process.

IV. SHIFTED LT CODES

A. Intuition

Intuitively, the robust soliton distribution of LT codes is too sparse (*i.e.,* there are too many lower degree encoded symbols) 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 available at the decoder.

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, these n symbols reduce the degree of each encoding symbols by an expected $\left(1 - \frac{n}{k}\right)$ fraction, possibly making some of the received encoded symbols redundant. Our goal is to counter this effect by creating encoded symbols in a way that redistributes the encoding degrees to the original robust soliton distribution after these n input symbols have been removed.

Definition The Shifted LT distribution is given by

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

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 $n < k$, $\gamma_{k,n}(j), j = 1 \ldots k$ 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 LT code exactly as an LT-code, but based on the Shifted LT distribution γ. 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 μ_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 d of the input symbols used in the encoding to not be from the n known to the decoder. Note that our construction applies regardless of the constants c and δ chosen for the LT distribution.

C. Analysis

The following lemma, adapted from the results of [7], quantifies the communication complexity of Shifted LT codes.

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

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

encoding symbols under the Shifted LT 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 symbols needed for standard LT codes, which is particularly effective as n approaches k. The downside of this shift is that encoding symbols have relatively higher expected degrees.

Lemma IV.3 *The average degree of an encoding node under the γ distribution is given by $O\left(\frac{k}{k-n}\ ln\ (k-n)\right)$ (for $0 \le n \le k-e$, e being the base of the natural logarithm).*

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

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

and the result follows from the average degree of μ given in [7]. ■

In practice, $k \gg e$, and the degree is set to n for $n \ge k-e$ at the encoding host.

The first step in decoding a Shifted LT 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 δ, the expected number of edges E removed from the decoding graph upon knowledge of n input symbols at the decoding host is given by $E = O\left(n\ ln(k-n)\right)$*

Proof: The total degree of encoding symbols 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 δ is presumed constant:

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

■

We now assemble the various lemmas to determine the computational complexity of Shifted LT decoding. Specifically, the decoding computational complexity is the sum of the operations for first removing the edges corresponding to the n input symbols known at the decoder (Lemma IV.4) and subsequent operations for decoding an LT code comprised of $k-n$ input symbols ($O((k-n)log(k-n))$ from [7].

Theorem IV.5 *For a fixed probability of decoding failure δ, the number of operations needed to decode using a Shifted LT code is $O\left(k\ ln(k-n)\right)$.*

V. Heuristics for Practical Implementation

A. Shifted LT codes with feedback

Shifted LT codes lend themselves to a scheme in which a recent value of n is available to the encoder via feedback. In such a scheme the transmitting node would start with $n=0$ for which the probability distribution of the Shifted LT code is the robust soliton distribution of LT codes, and update the distribution on the fly as feedback is received.

Ideally, the Shifted LT encoder (broadcaster) would change the degree distribution for every new value of n at the decoding receiver. Unfortunately, this would involve sending a large number of feedback messages *i.e.,* every time n changes at the decoder. Fortunately, Shifted LT codes perform well even when the encoder has an approximate value of the actual number of decoded input symbols n at the decoder. In particular, an underestimate of n at the encoder may increase the number of encoded symbols required to decode, but will not lead to decoding failure in general. For example, using the original LT code is equivalent to fixing the value of $n=0$ at the encoder and never modifying the robust soliton degree distribution.

1) Non-uniform restriction on feedback: The rate of change in the average degree of an encoded symbol (Lemma IV.3) with increasing n is not uniform during the decoding process of Shifted LT codes. In fact, most input symbols are decoded after n surpasses a certain value $n=\alpha k, 0 \leq \alpha \leq 1$. A feedback message containing the most recent value of n is sent only when the average degree changes by a constant (since the previous feedback). This leads to a *non-uniform restriction* heuristic that limits the feedback of Shifted LT codes to $O(\sqrt{(k)})$ transmissions, the same as RT codes.

To investigate this heuristic for limiting the feedback of Shifted LT codes and determine the constant mentioned above, we start by computing the value of α for which the rate of change in the degree with respect to n is at least 1.

Lemma V.1 *For $n \leq k-e$, the value of α for which the derivative of the average degree with respect to n is at least 1 is approximately $1-\sqrt{\frac{log\ k}{k}}$.*

Proof: First, note that the average degree of a Shifted LT encoded symbol (Lemma IV.3) is a non-decreasing function for $0 \leq n \leq k-e$. Consider its derivative with respect to n (we ignore the

