STUDY OF THE DECODING COMPLEXITY FOR RATELESS ERASURE CODES

Anghel BOTOŞ[1], Vasile BOTA[1], Aurel VLAICU[1]
[1]*Technical University of Cluj-Napoca*
Anghel.Botos@com.utcluj.ro, Vasile.Bota@com.utcluj.ro , Aurel.Vlaicu@com.utcluj.ro

***Abstract*-Over the Internet, bit errors within the data packets translate into packet losses at the higher layers of the OSI model, yielding a packet erasure channel. Modern erasure correcting codes promise to offer a very simple and efficient solution to data transfers over these channels, opening up also other interesting applications. Amongst them one can enumerate reliable large scale content distribution, high quality real-time data transfers, distributed storage and others. These considerations make the study of such codes an actual and interesting topic.Most of the analyses presented in literature focus on the evaluation of the performances ensured by these codes. This paper presents an evaluation of the decoding complexity of some rateless erasure codes, which is another relevant issue that affects the applicability of these codes. The complexities of several decoding methods are evaluated using several metrics which reflect different operations performed during the decoding. All results used in the evaluation are based on simulations.**

Keywords: *rateless erasure codes, computational complexity*

I. INTRODUCTION

Large-scale content distribution is an important aspect of the current Internet. Present technologies which offer such services usually rely on classical approaches to this issue, by breaking down the problem into several single source-single destination data transfers. These approaches may be suboptimal since the underlying network is thus not aware of the different nature of these transfers. The main effects of this approach may be unnecessary load in the network and poor service quality perceived by users. Furthermore, the Internet is being increasingly used over wireless channels, where non-negligible bit error probabilities lead to significant packet loss rates. Protocols like TCP, still extensively used, were designed when most links were wired and exhibited low packet error rates. As a result, these protocols perform poorly when used on wireless channels, providing lower throughput and greater delays.

Modern erasure correcting codes like the LT, Raptor or Online codes, described in [1][2][3][4] , promise to achieve efficient erasure mitigation over lossy channels. Besides efficient erasure mitigation, these codes can be used to improve the transmission quality for real-time data transfers, efficient information propagation in packet [5] and swarm networks [6], throughput improvement in wireless networks, distributed data storage [7][8] and others. Most of the studies presented in literature ([9] and others) focus on the performances provided by these codes, e.g. the probability of full recovery for a given overhead or other equivalent metrics. But another aspect that affects the applicability of these codes is the complexity, or the computational amount, involved by the algorithm employed for the decoding of these codes. This paper is a practical study of the decoding complexity of some of these modern rateless erasure correcting codes which can be used in content distribution technologies. Several decoding algorithms for these codes have been evaluated according to several complexity metrics, which are considered to be significant for the applications outlined earlier.

This paper is organized as follows: Section II reviews some theory regarding the codes used in the study, Section III gives details about the decoding algorithms and the complexity metrics used for the evaluation, Section IV presents the main results of the study, while Section V concludes the paper outlining the major insights gained from this study.

II. THEORETICAL FUNDAMENTALS

Most modern erasure correcting codes can be viewed as random linear codes over Galois fields. Let $F=(GF(2^w))^n$ be the set of all $w\times n$ bit length strings, for some positive non-zero integer values of w and n. The elements of this set can be viewed as n-dimensional vectors having elements from $GF(2^w)$ as coordinates for each dimension. In the following, consider $\Phi=(F,\oplus)$ the finite n-dimensional vector space over F induced by some addition operator $\oplus$. The input symbols for the rateless erasure code and the code-words at the output of the encoder are elements of Φ. Furthermore, choose $V=GF(2^w)$ as the set of scalars and introduce some multiplication operator, $\otimes$, such that $\Gamma=(\Phi, V, \otimes)$ forms a vector field over Φ.

Manuscript received September 8, 2011; revised November 28, 2011

The data that is to be encoded is partitioned into $K\in\mathbf{N}^*$ input symbols, denoted by $\mathbf{x}_i$, $i\in\{1,\dots,K\}$, $\mathbf{x}_i\in\Phi$, i.e. each symbol is $w\times n$ bits long.. If the length of the actual data is not an integer K times $w\times n$ bits, then it can be padded with"0" bits up to the next integer value of K. The encoder produces an output symbol $\mathbf{y}_j$ according to (1), by choosing randomly K elements a_{ji} from V, according to some distribution and by using the $\oplus$ and the $\otimes$ operators for the addition and multiplication:

$$\mathbf{y}_j=\sum_{i=1}^{K}a_{ji}\mathbf{x}_i \qquad (1)$$

This procedure can produce, from a set of K input symbols, a total number of $N_{max}=(2^w)$ encoded symbols, not all of them being linearly independent. Provided that K is usually set on the order of hundreds or thousands, the number of encoded symbols that can be produced from any set of input symbols is very large, hence the name *rateless erasure codes*. The random coefficients are chosen using a random number generator (RNG), which is also a parameter of the code.

