**University of Alberta**

# Customized Raptor Code Designs for Finite Lengths and Practical Settings

by

**Kaveh Mahdaviani**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

**Master of Science**

in

Communications

Department of Electrical and Computer Engineering


Fall 2012
Edmonton, Alberta

*To my family.*

# Abstract

In this dissertation we present new methods for designing efficient Raptor codes in finite and practical block lengths. First we propose an extension of Raptor codes which keeps all the desirable properties, including the linear complexity of encoding and decoding per information bit, and improves the performance in terms of the reception rate. Our simulations show a $10\%$ reduction in the required overhead at the benchmark block length of 64,520 bits, and with the same complexity per information bit. Second, we consider the practical setting with short block lengths of $10^3 < k < 10^4$. Based on a new vision of the inactivation decoding process, we set a new degree distribution design criterion for the Luby transform (LT) part of Raptor codes. A family of degree distributions that satisfy the new design criterion is analytically derived. The finite length performance of this family is investigated by using computer simulations and is shown to outperform the conventional design.

# Acknowledgements

I would like to express my sincere and deep grateful thanks to my supervisors Dr. Masoud Ardakani, and Dr. Chintha Tellambura. I am certain that I would not be able to achieve any success during my M.Sc. program without the never-ending support, kindness, and encouragement of Dr. Ardakani and his valuable advises at the critical moments. He spent limitless time to help me with his lightening advises through every single step I have taken in this program. His caring and supportive attitude is iconic and I feel very lucky to have the chance to work under his supervision. I have learned a lot from him regarding finding a good problem, to see the value in the ideas, technical writing, presentation skills, etc, and yet I believe I have learned even more from his extraordinary great personality.

I would also like to mention how I deeply respect the patient and support from my other supervisor, Dr. Tellambura. His support left no room for discouragement and no reason for doubt. His continuous advice, guidance and encouragement have been instrumental in making this work a success. It has been a great privilege to be a part of his research team.

I also wish to thank the members of my thesis committee, Dr. Hai Jiang, and Dr. Mohammad R. Salavatipour, for their valuable comments and advice in improving this thesis.

Many thanks to my colleagues and friends at the 5th floor of ECERF building for their help and support in various aspects and specially to my friends Dr. Raman Yazdani, Mr. Mahdi Ramezani, Mr. Moslem Noori, Dr. Hossein Bagheri, and Dr. Payam Dehghani with whom I have had the chance to go through many insightful discussions. I would like to extend my special thanks to all my lab-mates

at the iCORE Wireless Communications Laboratories and my friends in the Computing Science Department of the University of Alberta, and specially Mrs. Parisa Mazrooei. It has been a great experience for me to live and work in such a friendly atmosphere with them.

Finally, I would like to express my eternal gratefulness to my family for their everlasting support in all aspects of my life even though I left them to study thousands of miles away. I can not imagine any motivation in my life without their limitless support. This work along with every other step I take in my life is dedicated to them.

# Table of Contents


**1 Introduction** **1**

1.1 Motivation . . . . . 1

1.2 Contribution and Outline of Thesis . . . . . 3

**2 Background** **6**

2.1 Erasure Channels . . . . . 6

2.2 Retransmission Scheme and Universality . . . . . 8

2.3 Conventional Forward Erasure Correction Scheme . . . . . 9

2.4 Fountain Codes and Rateless Property . . . . . 11

2.5 LT Codes . . . . . 14

2.5.1 Encoding . . . . . 15

2.5.2 Decoding . . . . . 16

2.5.3 Degree Distribution Design . . . . . 20

2.6 Raptor Codes . . . . . 24

**3 Annotated Raptor Codes** **29**

3.1 Introduction . . . . . 29

3.2 Background and Notations . . . . . 30

3.2.1 Encoding . . . . . 30

3.2.2 Edge Deletion Decoding . . . . . 31

3.3 Main Idea . . . . . 32

3.4 Annotated Raptor Codes . . . . . 35

3.4.1 Encoding . . . . . 36

3.4.2 Decoding . . . . . 37

3.5 Some Comments on Design . . . . . 37
3.6 Example Code and Numerical Results . . . . . 39
3.7 Summery . . . . . 42

**4 On Raptor Code Design for Inactivation Decoding 43**
4.1 Introduction . . . . . 43
4.2 Encoding and Decoding of Raptor Codes . . . . . 44
4.3 Degree Distribution Design . . . . . 48
4.3.1 Evolution of $\Omega(x)$ During ID . . . . . 48
4.3.2 Previous Results . . . . . 49
4.3.3 A New Design Criterion . . . . . 50
4.3.4 The Proposed Code Design . . . . . 52
4.3.5 Finite Maximum Degree Design . . . . . 53
4.4 Numerical Results . . . . . 55
4.5 Summary . . . . . 58

