The work presented here is a brief overview of some of the latest researches on fountain codes and their variants and how they progressively presented improvements over several points of view. The work is articulated like this: section 1 is a brief introduction on the main scenarios problems to takle and the solutions preceeding fountain codes; later, in section 2 a description of the digital fountain approach is depicted, with stress on the Tornado codes; section 3 goes deep into the concept of LT-codes, presenting their features and their strengh in terms of encoding probability distributions and Bit Error Rate (BER) performance; finally, in section 4, we give a description of the so-called Raptor codes, briefly analyzing their BER behavior and comparing it to the one of LT-codes.

1 Introduction

Multicast or broadcast transmission is an immediate solution for software companies that intend to spread new software over the Internet directed towards million of users. This is one of the several applications that this kind of transmissions can have nowadays, and this is motivated by the fact that currently there is an incresing proliferation of applications that must reliably distribute huge amount of data. For the nature of the problem, these transmissions have to be fully reliable, support a vast number of receivers and have a low network overhead. All these requirements together imply that retransmission of lost or corrupted packets is not feasible, as the clients requests for retransmission can rapidly overwhelm the server, with consequent *feedback implosion*. [1] These problems have led researchers to focus more on the application of erasure codes in Forward Error Correction (FEC) techniques for reliable multicast. Erasure codes are centered around the simple principle that the sender can transmit the k packets, which we could represent the original source data with, together with additional redundant packets that can subsequently be used to recover lost source data at the receiver. This receiver can reconstruct the original source data once it has received a sufficient number of packets and a big benefit in this is that different receivers can recover different lost packets using the same redundant data.

A good application of this concept can be seen in the figure of *digital fountain codes*, a better implementation than the one involving Reed-Solomon codes.[1]

2 The digital fountain

The digital fountain approach takes its name from the analogy with spraying water drops that are collected into a bucket. When the bucket is being filled by a fountain, we are not interested in knowing which droplets are falling into the bucket, as long as it is being filled. Once the filling part is done, the collection process ends and the content decoding will start. [2]

In tha same way, this approach injects a flow of distinct encoding packets into the network in such a way that the source data can be reconstructed entirely from any subset of them having the same length, in total, as the source data. A straightforward implementation of a digital fountain directly uses an erasure code that takes k-packets source data and produces as many encoded packets as the user demand needs. Thus, using for example the Reed-Solomon erasure code in such a way that the client side decoder can recontstruct the original source data whenerver k transmitted packets are received, the file is *stretched* into n encoding packets, with $\frac{n}{k}$ being the *stretch factor* or erasure codes. This technique comes with some side effects, one of which is the decreased efficiency of encoding and decoding operations: with the standard Reed-Solomon codes, for example, the behaviour is prohibitive even for small values of k and n. Hence, an alternative to these codes is proposed in [1], by using the so-called *Tornado codes*. In this codes, the equations have the form $y_3 = x_2 \oplus x_7 \oplus x_9$ where $\oplus$ represents the *bitwise* exclusive or and is used in place of the normal exclusive or $+$. On the other side, the decoding process of the Tornado codes uses two basic operations that are:

- replacing the received variables with theyr values in the equations they appear in;
- a simple *subsitution rule*;

Through these operations, Tornado codes avoid both the filed operations and the matrix inversion inherent in decoding Reed-Solomon codes. Furthermore, while the latter are built using a dense system of linear equations, the former are built using random *sparse* equations, and this brings to a significant reduction in decoding time. The price we pay for faster coding is that k packets are no longer enough to reconstruct the source data, translating into higher decoding inefficiency. In practice, however, the number of possible substitution rule applications stay minimal as far as slightly more than k packets are

Table 1: Properties of Tornado and Reed-Solomon codes

	Tornado	Reed-Solomon
Decoding inefficiency	$1+\epsilon$ required	klP
Encoding Times	$(k+l)ln(1/\epsilon)P$	klP
Decoding Times	$(k+l)ln(1/\epsilon)P$	klP
Basic Operation	XOR	Field operation

received, and often a single arrival generates a whirlwind of subsections that permit to recover all the remaining source data packets. Hence the name of Tornado codes. We can conclude that the advantage of these codes is that they are a tradeoff between a small increase in decoding inefficiency and a significant decrease of encoding and decoding times. [1]. 1 highlights the main propoerties of the Tornado codes, compared to the ones of Reed-Solomon codes.

