Feedback in LT Codes for Prioritized and Non-Prioritized Data

Jesper H. Sørensen, Petar Popovski, Jan Østergaard,
Aalborg University, Department of Electronic Systems, E-mail: {jhs, petarp, jo}@es.aau.dk

***Abstract*—In this paper feedback in LT codes is investigated. The considered type of feedback is acknowledgments, where information on which symbols have been decoded is given to the transmitter. Our analysis reveals that acknowledgments has a very low potential in LT codes with standard degree distributions. Motivated by this, we analyze the impact of acknowledgments on multi-layer LT codes. In this case, feedback proves advantageous. By using only a single feedback message, it is possible to achieve a significant performance improvement compared to traditional LT codes.**

I. Introduction

Rateless codes are capacity achieving erasure correcting codes. Common for all rateless codes is the ability to generate an infinite amount of encoded symbols from k input symbols. Decoding is possible when $(1+\epsilon)k$ encoded symbols have been received, where ϵ is close to zero. Rateless codes are attractive due to their flexibility. Regardless of the channel conditions, a rateless code approaches the capacity, without the need for feedback during transmission. Only the terminating feedback is necessary. Successful examples are LT codes [1] and Raptor codes [2]. LT codes are the focus of this work.

Although LT codes do not rely on feedback, a feedback channel might be available in the network, e.g. due to the system design of layers lower than the one, at which rateless coding is applied. In certain scenarios, a rateless code might be applied because the feedback channel is unreliable or inadequate. Instead of disregarding the feedback channel, it should be exploited. The question of how has recently received attention.

An approach called doped fountain coding is proposed in [3], where the receiver feeds back information on the current buffer content. This makes the transmitter able to transmit input symbols, which accelerate the decoding process the most. In [4], another approach called Real-Time oblivious erasure correcting is proposed. This approach utilizes feedback telling how many of the k input symbols have been decoded. With this information the transmitter chooses a fixed degree for future encoded symbols, which maximizes the probability of decoding new symbols. The same type of feedback is applied in [5], but with the purpose of minimizing the redundancy in the code. The suggested reaction to the feedback is to shift the degree distribution towards higher degrees, in order to increase the probability that future encoded symbols include data, which has not been decoded yet.

In this work, feedback in the form of acknowledgments (ACKs) is considered. In this context an ACK is a message telling the transmitter which input symbols have been decoded, not just how many, as is the case in [4], [5]. To the best of our knowledge, this is the first time rateless codes in combination with ACKs have been investigated. Such a study is of fundamental interest due to the property of LT codes to adapt using only the terminating feedback. Our study shows that intermediate feedback may increase the performance, but it essentially interferes with the structure of the LT code. We will analyze its impact on LT codes, both for prioritized and non-prioritized data. In the case of prioritized data, we analyze multi-layer LT codes [6] in combination with our proposed application of feedback. We show that these LT codes are able to benefit much more from ACKs than traditional LT codes for non-prioritized data. Another important contribution of this work is the identification and description of an inherent adaptive mechanism in LT codes. That mechanism makes LT codes able to adapt to the current decoder state, without the need for feedback during transmission. When applying feedback, it is important to bear this mechanism in mind, because any actions based on the feedback should not negatively interfere with it.

The remainder of this paper is structured as follows. Section II gives an introduction to LT codes. The analytical work is described in section III, followed by a numerical evaluation in section IV. Finally conclusions are drawn in section V.

II. Background

Here an overview of standard LT codes is given. Assume we wish to transmit data, which is divided into k *input symbols*. From these input symbols a potentially infinite amount of encoded symbols, also called *output symbols*, is generated. Output symbols are XOR combinations of input symbols. The number of input symbols used in the XOR is referred to as the *degree* of the output symbol, and all input symbols contained in an output symbol are called *neighbors* of the output symbol. The output symbols of an encoder follow a certain degree distribution, $\pi(i)$. The encoding process of an LT code can be broken down into three steps:

[Encoder]

1) Randomly choose a degree i by sampling $\pi(i)$.
2) Choose uniformly at random i of the k input symbols.
3) Perform bitwise XOR of the i chosen input symbols. The resulting symbol is the output symbol.