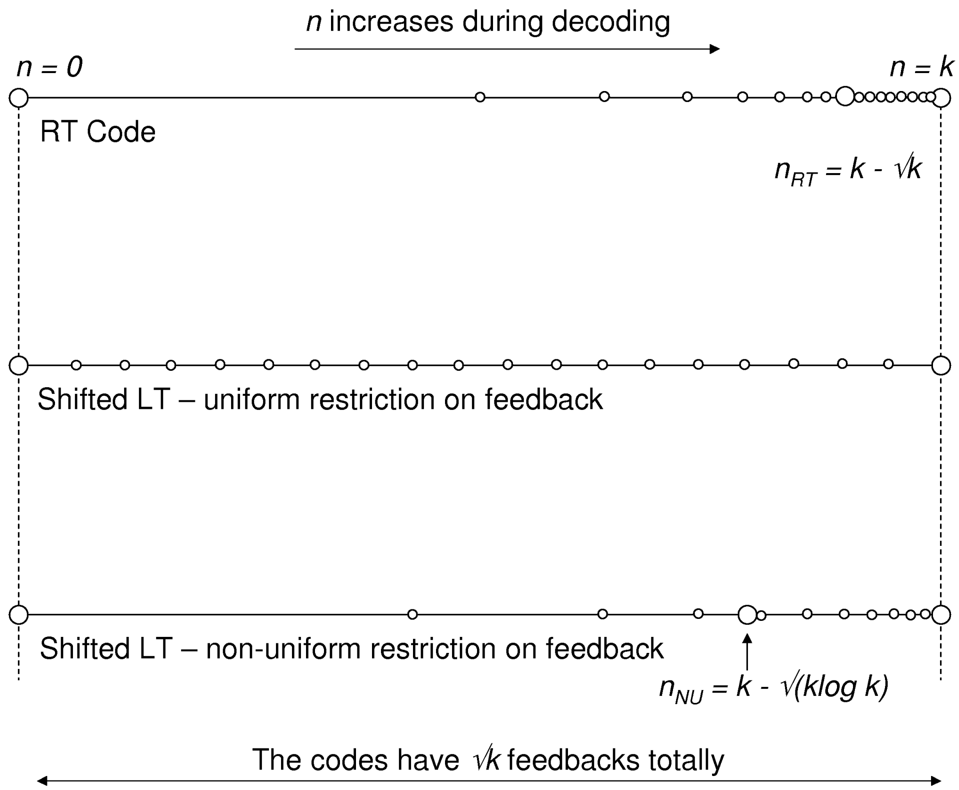

Fig. 1. Feedback strategies for uniform and non-uniform restrictions on Shifted LT and RT codes. Each circle qualitatively corresponds to a situation in which the current value of n is fed back to the encoder.

constant associated with big-Oh notation for this analysis).

$$\frac{d}{dn}\Big(\frac{k}{k-n}log(k-n)\Big) = \frac{k}{(k-n)^2}(log(k-n)-1) \tag{3}$$

Substituting $n = \alpha k$, equating the derivative to 1, and noting that $k \gg 1 \geq \alpha \geq 0$, $\frac{log(1-\alpha)}{k} \leq 0$, and $\frac{1}{k} \to 0$, yields the statement of the Lemma. ■

Since $k \gg e$ usually, it is not practically important to analyze the case when $k - e \leq n \leq k$.

The corresponding value of n, denoted by $n_{NU} = k - \sqrt{klog\ k}$ is somewhat smaller than that for RT codes in [3], where the authors show that the degree increases by more than 1 per change in n for $n_{RT} = k - \sqrt{k}$, as illustrated in Fig. 1.

Early in the decoding, when $n < n_{NU}$, the average degree of an encoding symbol increases by

$$\frac{1}{2}\sqrt{k\ log\ k} - log\ k \quad \approx \frac{1}{2}\sqrt{klog\ k}, \tag{4}$$

obtained by evaluating the expression in Lemma IV.3 at $n = 0$ and at $n = n_{NU}$ and subtracting the former from the latter. We limit the feedback to every time the average degree changes by $\sqrt{log\ k}$ (from its value at the previous feedback), leading to approximately $\frac{1}{2}\sqrt{k}$ feedbacks (obtained by dividing (4) by $\sqrt{logk}$).

During the later decoding stage ($n \geq n_{NU}$) the heuristic sends at most $\sqrt{k}$ feedbacks, one each time the degree changes by (at least) $\sqrt{log\ k}$. Therefore in total, this heuristic sends $O(\sqrt{k})$ feedbacks as n increases from 0 to k, which is equal to the RT code's feedback.

2) Uniform restriction on feedback: In this scheme, the current value of n is communicated back to the encoder every time n increases by $\sqrt{k}$, resulting in $\sqrt{k}$ feedbacks as n increases from 0 to k, as illustrated in Fig. 1. This heuristic has the advantage of not congesting the feedback channel toward the end of decoding, unlike RT codes and the non-uniform restriction on feedback.

3) Performance without final Feedback : Fig. 1 illustrates the congestion issue toward the end of decoding in RT and Shifted LT codes using the non-uniform restriction heuristic. In the worst case, it may happen that all the feedback packets are dropped due to congestion toward the end of decoding. We now show that Shifted LT codes will outperform RT codes when no feedback is transmitted to the encoder after $n = k - \sqrt{k}$. In [3] it is shown that RT codes require $\sqrt{k}\ logk$ encoded symbols to recover the final $\sqrt{k+1}$ input symbols without feedback. Shifted LT codes outperform RT codes in this regard as they require less packets to recover the final $\sqrt{k+1}$ input symbols without feedback. As a consequence of Lemma IV.2 Shifted LT codes requires $\sqrt{k+1} + (k+1)^{\frac{1}{4}}\log^2\ (\sqrt{k+1})$ encoded symbols to recover the final $\sqrt{k+1}$ input symbols without any further feedback.

As k grows toward infinity, a ratio test comparing the relative growth rates of the two functions shows that the function $\sqrt{k}\log k$ grows strictly faster than $\sqrt{k+1} + (k+1)^{\frac{1}{4}}\log^2\ (\sqrt{k+1})$. Therefore, Shifted LT codes requires fewer encoded packets and is able to recover the final $\sqrt{k+1}$ input symbols more quickly if all feedback is lost toward the end of the decoding process due to congestion in the feedback channel.

B. Systematic code heuristic

A simple heuristic mentioned in [3] is to use systematic versions of the codes; first transmit all the input symbols and then use the coding scheme to recover the missed symbols. This reduces the amount of transmission required by both SLT and RT codes to obtain the file. Additionally, at low levels of loss this mitigates the need for restricting feedback.

C. Heuristic for multiple receivers

LT, Shifted LT, and RT codes can be applied to broadcast scenarios with one broadcaster and multiple receivers. LT codes are advantageous because they require no back channel, and the operation of the receivers can be completely asynchronous. That is, each receiver can start receiving and decode encoded packets at any time because no receiver feedback is utilized. On the other hand, RT and Shifted LT codes modify the degree distribution based on feedback from the receivers (in fact, they would be completely ineffective for a newly joining receiver if the minimum degree of encoded packets is ≥ 2). On the other hand, Shifted LT codes generally require much less communication than LT and RT codes and result in significant communication and corresponding power savings across the receivers, as we show in Sections VI and VII.