3 The LT-codes

A further improvement in the field of erasure codes is given by the important calss of LT codes, ratless codes introduced and described by Luby in [3]. LT codes are the first realization of universal erasure codes, meaning that the codes symbol length can be arbitrary. Moreover, these codes are rateless, which means that the number of encoding symbols that can be generated from the data is potentially limitless.

Comparing LT codes to the preceding Tornado codes, it can be seen that their analysis is slightly different: in particular the Tornado codes analysis can only be applied to graphs with constant maximum degree, while LT codes graphs are of logarithmic density. Moreover, while for the Tornado codes the reception overhead is at least a constant fraction of the data length, for the LT codes it is an asymptotically fading fraction of the data length, even though with the tradeoff of lightly higher asymptotic encoding and decoding times. Another considerable advantage of these codes is that, once k and n have been chosen, the Tornado encoder generates n symbols and cannot produce others on the fly like LT codes do. Finally, there is a notable difference in the degree distribution of input and encoding symbols: for Tornado codes the degree distribution in input symbols is similar to the so-called *Soliton distribution*, while the degree distribution of encoding symbols can be approxhimated by the Poisson distribution. This thing, though, cannot be applied to the LT codes, since the Soliton distribution cannot be the resulting degree distribution on input symbols when encoding symbols are generated independently, regardless of the distribution of neighbors of encoding symbols. [3]

3.1 The encoding process

In [3] Luby gives a complete explanation of the encoding and the decoding process, and the former can be here briefly described by the steps:

- From a degree distribution, randomly choose the degree d of the encoding symbol;
- Choose uniformly at random d distinct input symbols which can be neighbors of the encoding symbol;
- Perform the exclusive-or of the d neighbors to obtain the value of the encoding symbol;

There are several degree distributions that can be used for our encoding symbols and the choice of each one comes with different performance in terms of coding times, efficiency and BER, which we will see further on in this work. In general the most important ones are:

- The *All-At-Ones distribution*: $\rho(d) = 1$;
- The *Ideal Soliton Distribution*: $\rho(1) = 1/k$ and $\forall i = 2, \ldots, k$ then $\rho(i) = 1/i(i-1)$;
- The *Robust Soliton Distribution*:
  - $\beta = sum_{i=1}^{k} \rho(i) + \tau(i)$
  - $\forall i = 1, \ldots, k$ then $\mu(i) = (\rho(i) + \tau(i))/\beta$

  with, for an $R = c \cdot ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$, the quantity $\tau(i)$ being:

$$\tau(i) = \begin{cases} R/ik & \text{for } i = 1, \dots, k/R - 1 \\ Rln(R/\delta)/k & \text{for } i = k/R \\ 0 & \text{for } i = k/R + 1, \dots, k \end{cases}$$

3.2 The decoding process

We can look at the quality of the coding scheme by analyzing the performance of the decoding process under different points of view, from the finite lenght analysis of the error probability [4], to the analysis of the success probability and the overhead dimension for a small number of packets N [5], to the opposite analysis with a high N and another decoding technique [2].

3.2.1 Error probability for the Belief Propagation (BP) decoder

In [4], the authors outline a method for calculating the success probabilty in the case of the BP decoder, which provides for them a much more efficient decoder than the classical *Gaussian elimination* [6]. The BP decoder performs the following steps until either all the input symbols have been recovered or until no output symbols of degree one are present in the graph. At each step of the algorithm, the decoder identifies a degree-one output symbol: if not all the input symbols have been recovered yet but none exists,a decoding failure is reported; otherwise, the value of the output symbol of degree one recovers the value of its unique neighbor among the input symbols. Once this input symbol value is recovered, its value is added to the values of all the neighboring output symbols, and the input symbols and all edges emanating from it are removed from the graph. So, as the BP decoder recovers one input symbol at each step, following Luby's notation, we call the set of output symbols of reduced degree one the *output ripple* at step *i* of the algorithm. [6]

The start in [4] is from defining the shape of the distribution $\Omega = (\Omega_1, \dots, \Omega_k)$, chosen on the set $\{1, \dots, k\}$, and which induces a distribution $\mathcal{D}$ on $\mathbb{F}_2^k$ we can choose our output symbols from. An LT code so described has parameters $(k, \Omega(x))$, with x being the input symbol considered and $\Omega(x) = \Omega_1 x + \Omega_2 x^2 + \dots + \Omega_k x^k$ is the generating function of the distribution. The importance of the success probability of the decoder comes from the fact that the number n of encoding symbols to which we apply the BP strictly depends on it and the output degree distribution.