This process can be iterated as many times as needed, which results in a rateless code.

A widely used decoder for LT codes is the belief propagation (BP) decoder. The strength of this decoder is its very

978-1-4673-1881-5/12/$31.00

low complexity [2]. It is based on performing the reverse XOR operations from the encoding process. First, all degree-one output symbols are identified, which makes it possible to recover their corresponding neighboring input symbols. These are moved to a storage referred to as the *ripple*. Symbols in the ripple are *processed* one by one, which means they are XOR'ed with all output symbols, who have them as neighbors. Once a symbol has been processed, it is removed from the ripple and considered decoded. The processing of symbols in the ripple will potentially reduce some of the buffered symbols to degree one, in which case the neighboring input symbol is recovered and moved to the ripple. This is called a symbol *release*. This makes it possible for the decoder to process symbols continuously in an iterative fashion. The iterative decoding process can be explained in two steps:

[Decoder]

1) Identify all degree-1 symbols and add the corresponding input symbols to the ripple.
2) Process a symbol from the ripple and remove it afterwards. Go to step 1.

Decoding is successful when all input symbols have been recovered. If at any point, the ripple size equals zero, decoding has failed. The receiver then either signals a decode failure, or waits for more output symbols. In the latter case, new incoming symbols are initially stripped for already recovered input symbols. If the resulting output symbol has degree one, the decoding process is restarted.

III. Analysis

The considered performance metric is average overhead. Thus, impact of feedback on the probability of receiving redundant symbols is important. First single-layer LT codes for non-prioritized data, are analyzed, followed by multi-layer LT codes for prioritized data.

The type of feedback considered is ACKs. These are used to inform the transmitter, which input symbols have been decoded. This allows the transmitter to exclude the ACK'ed symbols from future encoding. This has the following impact on the encoding process: In step 1 of the encoder, the new decreased number of input symbols is taken into account in the degree distribution. Thus the parameter k is updated to $k' = k - M$, where M is the number of ACK'ed symbols. In step 2, the i input symbols are chosen among the k' undecoded input symbols. The cost of transmitting the ACKs is not taken into account and a lossless feedback channel is assumed.

We consider a single receiver scenario in this work, although the flexibility of LT codes makes them attractive in broadcast scenarios. The motivation behind this is to first achieve insight on how a single user can benefit from ACKs, when it is served individually. This insight is important when extending to multiple users.

A. Single-Layer LT Codes

First we analyze traditional single layer LT codes, where ACKs are not applied. This serves the purpose of understanding when redundancy occurs in such codes. We then go on to analyze the impact of adding feedback in the form of ACKs.

The output of a traditional LT encoder is a symbol of degree i. It might include already decoded data, which is removed upon arrival by the decoder. Hence, the degree is reduced, before the decoding process potentially restarts. We denote the *reduced degree* i' and its distribution $\pi'(i')$.

Lemma 1. *(Reduced Degree Distribution) For any traditional LT code, if the encoder applies $\pi(i)$, and L out of k input symbols remain undecoded, the reduced degree distribution, $\pi'(i')$, is found as*

$$\pi'(i') = \sum_{i=i'}^{i'+k-L} \left(\pi(i) \cdot \frac{\binom{L}{i'}\binom{k-L}{i-i'}}{\binom{k}{i}} \right) \quad \text{for } 0 \leq i' \leq L,$$

$$\pi'(i') = 0 \quad \text{for } L < i' \leq k.$$

Proof: A degree reduction from i to i' happens when i' neighbors are among the L undecoded symbols and the remaining $i-i'$ neighbors are among the $k-L$ already decoded symbols. The probability of this event is found using the hypergeometric distribution. A degree reduction to i' can happen from any symbol of original degree $i = i', \ldots, i' + k - L$.■

Lemma 1 says that the degree distribution experiences a shift towards lower degrees when a subset of the input symbols has been decoded. Thus, although the original degree distribution is constant, $\pi'(i')$ is dynamic. This is important, since low degree symbols are preferable late in the transmission. New symbols can serve one of two purposes: Either $i' \geq 2$ and the symbol is stored in the buffer, or $i' = 1$ and the decoding process restarts. In short, new symbols either build up or release potential. Late in the transmission, it is preferable to start releasing potential instead of building up even more. This is achieved by the inherent shift of $\pi'(i')$.