The Shifted LT construction presented in Section IV-B uses the value of n obtained via feedback. Using an underestimate of n to design the code will result in more redundant encoded symbols but will not stall the decoder. Therefore in case the number of decoded input symbols n is not equal across the receivers, the broadcaster can use the least value of n across the receivers while creating the Shifted LT encoded symbol. In the worst case, $n = 0$, resulting in the encoder creating LT encoded symbols and then the broadcast channel's usage is equivalent to using LT codes. However, for all $n > 0$, Shifted LT codes require lesser communication to deliver the k input symbols to the receivers as compared the LT codes.

VI. Simulations

In our analysis we have defined Shifted LT codes and outlined their properties as well as provided heuristics for practical implementations. We utilize the engineered constants $c = 0.9$ and $\delta = 0.1$, following the related literature [9]. In this section we show the properties of the Shifted LT codes and their variants, and compare them to LT codes and RT codes via simulation. Later, in Section VII, we outline the performance of these codes for the Deluge software updating application on sensor motes.

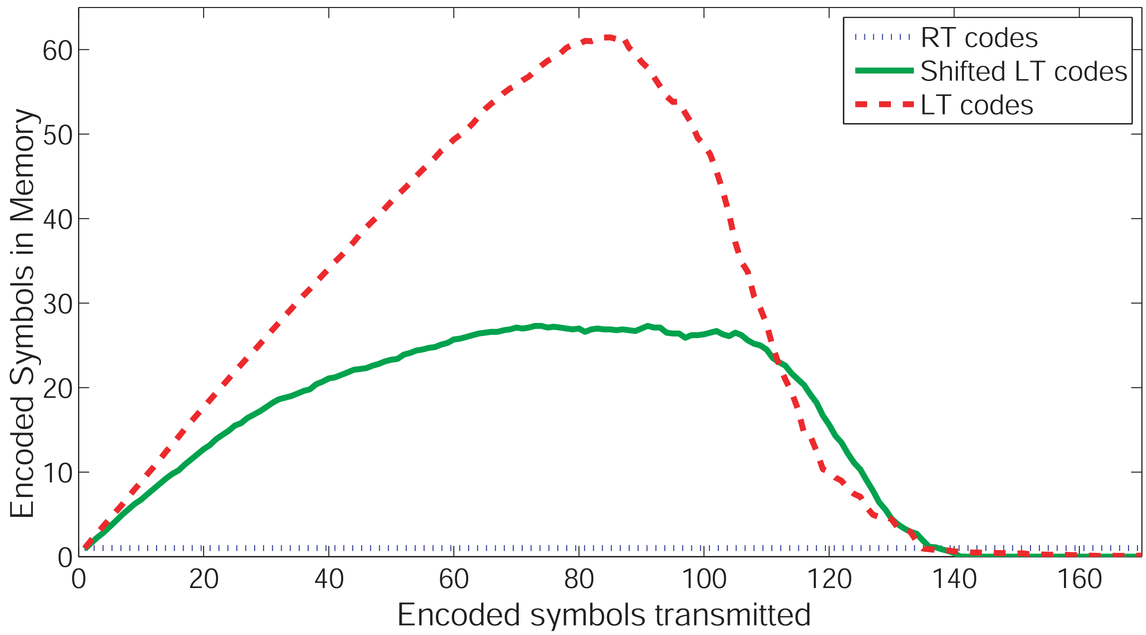

Fig. 2. Memory usage at the decoder as a function of the number of transmitted symbols.

A. Comparison of Shifted LT, LT, and RT codes

The first simulations compare Shifted LT (SLT), LT, and RT codes when there is a feedback channel between the encoder and decoder. In each round of the simulation an encoded packet is generated and transmitted, and decoding is attempted on the received packet (as well as any stored in memory) at the decoder. If an input symbol is recovered then feedback is sent as dictated by each code. For these first simulations Shifted LT codes do not limit their feedback.

For this simulation three metrics were examined: forward channel communication complexity, feedback channel communication complexity, and memory usage. As the number of input symbols increases Shifted LT codes requires fewer encoded transmissions than both LT and RT codes. For k=500, on average Shifted LT codes requires 59% less redundancy than RT codes and 21% less redundancy than LT codes (on average, over 100 trials). On the other hand, the feedback channel communication complexity for Shifted LT codes is greater than either RT codes or LT codes. While RT codes is limited by the changes in its degree and LT codes transmits no feedback, the Shifted LT code transmits feedback every time it recovers one or more input symbols. However, as we will show in Section VI-B most of this feedback is unnecessary. The final metric, memory usage, is shown in Fig. 2 which examines memory usage as the number of encoded transmissions increases for each code. Both LT codes and Shifted LT codes store encoded symbols that are not decoded and require more memory than RT which requires a constant amount of memory since it does not store any packets in memory.

B. Heuristics

To limit the amount of feedback we described non-uniform and uniform restrictions in Sections V-A.1 and V-A.2 respectively. Each of these restricts the amount of feedback to $O(\sqrt{k})$, the level of RT codes. Our simulations examine the forward channel and feedback channel communication complexity for varying k under these restrictions and compare them to RT and LT codes.

The number of encoded packets required to obtain all k input symbols is shown in Fig. 3. When k is small each restriction policy performs similarly, however, as the number of input symbols grows the non-uniform restriction performs best. At 1000 input symbols on average it requires 1314.8 encoded packets compared to 1412.3 for uniform restriction.

