Study of Second-Order Memory Based LT Encoders

Luyao Shang
Department of Electrical Engineering & Computer Science
University of Kansas
Lawrence, KS 66045
lshang@ku.edu

Faculty Advisor:
Erik Perrins

ABSTRACT

LT encoder design has always been a basic and crucial topic ever since the development of fountain codes. The memory based LT encoders (MBLTEs) aim at further improving the code performance by managing the input symbols' degree distribution after an output symbols' degree distribution is determined. Previous work has shown that the MBLTE has a faster decoder convergence and a lower bit error rate (BER) than the regular LT encoder with the belief propagation (BP) decoder over binary erasure channels (BECs). However, the study of MBLTEs is limited to first-order MBLTEs, higher-order MBLTEs have not been investigated yet. Therefore, in this paper we study the second-order MBLTE, and propose an algorithm for its realization. Simulation results show that our encoder outperforms the first-order MBLTE in terms of the BER. Our proposed second-order MBLTE performs better either with a short code or with a high erasure probability.

INTRODUCTION

In a multicast or broadcast communication system, a vast number of autonomous receivers with a traditional feedback scheme can cause a feedback implosion. In order to solve this problem, fountain codes [1, 2, 3, 4] were proposed as a solution. Like a digital fountain, the receiver doesn't care which particular codeword is received, it only cares about the total number of received codewords. As long as a certain overhead is reached, the decoder can fully recover the source data.

A fountain code has the characteristics of small overhead and flexibly adjustable rate, and thus has a wide range of applications. In broadcasting systems, where a large file is broadcast to a wide set of receivers, fountain codes show good performance without a great amount of feedback. In fact, 3GPP multimedia broadcast/multicast service has adopted fountain code for television delivery over IP networks [5, 6]. In cooperative cognitive networks, fountain codes are proposed as good codes to be used with mutual-information accumulation [7]. In addition, their flexible code rate means they are also applicable in hybrid ARQ schemes.

The first practical realization of fountain codes is the Luby transform (LT) code, which was proposed by Luby in 2002 [8]. In order to keep computational complexity low, the LT encoder employs a sparse generator matrix so that its output can be efficiently decoded by the belief propagation (BP) algorithm [9]. In order to obtain the generator matrix for source data with length K, we first sample a probability distribution Ω to obtain a degree of d. Then we sample a vector $x \in {F_2}^K$ of weight d uniformly at random to

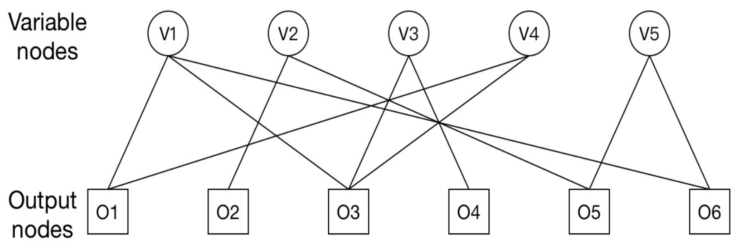

Figure 1: Graphical representation of $\mathbf{G}$.

obtain a column of the generator matrix. The pair (K, Ω) forms the parameters of the LT code and Ω is called the corresponding degree distribution for the output symbols.

The LT code belongs to a category of codes called "codes on graphs," which makes the BP algorithm the most widely used decoding algorithm for LT codes. The BP decoder operates on the bipartite graph, which is made of nodes and edges. For a LT code with K input symbols and N received output symbols, its graph consists of K variable nodes and N output nodes as well as the edges in between. The variable nodes are also called the left nodes and the output nodes are called the right nodes. Each variable node represents an input symbol and each output node represents an output symbol. The two kinds of nodes are connected by edges. The BP decoder starts from a degree-one output node. Since a degree-one output node only has one neighboring variable node, this variable node is recovered. After its recovery, more degree-one output nodes are released. In general, messages are passed between the two kinds of nodes repeatedly until no degree-one output node can be found, or until all the edges are removed. An example generator matrix is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{1}$$

and the graphical representation of this generator matrix is shown in Figure 1.