If the degree is reduced to zero, the symbol is redundant. Thus, $\pi'(0)$ is important when trying to minimize the overhead of the LT code. This probability increases as the decoding progresses. In Lemma 2 we show how ACKs can decrease this probability.

Lemma 2. *(Acknowledgments Reduce Redundancy) $\pi'(0)$ is a strictly decreasing function of M, $0 \leq M \leq k - L$, where M denotes the number of acknowledged input symbols.*

Proof: When $i' = 0$ and the decreased k, caused by the ACK'ed symbols, is taken into account, the equation in Lemma 1 reduces to

$$\pi'(0) = \sum_{i=0}^{k-M-L} \left(\pi(i) \cdot \frac{\binom{k-M-L}{i}}{\binom{k-M}{i}} \right).$$

The fraction of binomial coefficients is a strictly decreasing function of M for positive L, because

$$\binom{k-M-L}{i}-\binom{k-(M+1)-L}{i}<\binom{k-M}{i}-\binom{k-(M+1)}{i},$$
$$\frac{(k-M-L)!}{i!(k-M-L-i)!}-\frac{(k-M-1-L)!}{i!(k-M-1-L-i)!}<\frac{(k-M)!}{i!(k-M-i)!}-\frac{(k-M-1)!}{i!(k-M-1-i)!},$$
$$\prod_{j=0}^{i-1}(k-M-L-j)-\prod_{j=0}^{i-1}(k-M-1-L-j)<\prod_{j=0}^{i-1}(k-M-j)-\prod_{j=0}^{i-1}(k-M-1-j),$$
$$i\prod_{j=1}^{i-1}(k-M-L-j)<i\prod_{j=1}^{i-1}(k-M-j),\ \forall M,\ 0\le M\le k-L. \blacksquare$$

Unfortunately, reducing $\pi'(0)$ is not the only consequence of ACK'ing. It also impacts the rest of $\pi'(i')$. In general $\pi'(i')$ will experience a shift towards higher degrees, when symbols are ACK'ed. Hence, ACK'ing data counters the desirable dynamic property of the reduced degree distribution. In fact, when the maximum number of ACK's is made, the dynamic property is eliminated entirely.

Lemma 3. *(Acknowledgments Counter Dynamic Reduced Degree) For any choice of original degree distribution, if $M = k-L$, where M is the number of acknowledged symbols, the reduced degree distribution equals the original degree distribution.*

Proof:

$$\pi'(i') = \sum_{i=i'}^{i'+k-M-L}\left(\pi(i)\cdot\frac{\binom{L}{i'}\binom{k-M-L}{i-i'}}{\binom{k-M}{i}}\right)$$
$$= \sum_{i=i'}^{i'}\left(\pi(i)\cdot\frac{\binom{L}{i'}\binom{0}{i-i'}}{\binom{L}{i}}\right) = \pi(i') \qquad \blacksquare$$

We conclude that the proposed application of ACK's has both a positive and a negative effect on a standard LT code. On one hand, it reduces the probability of redundancy. On the other hand, it eliminates an important adaptive mechanism in LT codes. We propose to solve this by applying an adaptive degree distribution. More specifically, we apply the distribution in Lemma 1, with the adjustment that $\pi(0) = 0$, because we obviously do not want to generate symbols of degree zero. Moreover, since we truncate the distribution, we must normalize. In this way we mimic the adaptive mechanism and ensure that symbols with reduced degree 0 do not occur. The above observations can be summarized in the following corollary.