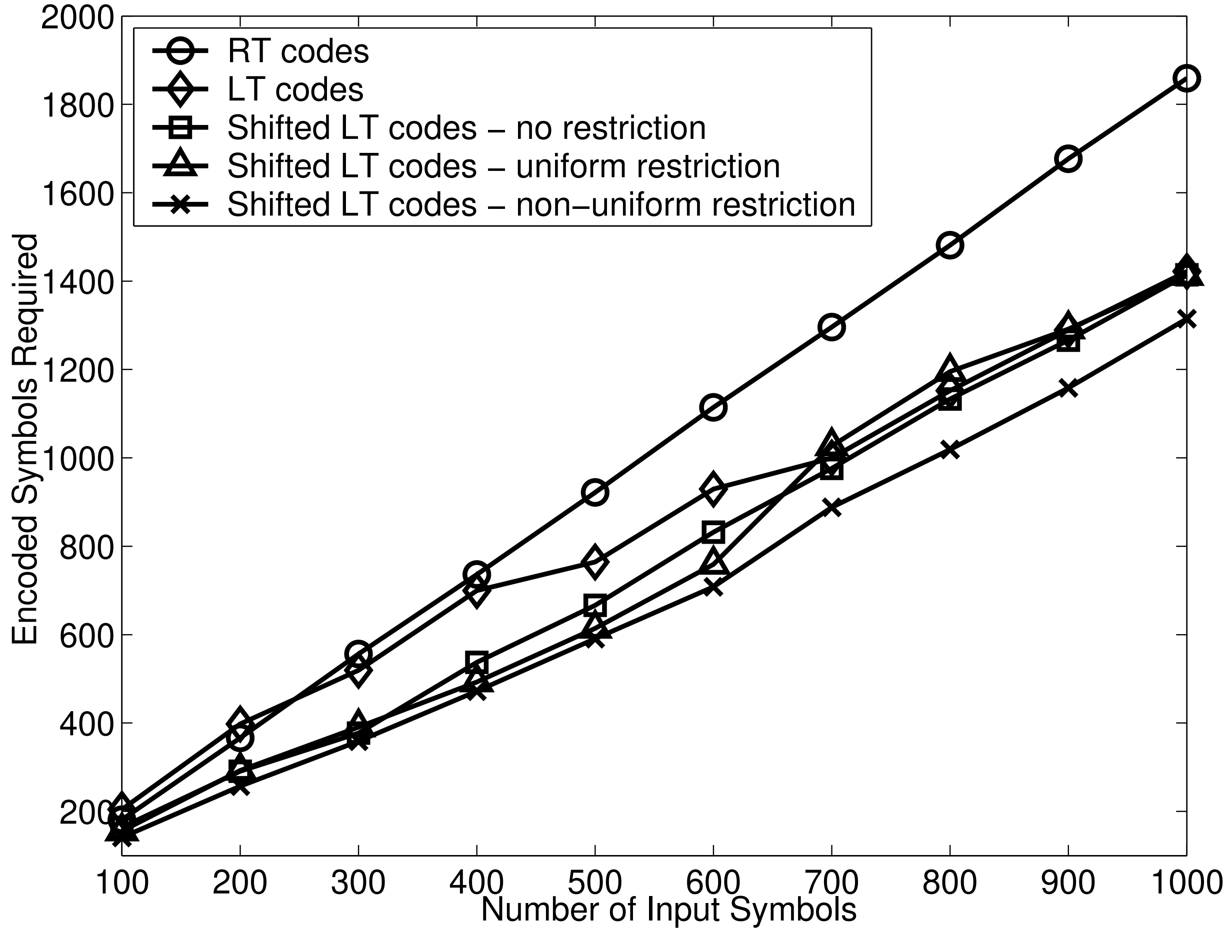

Fig. 3. Number of encoded symbols required to disseminate all k input symbols.

In Fig. 4 we show the number of feedbacks for the various heuristics and codes. It is interesting to note that both the uniform and non-uniform restriction on Shifted LT codes do not increase continuously with k. This is due to the sudden completion of the decoding toward the end of the decoding process and rounding issues of integer degrees.

The main issue with the feedback channel could be its lean rate of data transfer as compared to the forward channel. Simulations to investigate this scenario restrict the relative rate of the feedback compared to the forward channel; for example, by limiting the feedback channel rate to one tenth of the forward channel rate. Fig. 5 shows that under these conditions the Shifted LT code without any restriction on feedback performs poorly. This is due to the large amount of feedback of delayed and inaccurate information (about n) reaching the encoder. Both our heuristics (uniform and non-uniform restriction) perform well under limited feedback conditions.

Another heuristic discussed in Section V-B is a systematic version of each code. The entire input file is transmitted first without encoding and then encoded packets are transmitted to recover any lost packets. The simulations to investigate this heuristic consist of 100 randomized trials for 100 input symbols for RT and SLT codes and examine the communication complexity with packet loss on the forward and feedback channel. The effect of packet loss on the feedback channel is shown in Fig. 6 which compares RT codes and Shifted LT codes when the forward channel loss is fixed at 5 percent (chosen because of the similar performance with no feedback loss of each code). As the feedback loss increases the RT code has a significant increase in the number of encoded packets required, while SLT codes are more tolerant to the loss of feedback packets.