Assuming that the decoder, receives the random a_{ji} coefficients together with each encoded symbol $\mathbf{y}_j$., the decoder needs at least $J\geq K\in\mathbf{N}^*$ encoded symbols to perform the decoding. Note that the correct operation of the decoder does not require these J symbols to be consecutively produced by the encoder. As a result, recovering the K input symbols amounts to solving the system of equations given in (2), formed by the decoder:

$$\mathbf{Y}=\begin{bmatrix}\mathbf{y}_1\\ \vdots\\ \mathbf{y}_J\end{bmatrix}=\begin{bmatrix}a_{11} & \dots & a_{1K}\\ \vdots & \ddots & \vdots\\ a_{J1} & \dots & a_{JK}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{x}_1\\ \vdots\\ \mathbf{x}_K\end{bmatrix}=\mathbf{A}\times\mathbf{X} \qquad (2)$$

The most straightforward method to perform decoding is Gaussian elimination. Section III presents other possible decoding methods for some of the codes.

Having a solvable system in (2) requires matrix **A** to have full rank, i.e. *rk[A] = K*. The probability that *rk[A] = K* depends on the size of the *GF* used, on the numbers of input symbols and received encoded symbols, and on the distribution used by RNG which generates the coefficients a_{ji}. If only $J=K$ encoded symbols are received at the decoder, then this probability is lower bounded by [5]:

$$1-\prod_{l=1}^{K}\left(1-\left(\frac{1}{2^w}\right)^l\right)\leq P_{rk[\mathbf{A}]=K}(K)\leq 1 \qquad (3)$$

Equation (3) shows that having received only K encoded symbols would not always ensure the correct recovery of all the input symbols. On the other hand, if the decoder is required to ensure the recovery of all input data with a higher probability, then the decoder will need to receive a number $J\geq K$ of encoded symbols.

The average overhead, needed by the decoder to recover all input symbols with a imposed (high) probability is denoted by ε., and so the average number of encoded symbols needed for successful recovery of all input symbols is $n=(1+\varepsilon)K$. The probability of this recovery, given an $\varepsilon\in\mathbf{R}^+$, can be made arbitrarily close to 1 by any of the following means: increase of K, increase of w or a careful design of the distribution used by the random generator at the encoder. There is no closed-form expression for the average overhead ε as a function of the design parameters of the code (i.e., K, w, random distribution), but its value can be determined by simulations, which is a part of the study carried out in this paper.

A straightforward idea is choose the scalars a_{ji} from $GF(2)$ and thus decreasing the computational complexity of the encoding and decoding processes, since the addition operation is a bitwise XOR of the two symbols, and a value of "1" for an a_{ji} coefficient indicates that the corresponding source symbol should be added into that encoded symbol. For codes using $GF(2)$, the *degree of an encoded symbol* denotes the number of nonzero a_{ji} coefficients in the matrix row corresponding to it.

A. *LT Codes*

LT codes, [2], use $V=GF(2)$ as the set from which the random coefficients are chosen. A uniform random distribution cannot be used in this case because it would yield very large values for the average code overhead ε, [10]. Therefore, their practical implementations use finely tuned distributions for the random generator, i.e. the *Ideal Soliton* and the *Robust Soliton* distributions [2], to provide good performances.

B. *Raptor Codes*

Raptor codes, [3], are rateless erasure codes such that any subset of $K(1+\varepsilon)$ encoding symbols are sufficient to recover the original K symbols with high probability.

The main idea behind Raptor codes is to require that only a *constant fraction* of the input symbols be recoverable with high probability. This raises a new problem, since all input symbols need to be recovered, not only a constant fraction. This issue is addressed by encoding the input symbols with a traditional erasure code, and then applying an appropriate LT-code to the new set of symbols, in a way that the traditional code would be capable of recovering all the input symbols even in the presence of a fixed fraction of erasures.