**5 Conclusions 60**
5.1 Future Research Directions . . . . . 61

**Bibliography 63**

# List of Tables

3.1 Example Code with $k = 64520$ and $k_a = 800$. . . . . 41

4.1 Average performance measures for the 3GPP code and the proposed design with $m = m^*$. . . . . 59

# List of Figures

2.1 A $q$-ary erasure channel with an input alphabet of size $q$, and an output alphabet of size $q+1$. . . . . 7

2.2 Samples of decoding progress in terms of the number of decoded symbols with respect to the number of received symbols for LT codes for $k = 65536$ . . . . . 25

2.3 Average decoding progress in terms of the number of decoded symbols with respect to the number of received symbols for LT codes for $k = 65536$. The dashed red line shows the total number of input symbols. . . . . 26

3.1 Complexity per information bit vs. rate for capacity approaching sequences of LDPC codes designed for the BEC. . . . . 40

4.1 (a) The decoding graph with output nodes $c_1$ to $c_7$ and intermediate bits $b_1$ to $b_6$. The ripple is initiated with $c_1$, which recovers $b_1$ and then becomes empty. (b) The reduced degree two induced subgraph $G$ based on the remaining effective decoding graph. In $G$, every reduced degree two output node will represent an edge. Here, $G$ contains two connected components. The maximum component contains four nodes. Max-Component ID may, for example, choose $b_4$ as a node in the maximum component of $G$ for inactivation. This choice will refill the ripple with $c_3, c_5$, which, in turn, recover $b_3$ and $b_5$ in terms of $b_4$. . . . . 47

4.2 The decoding graph at some intermediate step of decoding. The receiver started the decoding after receiving $k' = (1 + \varepsilon)k$ output symbols, where $\varepsilon$ is a very small positive number. Up to this point, a $\delta = \frac{\ell}{k}$ of the intermediate bits have been recovered. The left part shows the non-effective part of the decoding graph at this moment, which could not participate in the decoding of the remaining bits. The right part is still effective and contains $(1 - \delta)k$ intermediate bits and approximately $(1 - \delta)k'$ output nodes as well. . . . . 51

4.3 CDF of the normalized number of inactivations required for successful decoding, $\ell = 1024$. Simulations are performed based for $10^4$ iterations. In each iteration a complete block of information is transmitted and its overhead is calculated. . . . . 56

4.4 CDF of the normalized number of inactivations required for successful decoding, $\ell = 8192$. Simulations are performed based for $10^4$ iterations. In each iteration a complete block of information is transmitted and its overhead is calculated. . . . . 57

# List of Symbols

| | |
|---|---|
| $GF(q)$ | Galois field of order $q$. |
| $\mathcal{O}\left(f(x)\right)$ | The big-O order of a function $f(x)$. |
| $\mathbb{E}[X]$ | Statistical average of random variable $X$. |
| $\ln(x)$ | Natural Logarithm function of a positive real number $x$. |
| $f_X(x)$ | Probability density function of random variable $X$. |
| $\Pr$ | Probability |
| $x\,!$ | Factorial |
| $\binom{N}{k}$ | Combinations of $k$ elements from $N$ elements. |
| $\varepsilon$ | Reception overhead. |
| $\Omega(x)$ | Generating polynomial of the output nodes degree distribution in Raptor codes. |
| $\iota(x)$ | Generating polynomial of the intermediate nodes degree distribution in Raptor codes and the input nodes degree distribution in LT codes. |
| $\bar{d}$ | Average degree of output nodes in Raptor and LT codes. |
| $\sigma$ | Practical gap to capacity for LDPC outer-codes in Raptor codes. |
| $\mathbf{O}_{\ell_1 \times \ell_2}$ | $\ell_1 \times \ell_2$ all zero matrix. |
| $\mathbf{I}_\ell$ | $\ell \times \ell$ identity matrix. |
| $\oplus$ | XOR operation. |
| $\Phi_i$ | Probability of producing a non-annotated output symbol of degree $i$ in Annotated Raptor codes. |
| $\Psi_i$ | Probability of producing an annotated output symbol of degree $i$ in Annotated Raptor codes. |
| $\bar{I}_{Rand}$ | Average normalized number of inactivations using the Random ID selection strategy. |
| $\bar{I}_{Max-Comp}$ | Average normalized number of inactivations using the Max-Component ID selection strategy. |

# Acronyms

**LT code** Luby Transform code

**RS code** Reed-Solomon code

**MDS** Maximum Distance Separable

**DSL** Digital Subscriber Line

**WiMAX** Worldwide Interoperability for Microwave Access

**DVB** Digital Video Broadcasting

**ATSC** Advanced Television Systems Committee

**BEC** Binary Erasure Channel

**EDD** Edge Deletion Decoding

**ID** Inactivation Decoding

**LDPC code** Low Density Parity Check code

# Chapter 1

# Introduction

## 1.1 Motivation

Communication networks such as the Internet and cellular networks have tremendously improved every aspect of our lives. Our modern life benefits from this technology to the extent that it cannot function without communication networks any more. The demand for this technology is ever increasing. According to the Cisco, the projected Internet traffic is in a hockey stick-like upward curve to reach 50 billion connected devices by 2020. This projected growth demands increased speed, bandwidth, and throughput, which have pressed network designers into a constraint-bound corner. Cisco Visual Networking Index (VNI) anticipates the annual global IP traffic will reach two-thirds of a Zettabyte (Trillion Gigabytes) by 2013 [1]. This number represents more than a fivefold increase over today's IP traffic.