Corollary 1. *(Feedback Based Adaptive Degree Distribution) Under the assumption that all decoded symbols have been ACK'ed, the following defines a distribution which avoids symbols of reduced degree zero and preserves the adaptive mechanism of LT codes:*

$$\rho(i) = \sum_{j=i}^{i+k-L}\frac{\pi(j)\cdot\binom{L}{i}\binom{k-L}{j-i}}{(1-\pi'(0))\binom{k}{j}}, \qquad \text{for } 1\le i\le L. \quad \blacktriangledown$$

A simulation has been performed with the purpose of evaluating the application of ACKs in standard LT codes. Three different schemes are compared; A standard LT code without feedback using the Robust Soliton distribution (RSD), an LT code with feedback using the RSD and an LT code with feedback using $\rho(i)$, where $\pi(j)$ is the RSD. The RSD is the de facto standard degree distribution for LT codes and was originally proposed in [1]. We use it with parameters $c = 0.1$ and $\delta = 1$, since these have been found to provide the smallest average overhead in [7]. For the schemes utilizing feedback, it is assumed that each individual input symbol is ACK'ed immediately after it has been decoded. For all schemes $k = 1000$. Fig. 1 shows the results. It is seen that when applying ACKs without changing the original degree distribution, the performance degrades significantly. Thus, the negative effect from ACK's by far outweighs the positive effect. However, when we eliminate the negative effect through an adaptive original degree distribution, we see a performance increase compared to the standard LT code with no feedback.

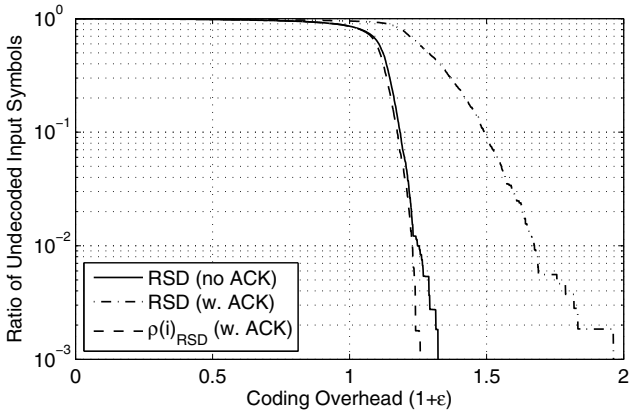

Fig. 1. The relative number of undecoded input symbols as a function of the amount of received symbols for the simulated schemes.

The gain from applying ACKs is quite small though. The explanation is that an LT code initially builds up potential and then eventually releases this potential. As a result, few input symbols are decoded early and then suddenly, within relatively few new received encoded symbols, all remaining input symbols are decoded. This is referred to as *avalanche decoding* [8]. It leaves a narrow window in which $\pi'(0)$ is significant, and since the gain from ACKs lies in the reduction of this probability, the performance improvement is negligible. We can thus conclude that standard LT codes using the RSD cannot benefit significantly from ACKs.

Motivated by this, we turn our attention to multi-layer LT codes [6]. Here data is decoded in stages, i.e. layers are decoded at different times, which is desired in e.g. video streaming. It leaves a longer window for the use of ACKs, which potentially increases the gain.

B. Multi-Layer LT Codes

Initially, we derive the probability of receiving symbols of reduced degree zero in the multi-layer LT code, as done in the single layer case. We then show how a single ACK can significantly decrease the overhead in a multi-layer LT code.

We use the approach to layering proposed in [6], where the uniform distribution used for selection of input symbols is replaced by a distribution which favors symbols from more

important layers. The k input symbols are divided into N layers, $s_1, s_2, \ldots, s_N$, each with size $\alpha_1 k, \alpha_2 k, \ldots, \alpha_N k$, where $\sum_{j=1}^{N} \alpha_j = 1$. We define the vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$. The probability of selection associated with input symbols from s_j is denoted $p_j(k)$, such that $\sum_{j=1}^{N} p_j(k)\alpha_j k = 1$ and $p_i(k) \geq p_j(k)$ if $i < j$. Note that if $p_j(k) = \frac{1}{k}\ \forall j$, then all data is treated equally, as in the single-layer LT code. We define a vector, $\boldsymbol{\beta}$, where $\beta_i = \frac{p_i(k)}{p_N(k)}$.