This code construction leaves a great degree of freedom in the choice of the constituent codes and their parameters. A good example of a practical Raptor code is described in [11].

C. *Online Codes*

On-line codes are *rateless, locally encodable codes* [4], which are characterized by two parameters, ε and q. ε determines the degree of sub-optimality; a message of size K blocks (symbols) can be decoded from $(1+\varepsilon)K$ output symbols, with an imposed high probability which is linked to ε by means of the q parameter. q also affects the success probability of the decoding algorithm. The algorithm may fail to reconstruct the original message with a probability of $(\varepsilon/2)^{q+1}$.

The overall structure of an On-line code has two layers. To encode a message, first a small number of *auxiliary symbols* are generated and appended to the original ones, thus forming a *composite message*. The composite message has the property that knowledge of any 1-(ε/2) fraction of its symbols is sufficient to recover the entire original message. Next, in the inner encoding, symbols are generated to form an infinite, rateless encoding of the composite message. These symbols are called *check symbols* because they serve as a kind of parity check during message reconstruction.

III. ALGORITHMS AND METRICS

A. Decoding Algorithms

The basic decoding algorithm for rateless erasure codes using GF (2) is given in [2] and is called *belief propagation* (BP). From a mathematical point of view, this algorithm is actually a method of solving a system of equations using substitutions. Although it has low computational complexity, , according to [2], this method is not optimal, since it may not be able to solve a given system of equations due to the fact that there are no encoded symbols of degree one, i.e encoded symbols that are exact copies of the source symbols, in the buffer of the decoder, even if the encoding matrix describing the system of equations is of full rank.

The optimal approach to solving such systems of equations is to use *Gaussian elimination* (GE). On the other hand, this method of solving a system of equations would be more computationally intensive.

The third decoding method combines the previous two, and is denoted by BPGE. For each encoded symbol that is received by the decoder several steps are performed. First, the decoder processes the encoded symbol using the BP method. If this operation has the result that a new input symbol was recovered, this effectively reduces the size of the encoding matrix that represents the encoded symbols in the buffer of the encoder. In this case, a GE decoding attempt is made on the remaining encoded symbols in the buffer, by using their corresponding encoding matrix. This method has the advantage that encoded symbols of degree one are directly recovered, and that the size of the encoding matrix is dynamically reduced as new source symbols are recovered. The relative performance of the three decoding methods will also be evaluated. For the rateless erasure codes using higher-order Galois fields only the GE method is applicable, since the probability to have encoded symbols that are simple copies of source symbols is very small.

B. Complexity Metrics

The complexity of the decoding algorithms summarizes all the operations which are needed in order to perform the successful decoding of a data block. The decoding process involves different types of operations, depending on the algorithm employed. As a result one cannot use a single metric to evaluate the complexity of these algorithms, but several are needed. The following metrics will be used to evaluate the decoding complexity of the codes and algorithms referenced in this paper:

- **Average number of symbol additions.** This metric represents the average number of symbol additions that need to be performed in order to decode a source symbol. For the LT, Raptor and Online codes this operation is a bit-wise XOR between two memory zones where encoded symbols are stored by the decoder. For random codes using higher-order Galois fields, this operation amounts to Galois field additions on groups of bits. Most Galois field software libraries are implemented in such a way, that the addition operation between elements is again the bit-wise XOR between Galois field elements. Such a software library was used for the implementation, therefore the computational cost of a symbol addition is the same, regardless of the order of the Galois field used for the code.
- **Average number of matrix row additions.** This metric represents the average number of matrix row additions that need to be performed by the decoder in order to decode a source symbol. Again, for the LT, Raptor and Online codes this operation is nothing more than a bit-wise XOR between the memory regions where the contents of the two rows of the encoding matrices used by the decoder, are stored. The previous considerations regarding codes using higher-order Galois fileds are valid in this case as well. As a result the computation cost of such an operation is the same regardless of the order of the Galois field used by the code.
- **Average number of matrix row exchanges.** This metric represents the average number of matrix row exchanges that need to be performed by the decoder in order to recover a source symbol. Since this data is stored in memory by using pointers, this operation implies only the exchange of the two pointers to the memory areas where the contents of these rows are stored.
- **Average number of symbol substitutions.** A symbol substitution represents the operations needed to eliminate a neighbor from the encoding vector/graph associated to a received encoded symbol. This operation appears only when decoding the rateless erasure codes which use GF(2) as the set for the scalar coefficients. The average number of symbol substitutions is the average number of such operations that need to be performed in order to recover one source symbol.