Failure to fulfil this demand affects the development and expansion of on-line services where, lowering the costs and saving time and energy is crucial for the stability and improvement of economy, especially in developed counties such as Canada. Moreover, such a failure will harm all related industries such as communication device manufacturing and all emerging on-line services such as Internet TV, on-line education, sensor networks, and the revolutionary cloud technology just to name a few.

Satisfying this demand based on the current methods of data transmission requires significant increase in the infrastructure at a tremendous cost. However,

thanks to the novel ideas in the coding theory, such as rateless coding, data transmission at much higher rates on the available hardware settings can be made possible. This is an accepted fact by the community of experts and new versions of international standards for communication over networks have already been revised to include the possibility of using rateless coding.

The main problem in communication networks is to achieve high throughput while facing noise, packet loss, interference, and fading all of which vary with time, sometimes even at time-scales shorter than a single packet transmission time. In such situations, achieving high data transmission rates with conventional transmission schemes requires perfect tracking of channel parameters and well adaptation to the instantaneous conditions. An ideal solution is a rateless coding scheme, in which the scheme of encoding in the transmitter does not need any explicit estimation or adaptation of the channel quality. However, the transmission rate will implicitly adapt to the channel's level of quality.

As an example, a packet broadcasting communication system is one which handles the communication of data from one source to several receivers in packets of usually very large number of bits. An important instance of such a system is the Internet. In particular, for multimedia files, a server broadcasts the same information packets to multiple clients. Communication in such systems, however, faces challenges and necessitates new specialized solutions to leave behind current limitations. One development that has recently attracted a great deal of interest is the idea of Fountain codes which was first mentioned without an explicit construction in [2, 3]. In particular, a family of Fountain codes, called Raptor codes [4, 5] has already been incorporated in several 3rd generation wireless (3GPP) and digital video broadcasting (DVB) standards, including but not limited to, 3GPP MBMS, DVB IPTV AL-FEC, and DVB RSC [6]. Thus, Fountain codes are currently the subject of extensive research both in academia and industry (e.g., Digital Fountain Inc.) for practical technologies.

Before the invention of Fountain codes, when broadcasting to receivers with different channel quality, the data rate had to be chosen according to the worst

receiver. That is, a high-speed receiver had to suffer because of the presence of a low-speed receiver. Fountain codes resolved this problem by transmitting different mixtures (i.e., different XOR combinations) of original data packets. Receivers use the received packets to decode the original data by solving a set of liner (XOR) equations. Theoretically, broadcasting continues until all the receivers are able to decode the data successfully. In practice, however, this process terminates earlier because of various limitations. The high-speed receiver is able to decode the data as soon as sufficient number of packets (enough linear equations) is received in a relatively short period of time. The low-speed receiver can also decode all the data, but compared to the high-speed receiver, it takes a longer time to receive enough packets, i.e., it has to listen longer to the channel. Hence, interestingly, each receiver pays for channel access according to its own channel quality.

In this dissertation we will address some of the imperfections rising in the application of Raptor codes in practical settings. The two main categories of these imperfections include the effect of finite length and the computational complexity. Regarding the first one, it worth to mention that all the analysis and design of Raptor codes are conventionally based on the assumption of an infinite information block length. However, in practice we always deal with finite block lengths. This fact invalidates some of the assumptions in the conventional analysis and causes some gap between the maximum achievable rates and the channel capacity. There has been some attempts to consider these effects and some modifications has been made to reduce this gap specially in very short block lengths, but most of these attempts lead to computationally complex solutions. Here, we will reconsider the design of Raptor codes long with the effect of these imperfections and introduce some modifications in the design of these codes to keep the complexity affordable and reduce the gap between the achievable transmission rates and the channel capacity.

## 1.2 Contribution and Outline of Thesis

The next chapter of this thesis will briefly review the the overall system and channel models used throughout the thesis. The conventional scheme of data transmission

over such channel is discussed and the classical fixed rate coding solutions for the channel model are reviewed. Moreover the basic idea of Fountain coding is introduced and finally LT codes and Raptor codes are presented as the two practical instances of Fountain codes.