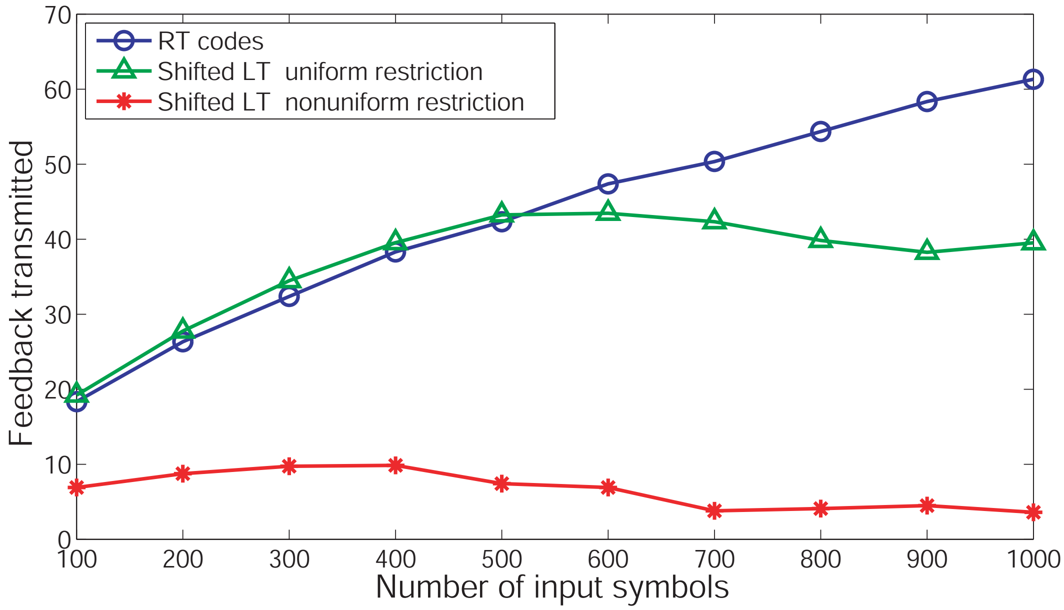

Fig. 4. The number of feedback messages sent for the different codes for increasing number of input symbols k. The "Shifted LT - no restriction" transmits too many ($O(k)$) feedbacks and has been left out of this figure.

C. Multiple receivers

As mentioned in the Introduction, the benefits of Shifted LT codes should be evident in multiple receiver environments as well. Fig. 7 shows this benefit concretely for a 50-node broadcast network, wherein Shifted LT codes require roughly 10% fewer transmissions than LT codes. We note that the amount shifted in these cases is conservatively based on the *minimum* number of input symbols decoded by any node in the network, which can even be a third of the *average* number of input symbols decoded on each node depending on how the (random) encoded packet loss affects each node's decoding process.

VII. Shifted Rateless Deluge

Over-the-Air programming (OAP) represents a key enabling technology for wireless sensor networks, allowing the dissemination of program images over a wireless channel to numerous, typically energy-limited motes. Several OAP protocols have been proposed in the literature, but the *de facto* standard is the Deluge protocol. The work in [10] extended the Deluge protocol to make use of random linear coding for robust software delivery. We further extend both these works to consider the use of Shifted LT, LT, and RT codes. To the best of our knowledge, this is the first implementation of the Deluge protocol using these three rateless codes. This extension of Deluge required a complete redesign of the data communication and request aspects of the Deluge protocol, but maintains the higher level API so that our version of rateless Deluge can be swapped with the traditional Deluge without changing existing tools.

For clarity, we describe experiments based on a fully-featured implementation of the Deluge protocol utilizing LT, Shifted LT, and RT codes for broadcast communication. Our experiments were conducted primarily on TelosB motes, containing an 8MHz micro controller with 10K RAM, and transmitting in the 2.4GHz spectrum at 250 kbps. The motes operated under the TinyOS operating system, and applications were written in nesC, a C-variant commonly used with TinyOS. For fine-grained energy measurements, we further utilized the TOSSIM TinyOS simulator [17] and its extension PowerTOSSIM.

A. Computation

Our first experiment consisted of two TelosB motes, in which one mote serves a single page (consisting of multiple packets) to the other mote. The objective of this experiment was to track computational

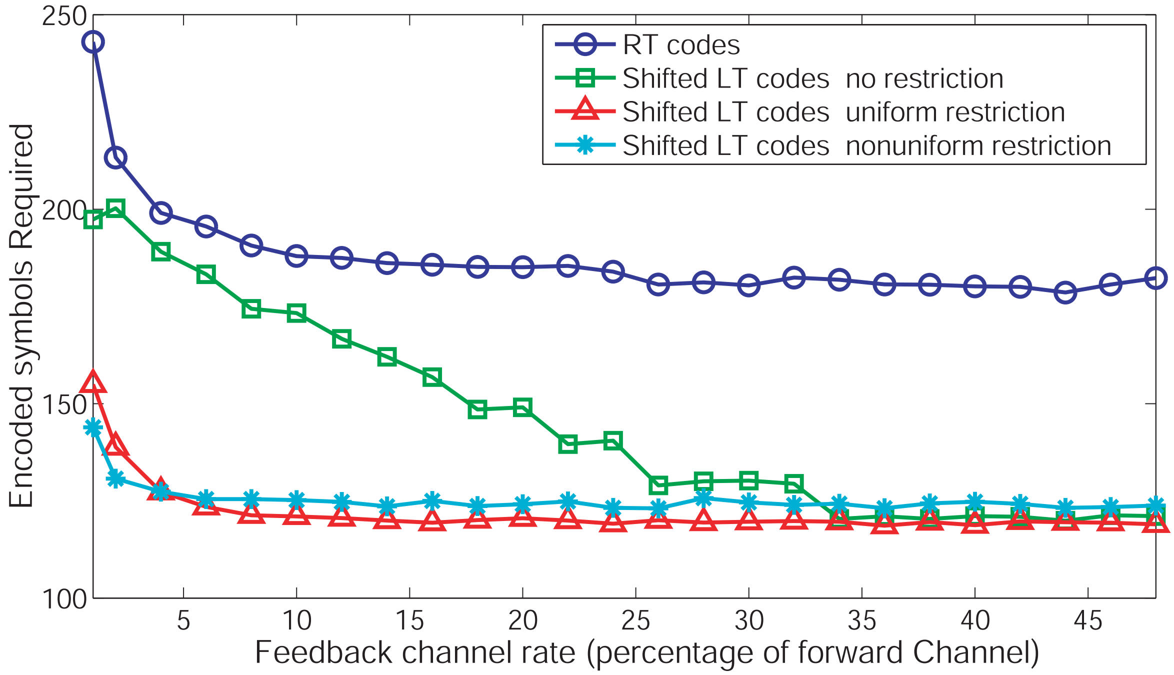

Fig. 5. The number of encoded symbols needed to decode 100 input symbols, as a function of the feedback channel rate.

load on the motes for the various rateless encoding systems: LT, SLT, and RT. Fig. 8 shows that the sample SLT computation load spikes more often than the RT load, mostly due to the need to recompute shifted distributions. These shifts can be instead pre-computed at the expense of increased memory usage. The overall computation time of RT and LT codes is comparable (8.83 seconds versus 10.82 seconds respectively), but the Shifted LT codes require significantly more computation time (23.22) overall when the distributions were recomputed.