With the BP decoder, the degree distribution Ω has a significant impact in the design of the LT code. From the description of the working scheme of the BP decoder above, we can see that the decoder will encounter an early termination if no degree-one output nodes are found before its completion. Also, the decoder will encounter a low efficiency problem if too many degree-one output nodes are released at each round. Therefore, a good distribution is needed to ensure that the decoder can proceed to completion while maintaining its efficiency. Beginning with Luby's proposal of the ideal soliton distribution (ISD) and the robust soliton distribution (RSD) [8], researchers have made numerous contributions to the problem of degree distributions. In [10], the degree distribution was optimized for some specific lengths of source data, K. In [11], two optimized degree distributions were proposed for short codes: one is by minimizing the average number of packets that are needed for a successful decoding, and the other is by maximizing the probability of decoding a message with exactly n sent packets. In [12], the authors proposed a suboptimal degree distribution and showed that it outperforms the RSD in terms of overhead. In [13], the authors proposed a new ripple design criterion and showed that the overhead can be reduced by varying the ripple size.

Although lots of work has been done on the design of the degree distribution of the LT code, the majority of this work has focused on the output symbols only, leaving the input symbols as being Poisson distributed. Hussain was the first to study the degree distribution of the input symbols. Instead of uniformly selecting d input symbols to be connected to the degree-d output symbol, Hussain proposed an algorithm [14] to select the d input symbols so that the final degree distribution of the input symbols is regular. Simulation results showed that this left-regular LT code provides a lower error floor than previous codes.

Recently, the authors in [15] proposed a memory based LT encoder (MBLTE) which was shown to be better than Hussain's in terms of the bit error rate (BER) and decoding convergence speed; this improved code has a memory-order of one. Motivated by the promising initial results for first-order MBLTEs, we focus our attention on second-order MBLTEs. Our simulation results show that the proposed second-order MBLTE exhibits a better BER performance than that of the first-order MBLTE. The simulation results also show that our MBLTE has a smaller overhead than the first-order MBLTE for short codes, which are widely used in sensor networks and delay sensitive communication systems such as video on demand (VOD) services.

The rest of the paper is organized as follows. The next section describes the system model of LT codes with their encoding and decoding processes, including the RSD degree distribution for the output symbols that will be used throughout the paper. Following that we introduce the idea of memory based encoding and our proposed second-order MBLTE. We then give numerical simulation results and our final conclusions.

SYSTEM MODEL

A. LT encoding

A LT encoder encodes a set of input symbols $s_1, s_2, \ldots, s_K$ into a set of output symbols t_1, t_2,..., t_N, where K and N are the lengths of the input and output sequences, respectively. Each output symbol t_n is obtained by adding some selected input symbols in modulo-2. The operation is performed as below:

$$t_n = \sum_{k=1}^{K} s_k G_{kn} \tag{2}$$

where G_{kn} is an element of the generator matrix. The generator matrix $\mathbf{G}$ is obtained column by column. Each column represents a selection of the input symbols, which are used to generate an output symbol. The selection of the input symbols depends on the output symbols degree distribution Ω. To generate each output symbol t_n, the encoder goes through the following procedure [4]:

1. Randomly choose the degree d_n from a degree distribution Ω.
2. Uniformly choose d_n rows at random in the n-th column of $\mathbf{G}$ in which to place a value of "1."

B. LT decoding

For a binary erasure channel (BEC) with erasure probability ϵ, the BP decoder starts to decode when N output symbols are received. The decoding process is summarized below [4]:

1. Find an output symbol t_n that is connected to only one input symbol s_k. If there is no such output symbol, the decoding process stops. If there is, go to Step 2.

2. Set the input symbol equal to the output symbol: s_k=t_n.
3. Find other output symbols t_i that are connected to s_k.
4. Add s_k to each t_i modulo 2.
5. Remove all the edges connected to s_k.
6. Return to Step 1.