In Chapter 3, we propose some modification in the original design of Raptor codes based on a technique referred to as the "*Annotation*". Annotation provides the possibility of turning a portion of useless transmissions in the original design of Raptor codes into innovative transmissions providing new information to the receiver. The Annotation scheme is introduced in details and the performance of annotated Raptor codes are compared with the original design of Raptor codes, using computer simulation. As confirmed by the simulations the annotated scheme is capable of reducing the number of useless transmissions by $10\%$ in the benchmark example, while keeping the complexity unchanged. Yet the main advantage of the annotation technique is to provide a framework for having different levels of protection to the information symbols in the transmission. This property can be easily used in other directions. As an example, one can use annotation for unequal error protection rather than reducing the number of useless received symbols and increasing the reception rate. This work was presented in the IEEE Information Theory Workshop (ITW) conference in October 2011, in Paraty, Brazil [7].

Design of Raptor codes specified for small block lengths are discussed in Chapter 4. Current practical settings suggest using information blocks as short as a few thousands. In this case a computationally more expensive decoding algorithm, named "*inactivation decoding*," introduced in [8], is preferred due to its better performance in terms of overhead. The effect of using inactivation decoding is considered .Accordingly a new design criterion has is introduced. The new design based on this criterion is performed and the performance of new design is compared to that of conventional Raptor codes already adopted in the 3GPP by the means of computer simulations. The results show notable improvements in the computational costs of this practical method. The new design reduces the number of computationally expensive inactivation operations significantly. This work is accepted for

publication in the IEEE Transactions on Communications and is currently scheduled for the upcoming issue [9].

Finally, Chapter 5 presents our conclusions and suggests some potential future research initiatives resulting from this work.

# Chapter 2

# Background

The goal of this chapter is to briefly review some major concepts which have been used in this thesis. The following section introduces the erasure channel which is considered as the model of the channel under study in this thesis. Retransmission as a practical solution which has been widely used in data transmission over erasure networks is reviewed in section 2.2, and the *"universality property"* is introduced. Next, we review the properties of *"Reed-Solomon"* codes as the most successful fixed rate erasure correcting codes in section 2.3. The *"MDS property* is also discussed in the same section. Sections 2.4 to 2.6 review the idea of rateless coding and the two famous practical instances of rateless codes namely LT codes and Raptor codes. Their encoding and decoding schemes along with their complexity properties are explained.

## 2.1 Erasure Channels

An, *erasure channel* is a mathematical model for channels which introduce corruption to the transmitted data in an extremist manner. The output of such a channel is either the perfectly transmitted input, or a report of failure without any partial information about the transmitted symbol (i.e. *soft information*). Although this model was more of an extreme theoretical case when it was introduced by Elias [10] first in 1955, it is a perfect model in many existing and emerging communication systems nowadays. Probably the most famous example of a practical erasure channel is a link in a packet network such as the Internet. The packets transmitted on these

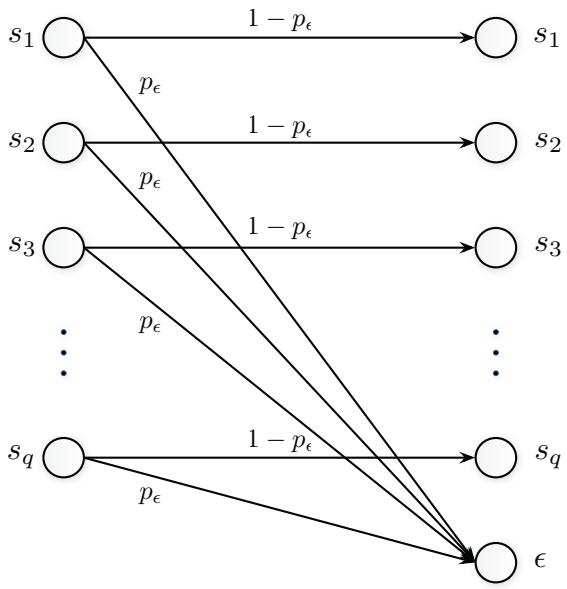



Figure 2.1: A $q$-ary erasure channel with an input alphabet of size $q$, and an output alphabet of size $q+1$.

links are either received without error or not received. In a more general view, any noisy channel protected with an error-correcting code behaves like an erasure channel since the received symbol is either decodable or completely useless. In the first case, the error correcting code recovers the transmitted symbol perfectly, and the transmission can be supposed to be performed noiseless. Otherwise, the received symbol is very poor and not decodable which then can be taken as an erasure.