B. Communication and Energy

Our second experiment consisted of eleven motes, one of which broadcast five pages in memory (totally 11.5K) to the ten other motes. All feedback channels from the ten motes to the broadcaster were set to have a 5% packet loss rate, and the forward channel loss rates were varied from 0% to 9%.

Our results in Fig. 9, averaged over fifty trials, show that Shifted LT codes transmit fewer packets than LT codes for complete dissemination of the five pages, and that both LT variants are significantly more efficient than RT codes, especially as packet loss rates increase. Equally importantly, Fig. 10 shows the energy measurements of the various codes measured using PowerTOSSIM simulator software (which, unfortunately, cannot simultaneously measure computation time). These experiments demonstrate that Shifted LT codes provide a roughly 15% improvement in energy savings compared to RT codes, which can be quite significant for battery-powered sensors. In effect, the significantly lower communication costs of SLT codes outweigh their additional computational burden, with respect to LT and RT codes. All the codes outperform standard Deluge (no coding) for even moderate packet loss rates in the forward channel.

VIII. Conclusions

In the typical case when a feedback channel is present, Shifted LT codes provide an easily implemented improvement over existing rateless codes, most notably Luby-Transform (LT) and Real-Time oblivious (RT) codes. The corresponding improvements in communication complexity, energy usage, and, in certain

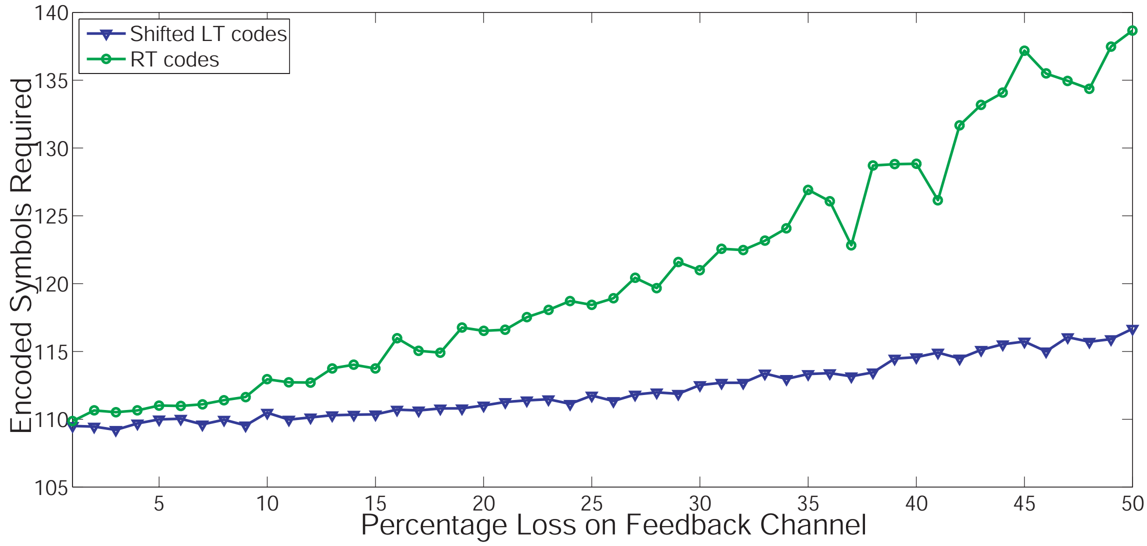

Fig. 6. The number of encoded symbols needed to decode 100 input symbols, as a function of the feedback channel loss rate. The forward channel loss rate is fixed at 5%.

cases, memory requirements are even starker within a broadcast environment where these savings are multiplied by the number of receivers.

The reason for the improvements is intuitively linked to increasing efforts needed by existing decoders to target undecoded symbols. Our shifted codes demonstrate that a modest amount of feedback can significantly reduce this problem, and result in valuable savings, which we demonstrate analytically, in simulations, and through a re-implementation of the popular Deluge software updating protocol for wireless sensor networks.

We expect that our improvements would carry over to derivatives of the rateless codes, such as the well-known raptor codes. Our basic approach of modifying the degree distribution via "Shifting" is applicable to codes other than LT codes. Applying our methods to other codes remains an important future goal. In our Shifted LT code the feedback channel only serves to communicate the state of the decoding process to the encoder; other side information, such as channel characterization information, may eliminate the need for the feedback channel.

Acknowledgment

The authors thank Philip J. Schroeder for simulating some experiments presented in this paper, Moshe Laifenfeld for useful discussions, and the anonymous referees for drawing our attention to the work in [13]. This work was completed while Andrew Hagedorn was an intern at Deutsche Telekom Laboratories. This research was supported in part by a grant from Deutsche Telekom Laboratories, and by US National Science Foundation grants CNS-0132802, CNS-0435312, and CCF-0729158.

References

[1] "Third generation partnership project", http://www.3gpp.org/.
[2] Amin Shokrollahi, "Raptor codes", *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.
[3] Amos Beimel, Shlomi Dolev, and Noam Singer, "Rt oblivious erasure correcting", *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1321–1332, 2007.
[4] J.W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale.", in *SenSys'04*, Baltimore, Maryland, USA, Nov. 2004.
[5] R.G. Gallager, *Low-density parity-check codes*, MIT Press, Cambridge, MA, 1963.
[6] A. Glavieux C. Berrou and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes", *Proc. IEEE Int. Conf. on Communications*, pp. 1064–1070, 1993.
[7] Michael Luby, "Lt codes", in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.