Later on in the work, the authors present a recursive approach centered on the probability distribution generating function $P_u(x, y)$, whose bit-complexity is shown to be proportional to $O(n^3 \log^2(n) \log\log(n))$ (n is the number of collected out symbols of the LT-code). The computation is therefore efficient and reveals to be useful also to derive recursions for the expected number of output symbols of degree 1 at each stage u as:

$$R(x) = x\Omega'(1-x) + x\ln(x) \qquad \text{for } x = \frac{u}{k} \tag{31}$$

3.2.2 Optimal degree distribution for small message length codes

Another point of view is given in [5] where, instead of focusing on the decoder, the authors decided to approach the problem from the perspective of finding the optimal degree distribution defining the number of blocks in each packet. They do so looking at the problem in two different ways: a Markov chain approach and a combinatorial one.

In the case of the Markov chain approach, from the receiver's point of view, the set of received and partially/fully decoded packets represents a state. At the same time, state transition probabilities depend on specific packets arrival probabilities, which in turn depend on the degree distribution used in the encoding process. From this it follows that the absorbing state is the one formed by the original blocks. It is worth notice that the state space needs only include the states that can not be reduced by the decoder (*irreducible*), plus further reductions can be performed playing on isomorphies on block paths as well as other tricks. However, the state space implosion is unavoidable and occurs already at the lowest values of number of blocks in the message N [5]. After these steps the transition probability matrix can be constructed like:

$$\mathbf{P} = \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{R} \\ \mathbf{0} & \boldsymbol{I} \end{pmatrix} \tag{32}$$

and we have $\boldsymbol{\pi_0} = (1, \dots, 0)$ for the initial distribution vector and $\boldsymbol{\pi_{abs}} = (0, \dots, 1)$ for the absorbing state. With the system set in this way, we can find the best degree distribution through two different criteria: either *minimizing the avg number of steps* needed for decoding, $E[T] = \boldsymbol{\pi_0}(\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{e}^T$, or *maximizing the success probability*, $\mathcal{P}_N = \boldsymbol{\pi_0}\boldsymbol{P}^N\boldsymbol{\pi}_{abs}$.

In the case of the combinatorial approach, the central observation is that in such case the orther in which the packets are received is irrelevant. The recursive algorithm that derives from it aims at calculating the probability $\mathcal{P}_N$ that a message consisting of N blocks is succesfully decoded with exactly N received packets. So in general we take this probability $\mathcal{P}_n = P_n(p_1, \ldots, p_n)$, with the trivial cases $P_0 = 1$ and $P_1(p_1) = p_1$ being the seeds for recursion, and condition it on $n - m$ of the n received packets having degree 1. Moreover we need to specify that having no duplicates and all the $n - m$ degree 1 packets distinct happens with probability $(n-1)!/m!n^{n-m-1}$ and this is needed to weight the remaining decoding problem, consituting the recursion, for the m other packets whose degree can be lowered from being ≥ 2 after removing the $n - m$ degree-1 packets. With these specifications, the recursive approach can be expressed as:

$$P_n(p_1, \ldots, p_n) = \sum_{m=0}^{n-1} \binom{n}{m} p_1^{n-m}(1-p_1)^m \tag{33}$$

$$\times \frac{(n-1)!}{m!n^{n-m-1}} P_m(p_1^{(n,m)}, \ldots, p_m^{(n,m)}), \tag{34}$$

where

$$p_j^{(n,m)} = \sum_{i=2}^{n} \frac{p_i}{1-p_1} \frac{\binom{m}{j}\binom{n-m}{i-j}}{\binom{n}{i}}, \qquad j = 1, \ldots, m \tag{35}$$

with the first fraction in 35 calculates the probability that a packet has degree i conditioned on that it is not of degree 1, while the second fraction is the probability that when "from an urn" containing n blocks, m of which are white, a degree-i packet is constructed, then the subsequent reduced degree is exactly j[5].