An encoded symbol of the multi-layer LT code can be seen as having N dimensions, one for each layer. The N-dimensional original degree is denoted $\mathbf{j}$, where j_n denotes the number of neighbors belonging to the n'th layer. Correspondingly, we can denote the reduced degree $\mathbf{i}'$, where i'_n denotes the number of undecoded neighbors belonging to the n'th layer. Moreover, L_n denotes the number of undecoded input symbols from the n'th layer. We also define $\hat{i}' = \sum_{n=1}^{N} i'_n$, and similarly $\hat{L}$ and $\hat{j}$. By $\mathcal{J}_i$, we denote all $\mathbf{j}$ which satisfy $j_n > i'_n$, $n = 1, 2, \ldots, N$, and $\hat{j} = i$. The N-layer reduced degree distribution can now be expressed as follows.

Lemma 4. *(N-Layer Reduced Degree Distribution) Given that the encoder applies $\pi(i)$ in an N-layer LT code with parameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the reduced degree distribution is found as*

$$\pi'(\mathbf{i}') = \sum_{i=\hat{i}'}^{\hat{i}'+k-\hat{L}} \left(\pi(i) \sum_{\mathbf{j}\in\mathcal{J}_i} \left(\Phi(\mathbf{j}, i, \boldsymbol{\alpha}k, \boldsymbol{\beta}) \prod_{n=1}^{N} \frac{\binom{L_n}{i'_n}\binom{\alpha_n k - L_n}{j_n - i'_n}}{\binom{\alpha_n k}{j_n}} \right) \right)$$

$$\text{for} \quad L_n < i'_n \leq \alpha_n k, \quad n = 1, 2, \ldots, N,$$

$$\pi'(\mathbf{i}') = 0 \quad \text{elsewhere.}$$

where Φ is Wallenius' noncentral hypergeometric distribution.

Proof: This proof follows the proof of Lemma 1, although with the change that the non uniform sampling of input symbols in the encoder is taken into account. This is done with a multivariate version of Wallenius' noncentral hypergeometric distribution. This describes the distribution of $\mathbf{j}$, given a total sample size of i drawn from a population of size k, which is divided into groups of sizes $\boldsymbol{\alpha}k$, whose relative selection probabilities are given by $\boldsymbol{\beta}$. By summing over all $\mathbf{j} \in \mathcal{J}_i$, we consider all ways the i neighbors can be distributed among the N layers, while requiring that $j_n > i'_n$, $n = 1, 2, \ldots, N$, such that a degree reduction to $\mathbf{i}'$ is possible. The multiplication with N hypergeometric distributions accounts for the probability of a degree reduction from j_n to i'_n, $n = 1, 2, \ldots, N$.■

We now investigate the case of $N = 2$, where we refer to the layers as base layer and refinement layer. The reduced degree can be viewed as two-dimensional, thus the probability of redundancy is denoted $\pi'(0, 0)$. In Fig. 2, $\pi'(0, 0)$ is plotted as a function of L_B and L_R, the number of undecoded symbols from the base layer and the refinement layer, respectively. The parameters, $\alpha_B = \alpha_R = 0.5$, $\beta = \frac{p_B}{p_R} = 9$ and $k = 100$ have been chosen. An optimized degree distribution is not provided in [6], thus we use the RSD. The plot shows that $\pi'(0, 0)$ increases faster for decreasing L_B than for decreasing L_R. In [6] it was shown that the multi-layer LT code provides the unequal recovery time property, which essentially means that

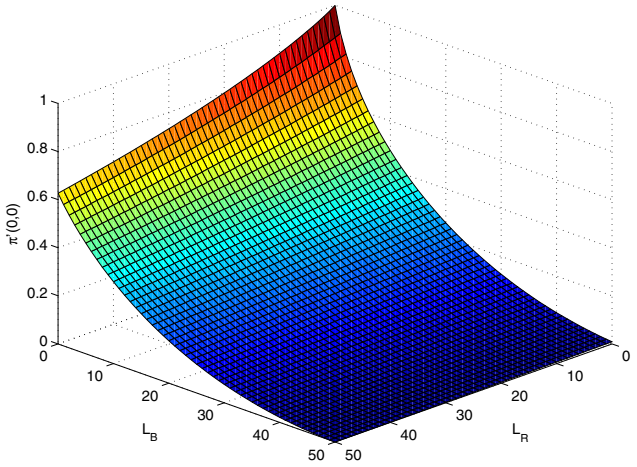

Fig. 2. The probability of reduced degree zero, for a two-layer LT code with parameters $\alpha_B = \alpha_R = 0.5$, $\beta = 9$ and $k = 100$.