C. RSD

The RSD [8] is the degree distribution of the output symbols we will be using throughout this paper. To obtain the RSD, we must first obtain the ISD [8]:

$$\rho(1) = 1/K \tag{3}$$
$$\rho(d) = \frac{1}{d(d-1)} \quad \text{for d = 2, 3, ..., } K. \tag{4}$$

The RSD is obtained by modifying the ISD: define δ as the bound on the probability that the decoder fails after K' symbols have been received. The expected number of degree-one output symbols throughout the decoding process is:

$$S \equiv c \ln(K/\delta)\sqrt{K} \tag{5}$$

where c is a free parameter with a value smaller than 1.

Define a positive function:

$$\tau(d) = \begin{cases} \frac{s}{K}\frac{1}{d}, & \text{for } d = 1, 2, \ldots, (K/S) - 1 \\ \frac{s}{K}\log(s/\delta), & \text{for } d = K/S \\ 0, & \text{for } d > K/S. \end{cases} \tag{6}$$

Then the RSD can be obtained by adding the ISD to τ and then normalizing it:

$$\Omega_d = \frac{\rho(d) + \tau(d)}{Z} \tag{7}$$

where Z is a normalization scalar such that $Z = \sum_d [\rho(d) + \tau(d)]$. Thus, the degree distribution for the output symbols is

$$\Omega(x) = \sum_d \Omega_d x^d. \tag{8}$$

The RSD allows the LT code to decode successfully after receiving $K' = KZ$ output symbols with a probability at least $1 - \delta$.

MBLTE Design

D. Degree distribution of input symbols

For most cases, "degree distribution" refers to the degree distribution of the output symbols. It is often neglected that it can also refer to the one of the input symbols. For regular LT codes, when the degree

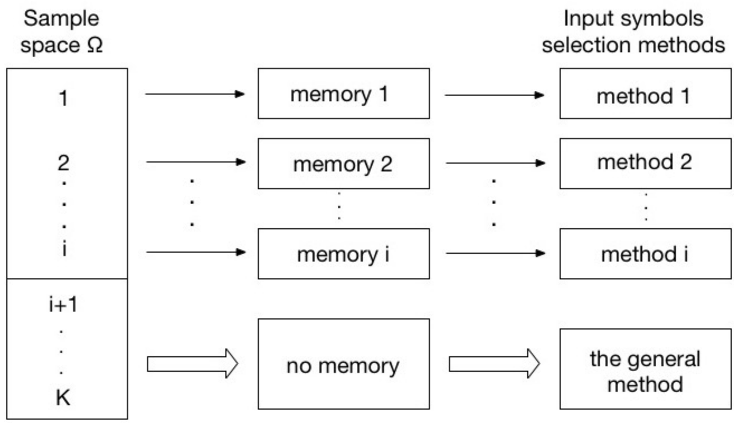

Figure 2: The memory-order-i MBLTE.

distribution of the output symbols is fixed, the input symbols are chosen in a uniform manner for each output symbol. The probability of an input symbol to be of degree-of-l is:

$$\Delta_l' = \binom{N}{l} p^l (1-p)^{N-l} \tag{9}$$

where N is the number of the output symbols, and p is the probability that an input symbol is chosen by an output symbol:

$$p = \frac{\bar{d}}{K} \tag{10}$$

$\bar{d}$ is the average degree of the output symbols:

$$\bar{d} = \sum d\Omega_d \tag{11}$$

We can see that equation 9 becomes poisson when N goes to infinity. For finite-length LT codes, the degree distribution of the input symbols can be better designed from poisson. Introducing memory to the encoder is one of the methods to manage the degree distribution of the input symbols. In this method, the input symbols are not chosen in a uniform manner anymore, instead, it follows different selection methods according to the degree of the corresponding output symbol.