The results The performance analysis have been done comparing the obtained optimal results to four main distributions: the uniform distribution, the degree-1 distribution ($p_i = 1 \quad (i = 1)$), the binomial distribution and the Soliton distribution. In the case of the Markov approach the authors show how the *MinAvg* and the *MaxPr* lead to similar results, which in general are better than the results from all the other distributions. In particular, Table 2 shows how Binomial and Soliton distribution work still reasonably well, but not the same can be said regarding the other two distributions. In the parallel case of the combinatorial approach, two main aspects are highlighted: first, with the state

	MinAvg	MAxPr	Binomial	Soliton	Uniform	Degree-1
p_1	0.524	0.517	3/7	1/3	1/3	1
p_2	0.366	0.397	3/7	1/2	1/3	0
p_3	0.109	0.086	1/7	1/6	1/3	0
$E[T]$	4.046	4.049	4.133	4.459	4.725	5.5
$\mathcal{P}_3$	0.451	0.452	0.437	0.397	0.354	0.222

Table 2: Optimal performance and weights in the case N = 3

explosion not being such a problem now, the fact that we can analyze cases for values of N up to 30; second, being the eigenvalues of A (The second derivative matrix of $\mathcal{P}_N$) a rapidly decreasing sequence, the function is extremely insensitive to changes along the last eigenvectors. This bringing to the result that typically three conditions are enough to define a close-to-the-optimum distribution [5].

(Cri says: sistema la nomenclatura! Nel paragrafo di prima il nostro N era il k degli altri paragrafi, forse è meglio se ti attieni al k...)

3.2.3 Optimal decoding algorithm for high values of k

The approach explained in [2] is slightly different and is based also on the assumption, showed in [6], that LT codes perform very well for big values of k. In this way the authors underline the difference of their approach form the work of Hyytä *et al.* presented in the previous paragraph, since those methods already appeared to be not scalable and being able only to handle symbols $k \leq 10$.

In the LT decoding process, at each round a symbol is removed from the ripple and the related packets are updated accordingly. The process terminates succesfully when there is no more symbol left in the ripple and all symbols are recovered. Hence it is important not to let ripple size become 0 before all symbols are recovered. In this process the degree distribution plays a big role, since the arrival of new symbols in the ripple also depends in it.The algorithm proposed by Lu *et al.* in [2], called *full rank decoding*, is intended to extend de decodability of LT codes, preventing them from terminating prematurely. To accomplish that, whenever the ripple is empty and would cause an early termination, the method *borrows* a particular symbol, decodes it through some other method (in particular $Wiedeman$ algorithm) and places it in the ripple for the process to continue.

The criteria the authors use to choose the right symbol to $borrow$ is a maximization problem described by:

$$b_j = argmax_j(V(j)|\mathbf{V} := \sum_{i=1}^{n} \mathbf{v_i}) \tag{36}$$

with the used quantities representing:

- j the index of the borrowed symbol b_j;
- $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ the coefficients vectors in the linear equations representing the encoding symbols;
- $V(j)$ the value at index j of the vector $\mathbf{V}$;

The results obtained in [2] show improved performance under the analysis of the expected number of packets needed for decoding, as well as the fact that the performance of the method is close to the ideal one. All of this resulting in in an overall lower number of packets needed to recover all the symbols and a lower runtime.

3.2.4 BER bounds on UEP fountain codes

Another interesting point of view is the one given in [7], where the main aim is not optimizing the general success probability, but instead defining an upper and lower BER bound in the case of Unequal Error Protection (UEP) for Distributed Network Coding (DNC) schemes over Rayleigh fading channels.

Distributed coding is an efficient channel coding strategy that improves the transmission reliability of cooperative communication networks. In many real-life applications, however, source nodes require a form of unequal protection from errors, instead of the Equal counterpart extensively addressed in [8], and this is at the origin of the design of a form of fountain codes called UEP fountain codes. In [7], the authors adopt the wieghted method to realize such codes, which consists in dividing the transmission data into several protection groups according to their protection requirements and assigning corresponding protection weights.

The system model The system model in the paper is the following: consider a Wireless Sensor Network (WRN) where a single destination node (the sink) communicates with K source nodes through a common relay network with N relay nodes. The authors also add the assumption that the K source nodes belongs to T source groups with different error protection requirements. The data delivery from the source node to the sink is then carried out in three phases:

- *The broadcast phase* - In which the information sequence of each source node is appended with a header, containing the protection weight ω_τ, and a Cyclyc Redundancy Check (CRC) code, to form a data packet;
- *The Relay phase* - in which each relay node decodes the received data packets and selects uniformly at random a fixed number d of them. Among the selected data, d^τ of them are from the source node in the protection group τ and the degree distribution $\mu^\tau(x)$ is generated by applying the UEP DNC guidelines to the degree distribution $\mu(x)$;
- *The data recovery phase*

Moreover, the benefits of a relay-type network structure for a WRN is largely investigated in the work of Comaniciu *et al.*, where several relay solutions are analised and compared in the specific combination of LT codes instead of Random Linear (RL) ones [9].
Finally, in the model there hase been defined two main protection level groups (for sake of simplicity), a High Error Protection (HEP) and a Low Error Protection (LEP) group.