L_B decreases to zero earlier than L_R during decoding. In the avalanche analogy, this means that we have two avalanches instead of one. With respect to Fig. 2, it means that the probability of redundancy is found along a path which starts at $L_B = L_R = 50$, then moves to a point with $L_B = 0$ and L_R still close to 50, and finally follows the edge of the surface towards $L_B = L_R = 0$. Note the high probability of redundancy during the second avalanche. This is a significant drawback of the multi-layer LT code, i.e. that decoding of refinement layers is inefficient.

We propose to eliminate this drawback through ACKs. From the single-layer analysis, we learned that ACKs during an avalanche is not very helpful. However, in this case decoding evolves as two avalanches separated in time. This gives an opportunity to apply ACKs efficiently. **The strategy is to ACK the base layer in its entirety, when it has been decoded, such that only refinement layer symbols are considered in future encoding.** This only requires a single feedback message, which makes this scheme practical, as opposed to the single-layer counterpart discussed in section III-A. It also has a clear advantage over the approach called focused checking from [9], where LLRs on individual input symbols are fed back, thus requiring a much stronger feedback channel. Another competitor of the proposed scheme is the simple approach where one applies two separate LT codes, one for each layer. First the base layer is transmitted only, and when ACK'ed, the refinement layer is transmitted. However, when the scenario is extended to multiple users, which is ultimately the goal, it is necessary to be able to serve multiple needs. One user might be decoding the base layer, while another user has moved on to the refinement layer. Also note that this simple scheme is actually a special case of our proposed scheme, where $\beta = \infty$.

A simulation of the multi-layer LT code has been performed. The two-layer LT code from the analysis is applied, although here with $k = 1000$. The code with and without ACK of the base layer are simulated. Moreover, the single-layer LT code is

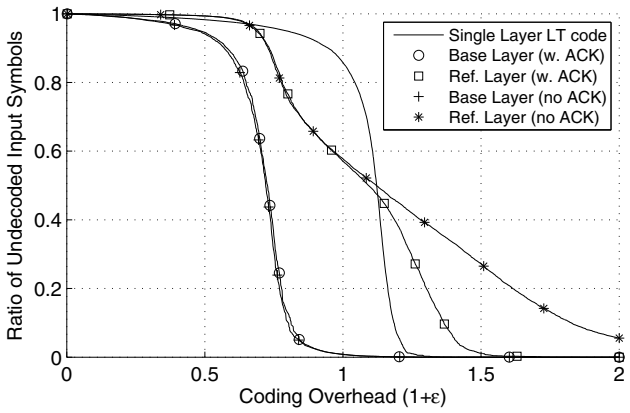

Fig. 3. The relative number of undecoded input symbols from the individual layers as a function of the amount of received symbols.

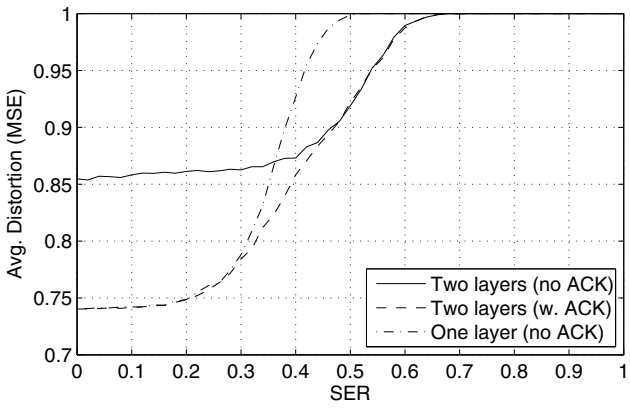

Fig. 4. Average distortion as a function of the symbol error rate.

included in order to show the effect of multiple layers. Fig. 3 shows the results. It is clear from the figure that the decoding of the refinement layer is inefficient when ACK is not applied. Applying the ACK yields a very significant decrease of the necessary overhead. It can thus be concluded, that the positive effect from ACK'ing in the applied two-layer LT code by far outweighs the negative effect. This is in contrast to the results for the single-layer LT code.