E. concept of MBLTEs

The MBLTE differs from the regular encoder in the selection method of the input symbols. For both encoders, they need first to sample the degree distribution Ω to obtain a degree of d. A regular LT encoder without memory will then select those d input symbols uniformly at random, while a MBLTE does this in a different way. For a MBLTE, it remembers certain previous outcomes of d. With different outcomes of d, the selection methods are different. We use the term "memory order" to refer to the number of special outcome types a MBLTE remembers, i.e. a memory-order-i MBLTE remembers i outcome types.

Figure 2 shows the working scheme of a memory-order-i MBLTE. It remembers the outcomes with values of d from $d = 1$ up to $d = i$. When $d = j$ with $1 \le j \le i$, the encoder points to "memory j" and further to "selection method j." It then follows this method in selecting the input symbols. If $j > i$, the encoder points to "no memory" and it proceeds with sampling the input symbols by the general method, which is uniform sampling. Algorithm 1 details the encoding process of the memory-order-i MBLTE.

Algorithm 1 The memory-order-i MBLTE encoding process.

```
1: for $n = 1 : N$ do
2:     sample $\Omega$ to get $d = j$
3:     if $j \leq i$ then
4:         for $kk = 1 : j$ do
5:             select an input symbol by selection method $j$
6:         end for
7:     else
8:         for $kk = 1 : j$ do
9:             select an input symbol by the general selection method
10:        end for
11:    end if
12: end for
```

F. First-Order MBLTE

Hayajeh was the one who proposed the first-order MBLTE [15]. This encoder remembers only one outcome type, which is $d = 1$. When $d = 1$, the encoder does not select an input symbol at random (i.e. uniform sampling), but instead selects the input symbol with the highest instantaneous degree, without replacement. This selection method is referred to as "method 1" in Figure 2. By connecting the degree-one output symbol to the input symbol with the highest instantaneous degree, the decoding graph is more connected. The direct consequence is that more degree-one output symbols will be released after the first decoding round. Also, this method follows the decreasing ripple size criterion that was proposed in [13].

G. Second-Order MBLTE

In order to keep releasing more degree-one output symbols during the early decoding rounds, we propose a second-order MBLTE. In addition to $d = 1$, this second-order MBLTE also remembers the outcome type of $d = 2$. When $d = 1$, it selects the input symbols following the same selection method as in the first-order MBLTE, i.e. "method 1." When $d = 2$, it follows selection "method 2," which is to select the first input symbol from those symbols with the highest instantaneous degree, then to select the second input symbol from those symbols with the second highest instantaneous degree. While these selection methods are *ad hoc*, their motivation is to release a large number of new degree-one output symbols after the first and the second decoding rounds. In this way, the ripple size keeps large during the first two decoding rounds and then decreases during the entire decoding process. The details are shown in Algorithm 2.

SIMULATION RESULTS

In this section we study the performance of the second-order MBLTE. The baseline comparison is the first-order MBLTE in [15]. The bit error rate (BER) and frame error rate (FER) are used to evaluate the code performance.

Figures 3 and 4 show the BER and the FER performance, respectively, of the MBLTEs. For both figures, the code block lengths are fixed at $N = 512$. The source data length K varies from 256 to 512.

Algorithm 2 The second-order MBLTE encoding process.

```
1:  for $n = 1 : N$ do
2:      sample $\Omega$ to get $d = j$
3:      if $j = 1$ then
4:          select the input symbol with the highest instantaneous degree without replacement, then put
5:          this input symbol into set $S_1$
6:      else if $j = 2$ then
7:          first select an input symbol from the set $S_1$ uniformly at random with replacement; then select
8:          an input symbol with the highest instantaneous degree except for those in set $S_1$ without replac-
9:          ement
10:     else
11:         for $kk = 1 : j$ do
12:             select an input symbol uniformly at random without replacement
13:         end for
14:     end if
15: end for
```