A unique global metric for the computational complexity of the decoding algorithm could be obtained using a weighted sum of the metrics described earlier. The weights for each term in this weighted sum should be chosen depending on the implementation chosen for the decoder and the architecture of the system within which the decoder operates.

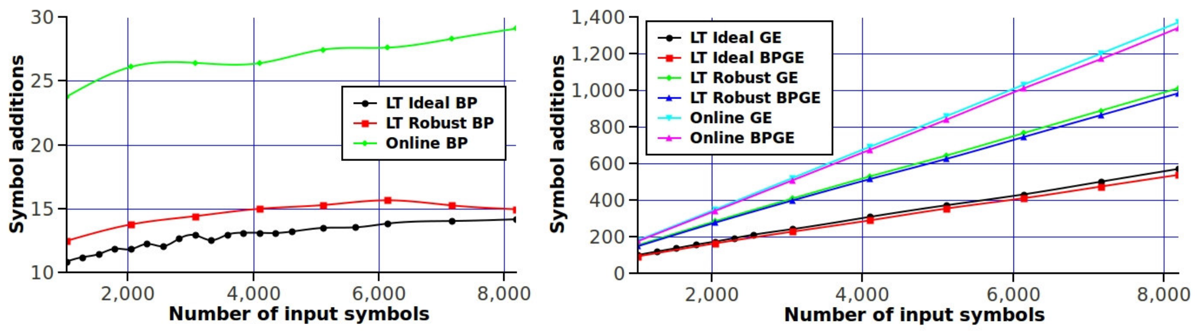

Figure 1. Average number of symbol additions for GF(2) codes.

IV. RESULTS

A. *Symbol Operations*

The main complexity metric for the decoding algorithms used with rateless erasure codes is the *average number of symbol additions* needed to recover an input symbol. It is the main metric because each such operation implies the addition of two encoded symbols. Encoded symbols may have any size depending on the application, but typical size may be on the order of kilobytes for real-time transfers up to one or several megabytes for swarm networks. As a result, a symbol addition is a costly operation from a computational point of view.

Average number of symbol additions for the codes using GF(2) are shown in Figure 1 ,where Ideal denotes the Ideal Soliton, and Robust denotes the Robust Soliton distribution.

The simple BP decoding algorithm, where applicable, exhibits the lowest complexity for all codes (LT Ideal, LT Robust, Online), requiring on average only 10-20 symbol additions to recover an input symbol. The increase of the number of required operations with the increase of the number of input symbols is very low and non-linear (logarithmic). The drawback of this decoding method is the fact that it offers very poor performance of these codes in terms of average overhead required to recover all input symbols.

Decoding methods based on Gaussian elimination (GE and BPGE) exhibit a linear increase of the complexity, which is significantly higher than for the BP decoding method, with the increase of the number of input symbols. The combination of the BP with the GE decoding method results in a slight decrease of the computational complexity of the GE-based algorithms. On the other hand, the implementation of this method requires a larger number of processor instructions, which may be an issue for small dedicated equipment.

Comparing the LT code using the Ideal Soliton distribution with the LT code built with the Robust Soliton distribution, one can notice that the complexity in the second case is about two times higher for all decoding methods. For the LT code, the average value of the distribution is directly related to the average degree of the encoded symbols. This average degree is in turn also directly tied to the average number of symbol additions needed to be performed by the decoder in order to recover one source symbol. For all decoding methods, the Online code exhibits the same type of dependency between the complexity and the number of input symbols, as the LT code. In absolute values, the decoding complexity of the Online code is significantly higher than for the LT codes.

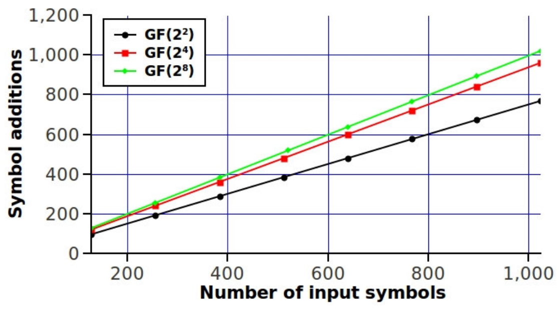

Figure 2. Average number of symbol additions for GF(2^m) codes.

The influence of increasing the dimension of the Galois field upon the average number of symbol additions for codes using higher-order Galois fields GF(2^m) is shown in Figure 2.