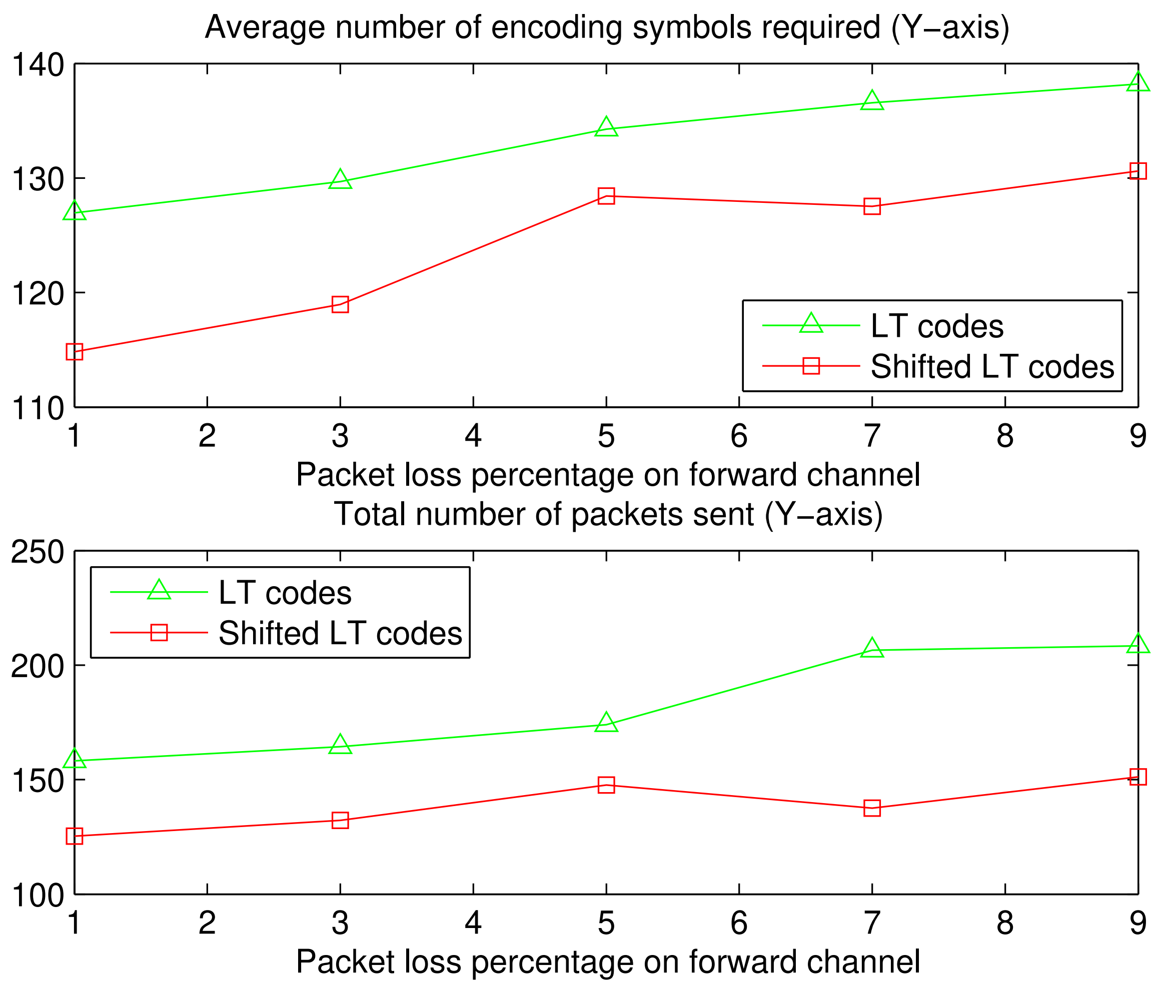

Fig. 7. The number of encoded symbols needed to decode 100 input symbols at 50 receiving nodes, for various forwarded packet loss probabilities.

[8] 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.

[9] Abhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein, "Growth codes: maximizing sensor network data persistence", in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2006, pp. 255–266, ACM.

[10] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes", in *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, 2008.

[11] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. F. Harris, and M. Zorzi, "Synapse: A network reprogramming protocol for wireless sensor networks using fountain codes", in *SECON '08: Proceedings of the IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2008.

[12] *Unequal Growth Codes: Intermediate Performance and Unequal Error Protection for Video Streaming*.

[13] S. Kokalj-Filipovic, P. Spasojevic, E. Soljanin, and R. Yates, "Arq with doped fountain decoding", in *ISSSTA 08': International Symposium on Spread Spectrum Techniques and Applications*, 2008.

[14] S. Agarwal, A. Hagedorn, and A. Trachtenberg, "Rateless codes under partial information", in *ITA '08: Information Theory and Applications Workshop*, 2008.

[15] R.G. Gallager, *Low Density Parity Check Codes*, PhD thesis, Massachusetts Institute of Technology, 1963.

[16] 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.

[17] Phil Levis, "Tossim: Accurate and scalable simulation of entire tinyos applications", in *In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

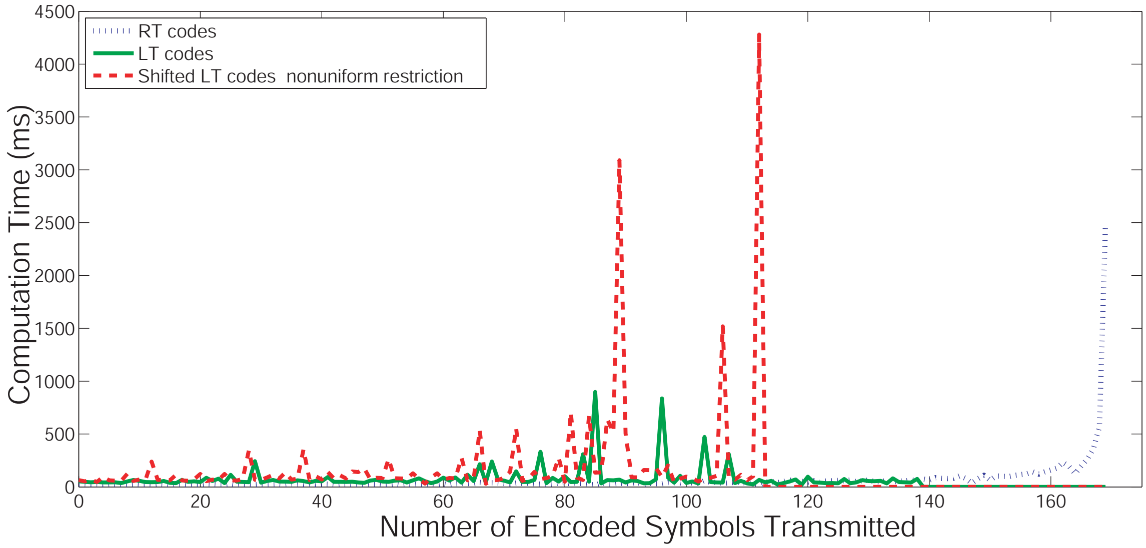

Fig. 8. The amount of time required to decode a randomly chosen encoded packet, as a function of the number of encoded symbols already transmitted.