In a more accurate way, the mathematical model of a $q$-ary erasure channel could be described as depicted in Fig. 2.1. The set of input alphabet for such channel is $\{s_1, s_2, \ldots, s_q\}$, while the output alphabet set consists of all the possible inputs, and an erasure report $\epsilon$. Each input symbol is then assumed to have a $1-p_\epsilon$ probability of successful transmission, and will be erased by the channel with probability $p_\epsilon$. The capacity of this channel can be easily calculated as [11]

$$C = (1-p_\epsilon)\log_2(q)$$

## 2.2 Retransmission Scheme and Universality

Conventionally, communication over an erasure channel has been performed by retransmission based on feedback. Under this approach, any receiver is supposed to inform the transmitter about the packets or segments of the data which are received by sending a small amount of information on the feedback channel. The transmitter will then be able to track the reception process, identifies the erased transmissions, and keeps transmitting them again until all the data is received successfully [12].

Besides simplicity, this scheme has a very interesting property which made it, conventionally, the widely accepted practical solution for transmission over an erasure channel. This important property is the independence of the transmission scheme from the channel state. In other words, no matter what the probability of erasure in the channel is, the transmission scheme always works the same way. As we will discuss later this transmission scheme is not even optimal, however, the simplification coming from its independence from the channel state, has been of such a great benefit in practice to compensate its lack of optimality. This universality property gives a single design of transmitter and receiver for any erasure channel. Hence, no channel parameter estimation is required and no changes will be required in the systems for the adaptation of transmission scheme to the channel state. However, yet this scheme provides a higher rate of data transmission in a channel with higher quality and vice versa. We will refer to this property as "*universality*".

In most of the real world applications there is no perfect feedback channel provided. In many applications, such as internet based communications, the feedback channel is exactly the same as the forward channel. Therefore, the feedback is also subject to erasure at the same rate. Moreover, in most cases feedback traffic also uses the same resources as of the data transmission, and hence reduces the portion of capacity achieved by data transmission rate.

The wastefulness of the feedback based retransmission scheme turns out to be even worse in a broadcast scenario to multiple receivers with different channel qualities. The common situation in an intermediate stage of this scenario is to have each receiver received a random fraction of the transmitted data which is independent of

the fraction received by other receivers. Therefore, transmitter has to keep broadcasting many packets while some of the receivers have already received them but others have not. Now assume there is a receiver with a very poor channel quality. This receiver will keep requesting many packets while most of the other receivers have already received them. However, the transmitter has to either drop the weakest receiver and do not answer its request any more, or to keep retransmitting the requested packets. This translates in either failure of weak receivers at any given time, or many redundant receptions for most of the receivers all over the transmission period. Adding the fact that for many applications such as wireless or internet based communications, the quality of channel changes over time, and thus many receivers will experience a period of time when their channel is in a very poor quality, shows that the strategy of giving up over weak receivers will fail many receivers. Hence, huge redundant reception looks inevitable under the feedback based retransmission for broadcast.

The important lesson learned from Shannon is that the capacity of the erasure channel does not depend on the existence of feedback channel [11]. This means that there should be an erasure correcting coding scheme which does not use feedback, or just use the feedback for a limited constant amount of data transmission, and yet is able to transmit data at rates arbitrarily close to the capacity.

## 2.3 Conventional Forward Erasure Correction Scheme

The study of forward erasure correcting codes has a long history. The most powerful code for erasure correction is widely accepted to be the Reed-Solomon (RS) code [13–16]. The power of these codes is in the sense that they are capable of correcting the highest number of erasures compared to all the other erasure correcting codes of the same block length. In other words, if we have a block of $k$ information symbols of a $q$-ary alphabet, (where $q = 2^I$ for some integer $I$), and encode them, using a RS code, to a block of $n$ encoded symbols of the same alphabet, for some $n$ such

that

$$k < n \leq 2^q - 1.$$

Then the decoder is capable of fixing up to $n - k$ erasures on any transmitted codeword, and obviously no other coding scheme will be able to do better than this. This property is referred to as the "*maximum distance separable*" (MDS), and any code having this optimal property is called an MDS code. As a result receiving any $k$ packet from an $n$-packet length encoded block suffices for the successful decoding.

Reed-Solomon codes have received much attention after their introduction in late 60's. They have been used in many application including the digital data storage devices such as Compact Discs, DVDs, Blue-ray Discs, as well as in digital communication applications such as digital subscriber line (DSL), worldwide interoperability for microwave access (WiMAX), and in broadcasting applications such as the digital video broadcasting (DVB) standard DVB-S, and advanced television systems committee (ATSC) standards. Albeit, they are now being replaced by modern alternatives such as low density parity check (LDPC) codes used in the new version of DVB-S, named DVB-S2.