The BER bounds With these premises we can present here the BER bounds on the bit error probability for the HEP group, while the bounds for the LEP group can be easily obtained by simple switch of the terms K_H and K_L and substitution of the other parameters of the protection group. Considering a UEP DNC based on fountain codes with degree distribution on HEP $\mu_{w(ch)^H}$, K_H, K_L, number of data packets collected Q, the error sequence $\mathbf{e}$, the $K \times Q$ matrix $\mathbf{G}$ and Hamming weight $w(\cdot)$, then the BER performance for the protection group HEP with Maximum Likelyhood (ML) decoding over Rayleigh fading channels can be upper bounded by $\xi_U^F < min\{1, \xi_U\}$, where ξ_U is given by:

$$\xi_U = \sum_{k=1}^{K} \sum_{k_h=max\{0,k-K_L\}}^{min\{k,K_H\}} \frac{k_H}{K_H} \binom{K_H}{k_H} \binom{K_L}{k-k_H} \tag{37}$$

$$\cdot \left\{ \left[\sum_{w(\mathbf{c}_H)} \mu_{w(\mathbf{c}_H)}^H \zeta(w(\mathbf{c}_H), w(\mathbf{e})) = k, w(\mathbf{e}_H) = k_H) \right]^Q + P_r(\mathbf{eG} \neq \mathbf{0}, w(\mathbf{e} = k, w(e_H)) = k_H) \right\} \tag{38}$$

At the same time, the same BER performance can be lowerbounded by $\xi_L^F > max\{0, \xi_L\}$, where ξ_L is given by:

$$\xi_L = \sum_{k_h=max\{0,1-K_L\}}^{min\{1,K_H\}} \frac{k_H}{K_H} \binom{K_H}{k_H} \binom{K_L}{1-k_H} \tag{39}$$

$$\cdot \left\{ \left[\sum_{w(\mathbf{c}_H)} \mu_{w(\mathbf{c}_H)}^H \zeta(w(\mathbf{c}_H), w(\mathbf{e})) = 1, w(\mathbf{e}_H) = k_H) \right]^Q + P_r(\mathbf{eG} \neq \mathbf{0}, w(\mathbf{e} = 1, w(e_H)) = k_H) \right\} \tag{310}$$

The proof of the previous results can be found in [7].

Results Figure 1 (a) and (b) show the upper and lower BER bounds at an average $E_s/N_0 = 0.9dB$: as it is evident, the upper and lower bounds become asymptotically tight as the expanding coefficient η grows. Moreover, the BER performance of the HEP group is better, compared to the Equal protection design, while the LEP group is only slightly worse.[7]

4 An evolution of LT fountain codes: Raptor codes

For many applications, there is then the necessity to construct universal fountain codes which have a fast encoding algorithm and for which the average weight of an output symbol is a constant. Such a class of fountain codes is well seen and described in [6] and has the name of *Raptor codes*. Previous results show how LT-codes cannot be encoded with constant cost if the number of collected output symbols is close to the number of input symbols. As a matter of fact, the decoding graph for LT-codes needs to have a number of edges in the order of $klog(k)$ ($k =$ the number of input symbols) so as to make sure that all the input nodes are recovered with high probability. Here Raptor codes come into play