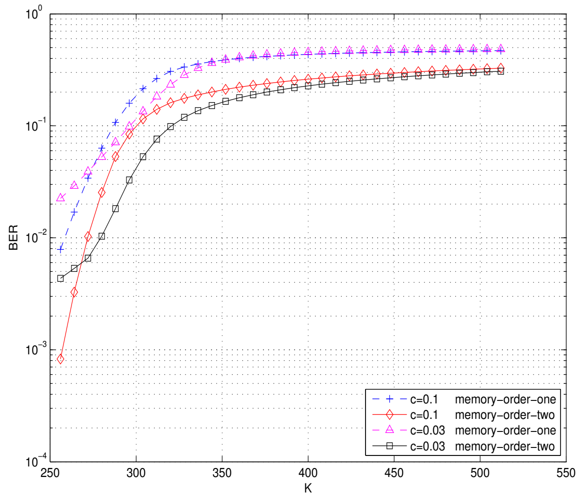

Figure 3: BER performance for a variety of data lengths.

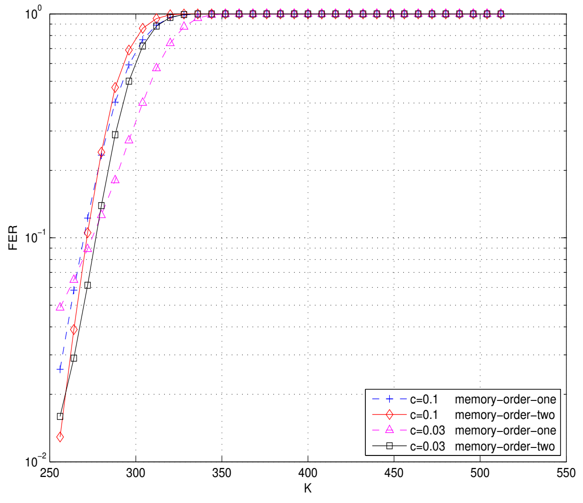

Figure 4: FER performance for a variety of data lengths.

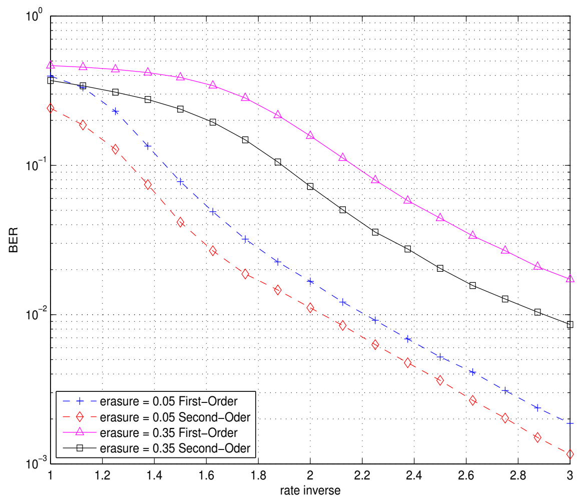

Figure 5: BER performance for $K = 64$.

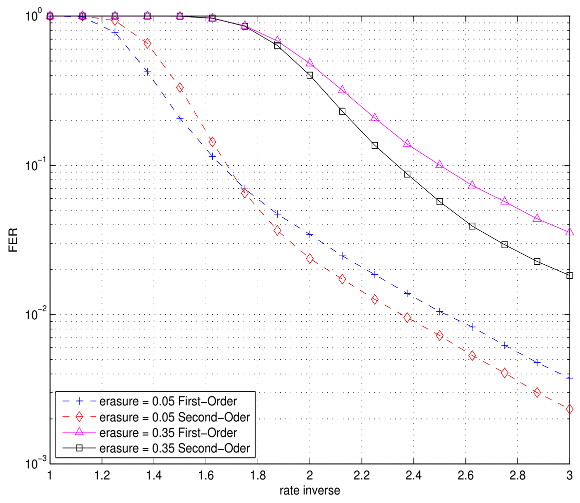

Figure 6: FER performance for $K = 64$.