Despite all the desirable properties of RS codes, like any other linear block code, they have a fixed predefined rate $R = k/n$. If the number of erasures occurred in the channel during the transmission of a $n$-packet long encoded block is less than or equal to $n - k$, then as described before the decoder is able to recover the whole $k$-packet long data. However, if the number of erasures exceeds this limit, then the decoder fails. This means that in order to guarantee the successful decoding the transmitter needs to know the maximum number of erasures in a period of $n$ transmission. Therefore, tracking channel quality is required. Moreover, the encoding and decoding scheme needs to be customized. In another sense, the RS code does not have the desirable universality property. This will degrade the performance in fast dynamically changing channels even more since in that case the rate of the code needs to be set according to the worst possible channel quality to keep the transmission active in all the instances, or the transmitter and the receiver need to keep changing the encoding and decoding scheme constantly over the period of data

transmission.

Now, considering the scenario of broadcasting to multiple receivers with different channel qualities again, will reveal even more problems. In this case even if the channel quality of all the receivers is known and does not change rapidly, yet the RS code rate has to be set according to the worst channel quality among all to make sure that even the weakest one will be able to decode successfully. This again translates to a lot of rate loss for all the strong receivers.

Furthermore, RS codes also suffer from their high computational complexity. According to [12], these codes are practical only for small $k,\ n$, and $q$, since the standard implementation of encoding an decoding has a computational cost of the order $\mathcal{O}(k(n-k)\log_2(n))$ packet operations. Then the idea of setting the coding rate according to the worst channel quality will be penalized by a super linear complexity increment. In simple words, the strong receivers will have to pay extra not only for reception delay while unnecessary extra redundant packets are transmitted but also in terms of computational complexity for decoding which they will not really benefit from.

According to the discussion above, we seek a coding scheme which has the universality property along with the forward erasure correction simultaneously. In the next section we will introduce the idea of "Fountain codes", first presented in [2, 3], and the two most well known practical coding schemes designed based on this idea: LT codes [17] introduced by Luby, and Raptor codes [4, 5] introduced by Shokrollahi which all show the required properties very well.

## 2.4 Fountain Codes and Rateless Property

The idea of Fountain codes, called the "*digital Fountain*," is to design a coding scheme in which the transmitter provides a theoretically endless stream of output symbols. Each symbol provides a partial information about the $k$ original information symbols, and any subset of $k$ (or slightly larger) received symbols provides enough information for the decoder to decode the whole $k$ information symbols. When the receiver finally collects enough symbols to be able to perform the decod-

ing successfully, it will send a single bit feedback message to the transmitter, and announces the termination of its demand. Therefore, Fountain codes do not have a predefined fixed coding rate and hence they are also called "*rateless*" codes.

As a result of such properties, the transmission process will have the universality property since the erasure probability of the channel will just affect the average number of transmissions required to guarantee the reception of a subset of $k$ received symbols. The encoding and decoding scheme as well as the average required number of receptions will remain the same for all the different channel qualities.

In addition to the universality property, Fountain codes will also provide a nearly MDS property as well. As described in the previous section, an MDS code has this property that any subset of $k$ encoded symbols in an $n$-symbol long encoded block suffices for the successful decoding, and then the code is capable of fixing any number of erasures less than or equal to $n - k$. In Fountain codes although there is no fix predefined length $n$ for the encoded block, the MDS property holds in the sense that any subset of size $k$ or slightly larger than $k$ of received symbols will enable the decoder for recovering the original $k$ information symbols.

One simple idea to generate such a Fountain code is to simply let every output symbol be a random linear combination of the original $k$ message symbols in which any information symbol appears with a uniformly randomly chosen coefficient. Since we assume the information symbols belong to an alphabet of size $q$, and we need to define multiplication and addition over the set of information packets, a natural choice for the symbol's alphabet is then a Galois field of size $q$ which will be referred to as $GF(q)$. Obviously then the coefficients for the random linear combinations will also be selected from the same alphabet.

There are a lot of approaches to inform the receiver about the coefficients of each linear combination. One solution is to use a common random generator whose state is known by the transmitter as well as every receiver. Another solution which is more practical in the packet networks is to include the information about the coefficients of linear combination in the header of each output packet. Note that in the latter case, each output packet will be a random linear combination of the

original $k$ information packets, while each packet itself consists of a large number, say $d$, of symbols from $GF(q)$. Therefore, when the number of symbols in each packet is much larger than the total number of packets, (i.e. $k << d$), then the header which contains $k$ coefficients of $GF(q)$ will be negligible compared to the size of output packet.