The codes using higher order Galois fields are decodable only using the GE method, which, in this case, exhibits the same linear dependency of the complexity with the number of input symbols. Their complexity is about 5 times higher for the same number of input symbols when compared to GF(2) codes, and also increases with the order of the GF(2^w) within which the code is constructed.. The advantage of these codes is the fact that they offer better performance in terms of average overhead required to ensure the high-probability recovery of all input symbols, compared to the codes built in GF(2) [9].

The average number of *symbol substitutions* required to recover a source symbol is presented in Figure. 3. This metric is shown only for GF(2) codes used in conjunction with a BP-based decoding method because this decoding method is the only one which makes use of this type of operation.

All codes exhibit an increase of this metric when the number of source symbols increases. For some of them (LT Robust BPGE, Online BPGE), this increase is linear, while for the others this increase is non-linear and has a logarithmic trend.

There should also be noted that the BP-decoded Online code requires more symbol substitutions than the other types of codes decoded with the same algorithm, because the resulting degree distribution used by this code has a higher average value than any of the Soliton distribution used by the LT code.

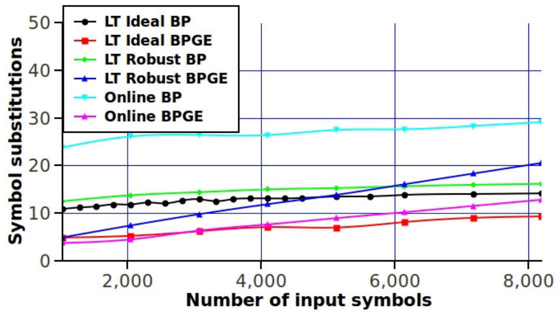

Figure 3. Average number of symbol substitutions for GF(2) codes.

Also, the employment of the BPGE decoding algorithm decreases significantly the amount of such operations for all codes. The combination between the BP and the GE decoding methods splits the entire computational between these two algorithms. Since the symbol substitutions are specific only to the BP decoding algorithm, it was somewhat expected that this metric would be lower for the combined algorithm compared to the simple BP algorithm.

B. *Matrix Operations*

All GE-based decoding methods make use of an encoding matrix which corresponds to the encoded symbols in the decoders buffer. These decoding methods also make use of matrix operations in order to perform the decoding.

The average number of *matrix row additions* required to recover a source symbol is shown in Figure. 4 for GF(2) codes.

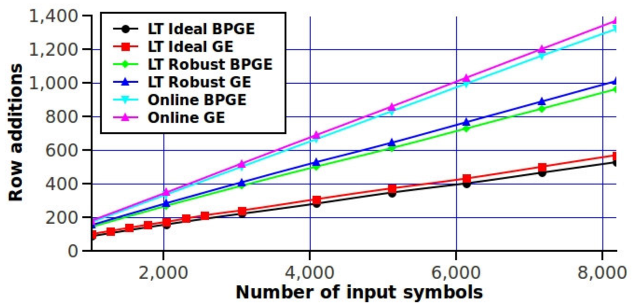

Figure 4. Average number of matrix row additions for GF(2) codes.

This metric was not recorded for higher-order GF codes, because they can be decoded only using the GE method. For this method, any symbol addition results also in a matrix row addition, and hence, the graphs, of. Figure. 2 right-hand and Figure 4, would be identical.

The dependency between this metric and the number of source symbols is again a linear one, with the number of matrix row additions being slightly smaller than the number of symbol additions for the same code using the BPGE decoding method.

The average number of *matrix row exchanges* required to recover an input symbol is shown in Figure. 5 for GF(2) codes. Decoding with the BPGE algorithm yields a linear dependency between this metric's values and the number of input symbols (LT Ideal BPGE has also a linear dependency, but with a low slope). The GE decoding method on the other hand, leads to constant values for this metric, roughly equal with the probability that a given position in the encoding matrix has a zero value for the a_{ji} coefficient.

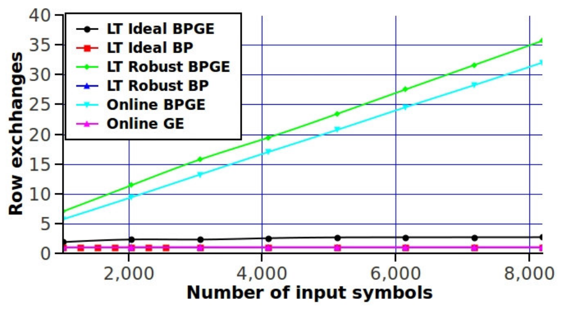

Figure 5. Average number of matrix row exchanges for GF(2) codes.