Two common parameter settings of the RSD are used: $c = 0.1$ and $c = 0.03$, both with $\delta = 0.5$. The channel used in the figures is the binary erasure channel (BEC) with erasure probability $\epsilon = 0.3$.

In Figure 3, we see that for each of the RSD settings, the second-order MBLTE has a lower BER than the first-order MBLTE. For $c = 0.1$, the BERs of both MBLTEs decrease as K decreases. When K decreases to $K \approx 300$, both BER curves start to decrease rapidly, but the curve for the second-order MBLTE is steeper than that of the first-order MBLTE. This happens also for the $c = 0.03$ case. From these curves we can see that to decrease the BER, decreasing the code rate of the second-order MBLTE is more effective than that of the first-order MBLTE.

In Figure 4 we see that for $c = 0.1$, the FER of the second-order MBLTE is lower than that of the first-order MBLTE when K is small. As K increases, the gap between the two curves narrows and they eventually cross each other. Beyond point, the FER of the second-order MBLTE is higher than that of the first-order MBLTE. Although this is the case, the FER of both MBLTEs rapidly approaches 1 as K increases. The FERs in the case of $c = 0.03$ show the same behavior as with $c = 0.1$. Comparing Figure 4 with Figure 3, we can see that when a frame is in error, the second-order MBLTE has fewer bit errors than the first-order MBLTE. This phenomenon is similar to the BER/FER behavior of turbo codes and low density parity check (LDPC) codes: once a frame is in error, the BER of turbo codes is typically low (usually 2% or 3%), while the BER of the LDPC codes is higher (usually around 10%).

Figures 5 and 6 show the BER and FER performance, respectively, of the MBLTEs at different coding rates with short codes. The parameters used in the simulations are: RSD with $c = 0.03$ and $\delta = 0.5$; data length $K = 64$; coding rate inverse varies from 1 to 2.5. Two BECs with erasure probabilities $\epsilon = 0.05$ and $\epsilon = 0.35$ are employed.

In Figure 5, in both BECs, the BERs of the second-order MBLTE are lower than the first-order MBLTE throughout the whole range of coding rates. Comparing $\epsilon = 0.05$ with $\epsilon = 0.35$, as the code rate decreases, BERs of the former decrease earlier than the latter. This is because codes with better channels can receive a sufficient amount of output symbols faster than worse channels. In addition to this, we can also observe that the gap between the MBLTEs with $\epsilon = 0.35$ is larger than that of $\epsilon = 0.35$, which suggests that the superior performance of second-order MBLTEs is more pronounced with higher erasure probability channels than with lower ones. This phenomenon can also be observed from Figure 6.

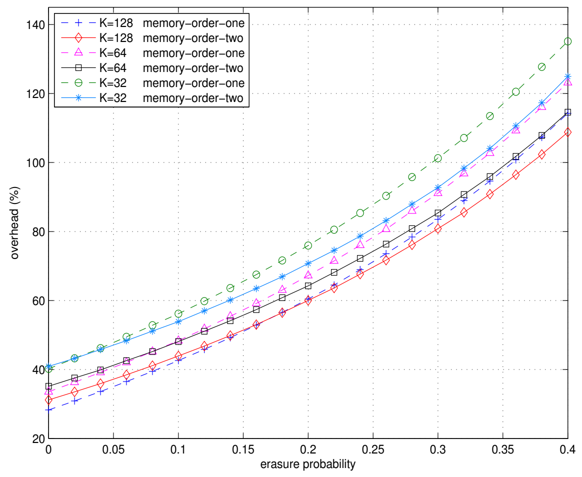

Figure 7: Overhead for a variety of erasure probabilities ϵ.