Now lets check how the simple idea of random linear combinations will provide the properties of a Fountain code. As a Fountain code, each output symbols of this coding scheme will provide a partial information about the original $k$ information symbols which is the corresponding random linear combination of them. Moreover, the number of transmissions required for successful recovery is not determined by the coding scheme and depends on the probability of erasure in the channel. In other words, this scheme is a rateless coding scheme. The number of output symbols required by the receiver to be able to decode the original $k$ information symbols is equivalent to the number of random linear combinations required for successful decoding. Obviously, successful decoding in this scheme means solving a system of linear equations in terms of the $k$ information symbol. Hence, the necessary condition for successful decoding translates to having a set of linear combinations received, which forms a full rank linear equation system. In other words, the receiver needs to receive enough symbols to have at least $k$ linearly independent linear combinations. Restating the Theorem 3.1 in [6] in the context of our discussion, we have the following result. Assuming the receiver has received $(1 + \varepsilon)k$ output symbols of such coding scheme, the probability of successful decoding is lower bounded as

$$\Pr[\text{successful decoding}] = \Pr[\text{rk}(A) = k] > 1 - \frac{1}{(q-1)q^{\varepsilon k}}.$$

where, $A$ is the coefficient matrix of the linear equation system corresponding to the set of received symbols, and $\text{rk}(A)$, is the rank of matrix $A$. Moreover, through this theses we will use the notion of "reception overhead" to represent the value of the quantity $\varepsilon$ as described above.

As a result, for any arbitrarily small reception overhead $\varepsilon$, an appropriate choice of the code alphabet size $q$ decreases the probability of failure in decoding to any

desirable value. Furthermore, as the number of information packets $k$ tends to infinity, with a small non-zero overhead, even for a small alphabet size $q$ the probability of decoding failure tends to zero.

This means that all the properties of a Fountain code is satisfied with this simple scheme. However, this scheme is not a practical solution due to the complexity of the decoding algorithm in this scheme. As mentioned above, decoding of this scheme is indeed a matrix inversion over the coefficient matrix of the linear equation system corresponding to the received symbols. The complexity of this decoding algorithm hence is known to be of the order of $\mathcal{O}(k^3)$ operations in $GF(q)$. Knowing that this code operates well (i.e. with small overhead), when $q$, and $k$ are large, this complexity makes the introduced scheme impractical.

The importance of the simple random linear Fountain coding scheme mentioned above is to prove the possibility of designing Fountain codes representing all the properties in their definition. The next two sections will briefly explain the structure of LT and Raptor codes as the most well-known instances of Fountain codes with practically tractable computational complexity.

The set of good Fountain codes with desirable properties is not limited to the LT and Raptor codes. Following the principals of digital Fountain design many other researchers have studied the design of Fountain codes both for basic practical settings [18, 19], as well as for specific applications such as unequal error protection or streaming [20–22].

## 2.5 LT Codes

Luby transform codes, or LT codes, are the first practical Fountain codes. Similar to the random linear Fountain codes introduced in the previous section, an LT code also works by transmitting a theoretically unlimited number of linear combinations in the form of output symbols. Based on the same discussion, all the properties of a Fountain code are achieved by the LT codes as well.

Let us define the degree of a linear combination of a set of variables $x_1, \ \ldots, \ x_k$ as the number variables contributing in the linear combination with non-zero coef-

ficients. The heart of LT code design is in the distribution of degrees of its random linear combinations. We will also interchangeably use the expression "*degree of output symbol*," to refer to the degree of the linear combination corresponding to that output symbol. As we will discuss later in this section, this specific random structure will enable an alternative version of decoding with a computational complexity of the order $\mathcal{O}(k\ln(k))$. Hence, comparing to the much more complex Gaussian elimination decoding used in the random linear Fountain codes, which was suffering from its $\mathcal{O}(k^3)$ complexity, LT codes are practical for a much larger information block length $k$.

### 2.5.1 Encoding

The goal of the LT encoder is to produce a stream of output symbols, where, similar to the random linear Fountain codes, each output symbol provides the information equivalent to a linear equation in terms of a subset of the original $k$ information symbols. Moreover, to enable a more efficient decoding in LT codes, the degrees of the linear combinations corresponding to the received symbols need to follow a specific distribution. This distribution is called the "*robust Soliton*" distribution, and we will discuss it in more details in subsection 2.5.3. Since the set of received symbols will be a random subset of all the transmitted symbols, to implement the required statistic in the set of received symbols we just need to implement it in the set of all transmitted symbols.

In order to achieve the goals of encoding, LT encoders perform a two step process for producing each output symbol as follows. In the first step the encoder samples a random variable $d$ with the robust Soliton distribution. The outcome of this sampling will then obviously inherit the distribution and will be used as the selected degree for the output symbol.

The second step consists of selecting $d$ information packets, $x_{i_1}, \ldots, x_{i_d}$, and a vector $a = [a_1, \ldots, a_d] \in GF(q)^d$ both uniformly at random. Finally the output

symbol $y$ will be produced as follows:

$$y = \sum_{j=1}^{d} a_j x_{i_j},$$

where all the summations and multiplications are done in the $GF(q)$. This process is then iteratively repeated by the transmitter to produce and transmit as many output symbol as needed until it receives an acknowledgement from the receiver(s) confirming the successful recovery of the whole message, or the data to be transmitted is outdated.