IV. Numerical Results

A simulation has been implemented in Matlab with the purpose of evaluating the investigated schemes. Since we consider multi-layer codes, where it is possible to decode at different rates, we will evaluate with respect to a distortion measure. We do not simulate the distortion using actual data. Instead we apply the theoretical lower bound from rate-distortion theory [10], which says that for an i.i.d., zero-mean, unit-variance Gaussian source the distortion-rate function is $d(r) = 2^{-2r}$. We assume a lossy source code with achievable rates r_z, $z = 0, 1, 2$, where z denotes the number of decoded layers. The values of r_z are $r_0 = 0$, $r_1 = 0.271\alpha$ and $r_2 = 0.271$. The decoding deadline of each data block is the equivalent of the transmission of $2k$ symbols. The RSD is applied in all schemes, and the parameters are $k = 100$, $c = 0.1$, $\delta = 1$, $\alpha = 0.5$ and $\beta = 9$. The transmission channel is modeled as a packet erasure channel, where erasures occur at symbol level at a rate denoted by SER.

Fig. 4 shows a plot of the distortion averaged over 100 data blocks as a function of SER. At low SER and high SER the two-layer LT code with ACK and the single-layer LT code performs similarly. However, at SER between roughly 0.3 and 0.6, the two-layer code has a significantly better performance than the single-layer code. The impact of the ACK is evident at low SER. The high probability of redundancy makes it unlikely that the refinement layer is ever decoded. The ACK clearly remedies this problem. This confirms the results in the analysis, that the inefficient decoding of the refinement layer can be significantly improved through a single ACK. Applications like audio and video streaming can potentially benefit from this. If a feedback opportunity occurs, this can be utilized to increase the probability of receiving higher quality media.

V. Conclusions

In this paper we have analyzed the impact of feedback in the form of acknowledgments in LT codes. Both standard single-layer LT codes and multi-layer LT codes with unequal error protection have been investigated. Results show that acknowledgments only hold a significant potential in multi-layer LT codes. Analysis shows that the probability of redundancy is a much bigger problem in these codes, mainly during decoding of lower prioritized layers. Using only a single feedback message, we managed to improve the performance of a two-layer LT code remarkably. Simulations also show significant improvements compared to the single-layer LT code.

References

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

[2] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory.*, pp. 2551–2567, 2006.

[3] S. Kokalj-Filipović, P. Spasojević, E. Soljanin and R. Yates, "ARQ with Doped Fountain Decoding," in *ISSSTA '08. IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, 2008.*, pp. 780–784, 2008.

[4] A. Beimel, S. Dolev and N. Singer, "RT oblivious erasure correcting," *IEEE/ACM Transactions on Networking.*, pp. 1321–1332, 2007.

[5] A. Hagedorn, S. Agarwal, D. Starobinski, and A. Trachtenberg, "Rateless Coding with Feedback," in *INFOCOM 2009, IEEE*, pp. 1791–1799, April 2009.

[6] N. Rahnavard, B. N. Vellambi and F. Fekri, "Rateless Codes With Unequal Error Protection Property," *IEEE Transactions on Information Theory vol. 53.*, pp. 1521 – 1532, April 2007.

[7] F. Uyeda, H. Xia and A.A. Chien, "Evaluation of a High Performance Erasure Code Implementation," *Technical Report, University of California, San Diego*, 2004.

[8] D. MacKay, "Fountain codes," *Communications, IEE Proceedings-*, vol. 152, pp. 1062 – 1068, December 2005.

[9] A. Oka and L. Lampe, "Data Extraction from Wireless Sensor Networks Using Distributed Fountain Codes," *IEEE Transactions on Communications, vol. 57, no. 9*, pp. 2607–2618, 2009.

[10] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *IRE National Convention Record part 4.*, pp. 142–163, 1959.