The average numbers of *matrix row exchanges* needed by higher-order GF codes to recover an input symbol are shown in Figure. 6.

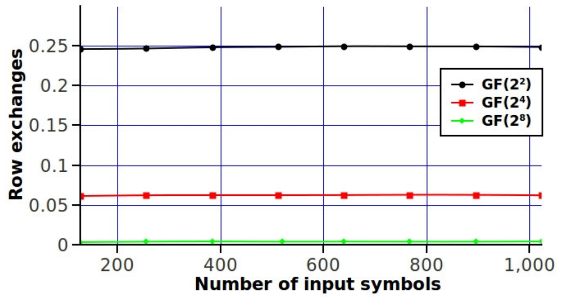

Figure 6. Average number of matrix row exchanges GF(2^m) codes.

The results of Figure.6 show that for higher-order GF codes, there is no dependency between the number of input symbols and the average number of matrix row exchanges needed to recover an input symbol. This metric is constant throughout the entire domain for the number of input symbols used in the study, and this constant value is approximately equal to $1/2^m$, thus favoring the codes which use the higher-order GF. A row exchange occurs when the algorithm needs a pivot for a given column from the encoding matrix, but the current row has a zero value in that column. As a result, the algorithm needs to find another matrix row that has a non-zero value in that column. Since these are constructed at random using a uniform distribution, the probability of having a zero value at any given location in the encoding matrix is $1/(2^m)$.

V. CONCLUSIONS

This paper analyzed the decoding complexity involved by the LT Ideal, LT Soliton and Online codes when decoded either with a Belief Propagation (BP), a Gaussian Elimination (GE), or a combined BPGE decoder. The metrics employed were the average numbers of various operations required to decode one input symbol.

The results obtained showed that the BP algorithm involves, for most of the considered metrics, a significantly lower computational complexity, than the algorithms using GE, but, as shown in [9], it also provides the worst performance in terms of average overhead. The combined BPGE algorithm requires slightly smaller amounts of computation than the „pure" GE algorithm.

The Online code is even more computationally demanding than both variants of the LT code. The additional computational load translates into a slightly better overhead performance, when compared to the LT code which uses the Robust Soliton distribution.

Codes constructed using higher-order Galois fields can be decoded only using the GE decoding algorithm. Compared to the other codes studied in this paper, these codes are more computationally intensive, requiring more operations, on average, in order to recover an input symbol. The increased computational load translates in this case also in a performance gain in terms of average overhead [9].

ACKNOWLEDGMENT

This work was funded by the Romanian Government and the European Union within the POSDRU/6/1.5/S/5 – "PRODOC" project.

REFERENCES

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," SIGCOMM Comput. Commun. Rev., vol. 28, no. 4, pp. 56–67, October 1998.

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

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

[4] P. Maymounkov and D. Mazieres, "Rateless Codes and Big Downloads," in In IPTPS'03, 2003.

[5] D. S. Lun, M. Medard, and M. Effros, "On Coding for Reliable Communication over Packet Networks," in *Proc. 42nd Annual Allerton Conference on Communication*, Control, and Computing, Sept.–Oct. 2004, invited, 2004.

[6] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in INFOCOM 2005. 24th Annual Joint *Conference of the IEEE Computer and Communications Societies*. Proceedings IEEE, vol. 4. IEEE, 2005, pp. 2235–2245.

[7] S. Acedanski, S. Deb, M. Medard, and R. Koetter, "How good is random linear coding based distributed networked storage," in In NetCod, 2005.

[8] R. G. Dimakis, V. Prabhakaran, and K. Ramch, "Decentralized erasure codes for distributed networked storage," *IEEE Transactions on Information Theory*, vol. 52, pp. 2809–2816, 2006.

[9] Anghel Botoş, Vasile Bota, Mihai P. Ştef, "Performance Evaluation of Rateless Erasure Correcting Codes for Content Distribution", unpublished, accepted for publication at RoEduNet 2011.

[10] D. J. C. Mackay, "Fountain Codes," *IEEE Communications*, vol. 152, pp. 1062–1068, 2005.

[11] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "*RFC5053 - Raptor Forward Error Correction Scheme for Object Delivery*," 2007.