Figure 7 shows the MBLTE performance in terms of the overhead in a rateless setting. The RSD used in this figure is parameterized by $c = 0.1$ and $\delta = 0.5$. The BEC erasure probability is varied from $\epsilon = 0$ to $\epsilon = 0.4$. Three source data lengths are compared: $K = 128$, $K = 64$, and $K = 32$. For $K = 128$, as ϵ increases, the overhead of the second-order MBLTE is higher than that of the first-order MBLTE at first and then becomes lower. The intersection of the two curves appears at $\epsilon \approx 0.18$. For $K = 64$, the behavior is similar but the erasure probability for the intersection decreases to $\epsilon \approx 0.1$. For $K = 32$, the crossover erasure probability further drops to $\epsilon \approx 0.02$, meaning that the second-order MBLTE outperforms the first-order MBLTE almost everywhere. From these curves we see that, although the overhead increases as the code length decreases, the overhead of the second-order MBLTE increases slower than that of the first-order MBLTE. The second-order MBLTE always performs better either with a shorter code or a higher erasure probability.

CONCLUSIONS

In this paper, we reviewed the previous results of first-order MBLTEs and proposed an algorithm for second-order MBLTEs. We first reviewed the regular LT encoding and decoding processes, as well as the most frequently used degree distribution, RSD, of the output symbols. We then proposed the concept of MBLTEs and organized the previous work as the first-order MBLTE. Based on this concept, we further proposed the second-order MBLTE with a realization algorithm. Our simulation results showed that the second-order MBLTE outperforms the first-order one in terms of BER and overhead. The simulation results also show that the advantages of the second-order MBLTE are more obvious with either a shorter code or a higher erasure probability. Future work includes the mathematical analysis of this proposed MBLTE.

REFERENCES

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 56–67, 1998.

[2] J. Byers, M. Luby, and M. Mitzenmacher, “A digital fountain approach to asynchronous reliable multicast,”

Selected Areas in Communications, IEEE Journal on, vol. 20, pp. 1528–1540, Oct 2002.

[3] P. Maymounkov, “Online codes,” tech. rep., Technical report, New York University, 2002.

[4] D. MacKay, “Fountain codes,” *Communications, IEE Proceedings-*, vol. 152, pp. 1062–1068, Dec 2005.

[5] “Multimedia broadcast/multicast service (mbms); protocols and codecs,” 2012.

[6] T. S. A. Begen, “Guidelines for implementing digital video broadcasting - iptv (dvb-iptv) application-layer hybrid forward error correction protection,” *RFC 6683*, August 2012.

[7] A. Molisch, N. Mehta, J. S. Yedidia, and J. Zhang, “Performance of fountain codes in collaborative relay networks,” *Wireless Communications, IEEE Transactions on*, vol. 6, pp. 4108–4119, November 2007.

[8] M. Luby, “LT codes,” *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pp. 271–280, 2002.

[9] A. Montanari and R. L. Urbanke, “Modern coding theory: The statistical mechanics and computer science point of view,” *CoRR*, vol. abs/0704.2857, 2007.

[10] A. Shokrollahi, “Raptor codes,” *Information Theory, IEEE Transactions on*, vol. 52, pp. 2551–2567, June 2006.

[11] E. Hyytiä, T. Tirronen, and J. Virtamo, “Optimizing the degree distribution of LT codes with an importance sampling approach,” 2006.

[12] H. Zhu, G. Zhang, and G. Li, “A novel degree distribution algorithm of LT codes,” in *Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on*, pp. 221–224, IEEE, 2008.

[13] J. H. Sorensen, P. Popovski, and J. Østergaard, “Design and analysis of LT codes with decreasing ripple size,” *Communications, IEEE Transactions on*, vol. 60, no. 11, pp. 3191–3197, 2012.

[14] I. Hussain, M. Xiao, and L. K. Rasmussen, “Error floor analysis of LT codes over the additive white Gaussian noise channel,” in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1–5, IEEE, 2011.

[15] K. Hayajneh, S. Yousefi, and M. Valipour, “Improved finite-length Luby-Transform codes in the binary erasure channel,” *Communications, IET*, vol. 9, no. 8, pp. 1122–1130, 2015.