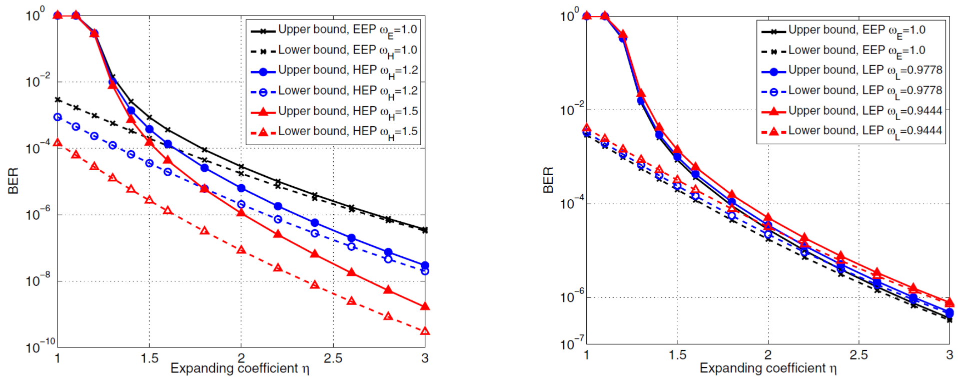

Figure 1: BER performance bounds comparison of the UEP DNC scheme based on fountain codes with ML decoding over Rayleigh fading channels with $T = 2$ and $K = 100$. The left figure is for HEP group ($\alpha_H = 0.1$), the right one is for LEP group ($\alpha_L = 0.9$) Source [7]

relaxing this condition and requiring that only a *constant fraction* of the input symbols to be recoverable. There is a way around the potential problems emerging from this, in particular to the one stating that it is not enough we recover only a constant fraction of the input symbols, all of them must be recovered.

The author addresses the issue encoding the input symbols with the use of a traditional erasure correcting code and later with the application of an appropriate LT-code to the new set of symbols in such a way that the traditional code is capable of recovering all the input symbols even in the case of a fixed fraction of erasures. This is why often Raptor codes are briefly identified as the combination of an LT-fountain code with an appropriate precode.

This class of codes can thus be formally defined in this way: let $\mathcal{C}$ be a linear code of block length n and dimension k, and let $\Omega(x)$ be a degree distribution; a *Raptor code* with parameters $(k, \mathcal{C}, \Omega(x))$ is an LT-code with distribution $\Omega(x)$ on n symbols, representing the coordinates of codewords in $\mathcal{C}$. The code $\mathcal{C}$ is called the *pre-code* of the Raptor Code, while the input symbols of a Raptor code are the k symbols used to construct the codeword in $\mathcal{C}$ consisting of n *intermediate symbols*.

4.1 Performance analysis of BER in a BIMSC

(Cri says: channel type spec) This class of codes has been widely used in several scenarios since it was first designed, with the convenient approach of looking at the boundary conditions for the fountain codes and expanding them to the case of Raptor codes. In particular, in [10] Etesami *et al.* investigate the supposition that, since the success of Fountain codes for erasure channels, a similar result can be expected for other binary symmetric channels and so also for Raptor codes. In the work, the authors show how some of the properties of LT- and Raptor codes over the erasure channel can be transferred onto any Binary Memoryless Symmetric Channel (BIMSC). In some details, they consider the degree distribution optimized for the erasure channel and study the residual BER of the decoder in the case of BP algorithm being applied and this as a function of the overhead chosen. In the paper there are some major emphases on the structure of the degree distribution, which now needs to be used from the perspective of edges, the channel structure, considering binary antipodal signaling and the feasibility of the classical Gaussian approximation technique for simplifying the density of the messages passed at each round to the BP algorithm. Also the important concept of *Reception overhead* of an algorithm is described: having $p_i =$ the probability that a received bit was 0 before transmission, $m =$ the number of collected output bits and $E = \sum_{i=1}^{m}(1 - h(p_i))$, then the decoding algorithm has a reception overhead ϵ if $E = k(1 + \epsilon)$, which is, the algorithm needs a number of output bits which is only ϵ away from the optimal number.

Concerning the simulations, these were performed for optimized degree distributions with respect to the Binary Erasure Channel (BEC), as already described in [6]. The experiments in [10] use the output distribution

$$\Omega(x) = 0.008x + 0.494x^2 + 0.166x^3 + 0.073x^4 + 0.083x^5 + 0.056x^8 \quad (411)$$

$$+ 0.037x^9 + 0.056x^{19} + 0.025x^{65} + 0.003x^{66} \quad (412)$$

In the performed experiments, the focus was on the BER at the end of the decoding process. The graphs in Fig2 in [10] show the advantage of Raptor Codes over LT-codes clearly, and it is observed that for a small average degree the LT-codes exhibit a bad error floor behavior. Nevertheless there is still room for improvement through the addition of a technique called *Gaussian approxhimation*: the method approxhimates message densities as a Gaussian, or a mixture of them, in such a way that it is possible to collapse the density of the messages passed to the BP algorithm to a single-variable recursion for the mean of a Gaussian.