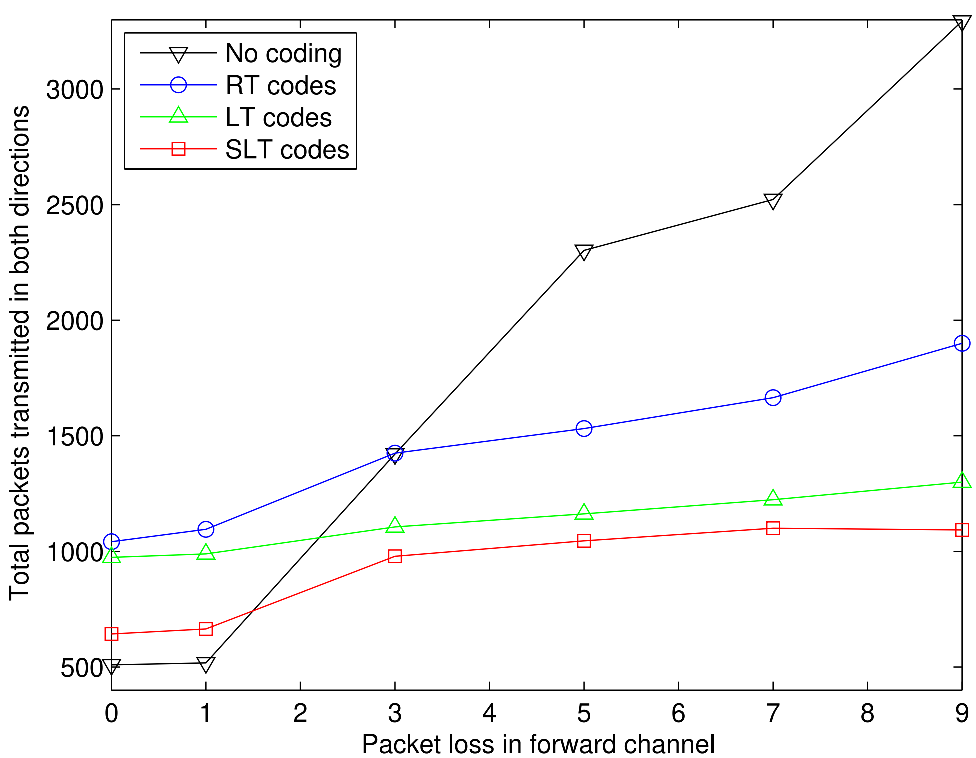

Fig. 9. The total number of packets transmitted on forward and feedback channels in order to disseminate a five page program to ten motes using variants of the Deluge over-the-air programming protocol.

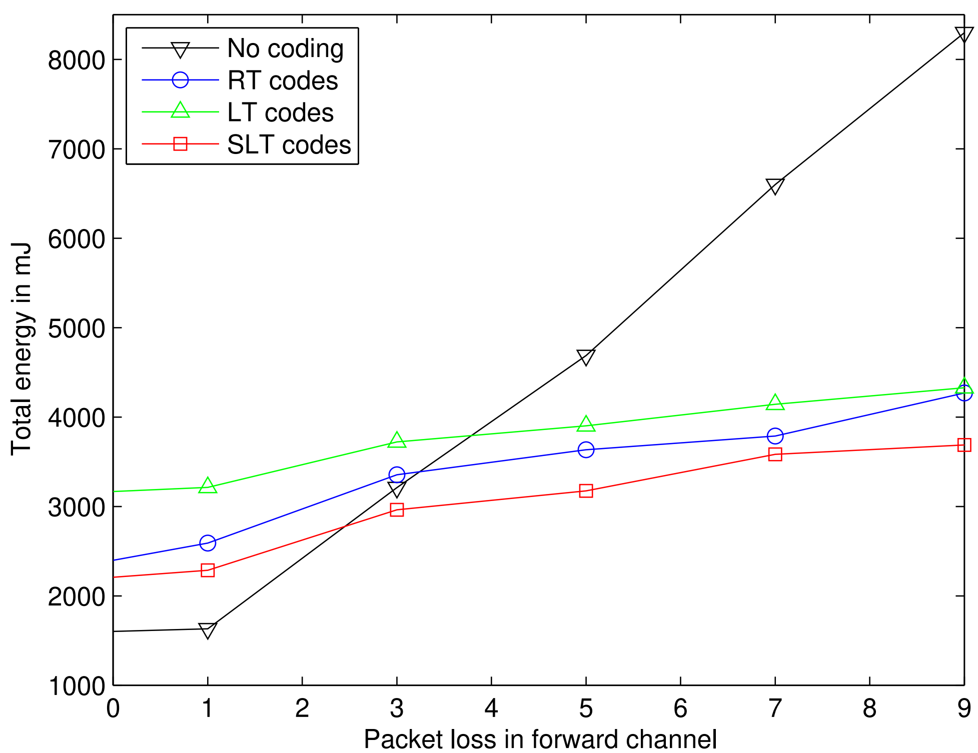

Fig. 10. Total energy used by all the motes for communication and decoding during the dissemination of a five page program using a variant of the Deluge over-the-air programming protocol.