4.2 BER bounds for Raptor codes in Rayleigh fading channels

Finally, another notable result is depicted in [11] where Yue *et al.* derive a lower bound for the LT-based DNC scheme over Rayleigh fading channels and upper/lower bounds for the BER in the Raptor based version, in which an inner LT code concatenates a conventional outer code to reduce the error floor.

The scenario of the paper provides a WRN with multiple source and relay nodes and a single sink. The main difference with the scenario depicted in section 3.2.4 is that here the relay nodes must be of two types: a precoding relay group and an LT-coding relay group. Finally, in this case the decoding is performed throught the ML criterium.

4.2.1 Lower BER bound for LT scheme

In [11] the authors first find a lower bound for the LT-based DNC scheme with ML decoding over Rayleigh fading channel, starting from the information sequence error probability

$$P_e = P_r\left(\bigcup_{\mathbf{e}:\omega(\mathbf{e})\neq 0} \hat{\mathbf{m}} = \mathbf{e}\right) \tag{413}$$

In this case, $\hat{\mathbf{m}}$ is the estimated information sequence and $\omega(\cdot)$ the Hamming weight as usual. This probability is given by the probability that only one symbol in the decoded sequence is recovered uncorrectly. Inside this value, the part representing the Rayleigh fading channel is given by $P_r(\mathbf{v} \to \mathbf{v}'|\hbar)$, which is the conditional probability that the decoded codeword is equal to $\mathbf{v}'$ give the channel fading coefficient $\hbar$.

For what concerns the bounds on BER for Raptor codes we can see that:

- The probability of decoding errors for a Raptor code of parameters $\Omega(x)$, $\mu(x)$, K_L, K, Q and expanding coefficient η is upperbounded by $P_b < min\{1, \xi_U^{Raptor}\}$, with ξ_U^{Raptor} given by:

  $$\xi_U^{Raptor} = \sum_{t=1}^{K} \frac{1}{K}\binom{K}{t}\xi_U^t(1-\xi_U)^{K-t}\cdot\left(\sum_{\omega(\mathbf{h})}\Omega_{\omega(\mathbf{h})}\zeta_t^{R,\omega(\mathbf{h})}\right)^{K-K_L} \tag{414}$$

  where

  $$\zeta_t^{R,\omega(\mathbf{h})} = \frac{\sum_{\alpha=even,0\leq\alpha\leq t}\binom{t}{\alpha}\binom{K-t}{\omega(\mathbf{h})-\alpha}}{\binom{K}{\omega(\mathbf{h})}} \tag{415}$$

- The probabilty of the aforementioned Raptor code is lower bounded by $P_b > max\{0, \xi_L^{Raptor}\}$, where ξ_L^{Raptor} is given by:

$$\xi_L^{Raptor} = \sum_{t=1}^{K} \frac{1}{K} \binom{K}{t} \xi_L^t (1-\xi_L)^{K-t} \Big(\sum_{\omega(\mathbf{h})} \Omega_{\omega(\mathbf{h})} \zeta_t^{R,\omega(\mathbf{h})} \Big)^{K-K_L} - \frac{1}{2} \sum_{t=1}^{K} \binom{K}{t} \times \tag{416}$$

$$\xi_U^t (1-\xi_U)^{K-t} \sum_{\tau_0}^{t} \sum_{\tau_1 = t-\tau_0} \sum_{\tau_2=0}^{K-\tau_0} \frac{\binom{t}{\tau_0}}{2^t} \frac{\binom{K-\tau_0}{\tau_2}}{2^{K-\tau_0}} \frac{\binom{K-\tau_0}{\tau_2}}{2^{K-\tau_0}} \Big(\sum_{\omega(\mathbf{h})} \Omega_{\omega(\mathbf{h})} \zeta(\tau_0, \tau_1, \tau_2, t) \Big)^{K-K_L} \tag{417}$$

where

$$\zeta_t^{R,\omega(\mathbf{h})} = \frac{\sum_{\alpha=even, 0 \le \alpha \le t} \binom{t}{\alpha} \binom{K-t}{\omega(\mathbf{h})-\alpha}}{\binom{K}{\omega(\mathbf{h})}} = \frac{\sum_{\alpha=even, 0 \le \alpha \le min\{\tau_p, \omega(\mathbf{h}_{\tau_\mathbf{p}})\}} \binom{\tau_p}{\alpha} \binom{K-\tau_p}{\omega(\mathbf{h})-\alpha}}{\binom{K}{\omega(\mathbf{h})}} \tag{418}$$

Moreover, we specify that ξ_U is the upper BER bound of the LT codes, while $\mathbf{h}$ is a column vector of the parity check matrix $\mathbf{H}$ and that τ_i is the length of the i^{th} of its columns.

4.2.2 Results

The results in [11] show several important points: first of all, that both the LT codes and the Raptor codes BER bounds become asymptotically tight to one another as the expanding coefficient η grows. Furthermore, as shown in figure Figure 2, the performance of Raptor codes are much better than LT-codes ones, regardless of the expanding coefficient. This behavior is coherent with the results that Yue *et al.* also find in [7].

It is also shown that there is an enhancement also with respect to the valure of E_s/N_0 since, as it increases, the difference $\xi_U^{Raptor} - \xi_L^{Raptor}$ becomes smaller; and finally, also that as the code rate of the codes used in the construction of the Raptor codes decreases, the BER bounds get better.

5 Conclusions

The brief state of the art presented with this work highlights the main achievements accomplished so far in the field of digital fountain codes, and in particular in the cases of

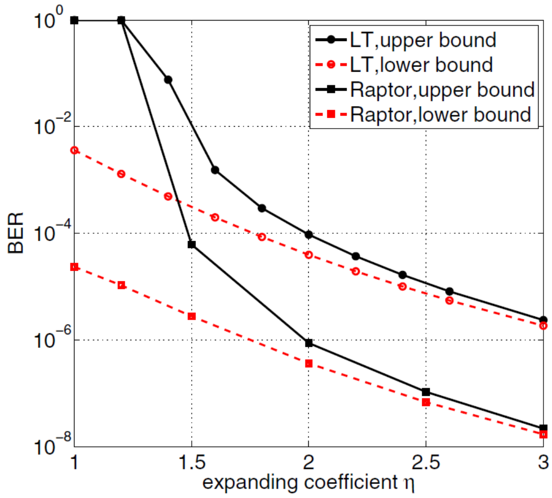

Figure 2: BER bounds comparison for LT-based and Raptor-based DNC schemes over Rayleigh fading channels with $K_L = 98$, $K = 100$ and $E_s/N_0 = 7dB$. Source [11]

LT- and Raptor codes. The results that emerge from the works analyzed clearly emphasize the improvements that these approaches can bring to a vast set of problems, from DNC to WRN in terms of decoding success probability, BER, overhead shortening and so decoding time and efficiency, while at the same time there is still much space for further insights and improvements.

References

[1] J. W. Byers, M. Luby, and M. Mitzenmacher, "A Digital Fountain Approach to Reliable Distribution of Bulk Data,"

[2] F. Lu, "LT Codes Decoding : Design and Analysis," pp. 1–5.

[3] M. Luby, "LT Codes,"

[4] R. Karp and M. Luby, "Finite Length Analysis of LT Codes," p. 2004, 2004.

[5] E. Hyytiä, "Optimal Degree Distribution for LT Codes with Small Message Length," pp. 2576–2580, 2007.

[6] A. Shokrollahi and S. Member, "Raptor Codes," vol. 52, no. 6, pp. 2551–2567, 2006.

[7] J. Yue, Z. Lin, B. Bai, and B. Vucetic, "Performance Analysis of Unequal Error Protection Distributed Network Coding Based on Fountain Codes," vol. 3, no. 3, pp. 285–288, 2014.

[8] K. Pang, Z. Lin, S. Member, and B. F. Uch^, "Distributed Network Coding for Wireless Sensor Networks Based on Rateless LT Codes," vol. 1, no. 6, pp. 561–564, 2012.

[9] C. Comaniciu and J.-m. Gorce, "Toward Increasing Packet Diversity for Relaying LT Fountain Codes in Wireless Sensor Networks," vol. 15, no. 1, pp. 52–54, 2011.

[10] O. Etesami and A. Shokrollahi, "Raptor Codes on Binary Memoryless Symmetric Channels," vol. 52, no. 5, pp. 2033–2051, 2006.

[11] J. Yue, S. Member, Z. Lin, S. Member, and B. Vucetic, "Performance Analysis of Distributed Raptor Codes in Wireless Sensor Networks," vol. 61, no. 10, pp. 4357–4368, 2013.