In order to calculate the complexity of this encoding scheme we need to know the total number of output symbols required for successful transmission. This quantity obviously depends on the average probability of erasure over the transmission period. It is then easy to see than the total number of transmissions required for having a $(1+\varepsilon)k$ received symbols converges to

$$\frac{(1+\varepsilon)k}{\mathbb{E}[p_\epsilon]}, \tag{2.1}$$

for large $k$. In the above equation, $\mathbb{E}[\cdot]$, denotes the statistical average operator, and as shown in [17], for the LT codes, the overhead $\varepsilon$ scales with $k$ as

$$\mathcal{O}\left(\frac{\ln^2\left(\frac{k}{\delta}\right)}{\sqrt{k}}\right), \tag{2.2}$$

where $\delta$ is the probability of failure. Moreover, as we will show in subsection 2.5.3, the average degree of an output symbol in LT codes is of $\mathcal{O}(\ln(k))$. Hence, the average number of operations in $GF(q)$ required for each output symbol, which we will refer to as the complexity per output symbol, is of the order $\mathcal{O}(\ln(k))$. Now from (2.1) and (2.2) we derive the total average complexity of encoding for an LT code is of the order

$$\mathcal{O}\left(\frac{\left(\sqrt{k}+\ln^2\left(\frac{k}{\delta}\right)\right)\sqrt{k}\ln(k)}{\mathbb{E}[p_\epsilon]}\right) = \mathcal{O}\left(k\ln(k)\right). \tag{2.3}$$

### 2.5.2 Decoding

The decoding process in LT codes, is equivalent to solving a linear system of equations in an efficient way with a complexity as low as $\mathcal{O}(k\ln(k))$. The basis of this

decoding algorithm is to iteratively reduce the linear equation system through back substituting values of the recovered information symbols in the equations corresponding to the received symbols, and recover the value of new information symbols from the equations which will reduce to trivial degree-one form.

Description of this process could be put in several different ways including the graph theory representation, or in the context of matrix form representation as done in the next chapter. In this subsection we will describe it using the graph theory literature since this will help for the analysis behind the design of the degree distribution which will be briefly explained in the following subsection. In this regard, we first need to go through some notations.

The main concept in this context is the "*decoding graph*". Assume, we have received a set of $(1+\varepsilon)k$ symbols $r_1, \ldots, r_{(1+\varepsilon)k}$. The decoding graph is a bipartite graph with two vertex sets called "*input nodes*", and "*output nodes*". Input nodes consists of $k$ vertices, corresponding to the $k$ information symbols, and similarly, the other vertex set, output nodes, consists of $(1+\varepsilon)k$ vertices, corresponding to the received symbols. Each output node is connected to an input node if and only if the information symbol corresponding to that input node has a non-zero coefficient in the corresponding linear combination. With this connectivity rule, the degree of each output node is equivalent to that of its corresponding received symbol.

The other notation used in the graph representation of LT code's decoding algorithm is the set of (reduced) degree-one output nodes called the "*ripple*".

At the beginning, we first initiate the ripple with all the degree-one received symbols. As one can infer from this method of initiation, the degree distribution used at the transmitter for producing the output symbols needs to provide a non-zero probability for degree one since otherwise this decoding algorithm will not be initiated. Consequently, as long as the ripple in not empty, the decoder iteratively performs the following steps.

First note that each degree-one output node $r_i$ in the ripple has a single $x_j$ neighbour among the input nodes. Besides, since $r_i$ is of degree one, its value (upon a known coefficient) reveals the value of its neighbour $x_j$. Hence, in the first step the

value of any input node adjacent to a degree-one output node in the ripple will be recovered. In the second step, since the degree-one output nodes do not provide any further information about the value of any other input node, and hence they are not useful for the rest of decoding, we simply delete them from the decoding graph after recovering the value of their neighbours. Having the value of some of the input nodes recovered, in the third step we substitute their values in the linear equations of the remaining output nodes and reduce their degrees. To keep the decoding graph updated with this process, we simply delete them along with all of their edges from the graph. This in turn will reduce the degree of the remaining output nodes. The reduction of degrees in the remaining output nodes provides a positive probability of achieving new reduced degree-one output nodes to refill the ripple, which forms the fourth step in a single iteration of this decoding.

A careful selection of the output node degree distribution, as shown in [17], will guarantee that with a reception overhead of order $\mathcal{O}\left(\ln^2\left(\frac{k}{\delta}\right)\sqrt{k}\right)$, the ripple will always remain non-empty through the iterations of decoding. While the ripple is not empty, the decoder repeats the iterations and recovers more information symbols. When the ripple gets empty, if the decoding is not accomplished yet, the receiver should receive more symbols from the channel. The linear equations corresponding to each new received symbol will first be reduced by substituting the already recovered information symbols. Then, the reduced equation will be added to the graph as a new output node with the corresponding reduced set of input nodes neighbours. As soon as a new (reduced) degree-one output node is added to the decoding graph, the decoder refills the ripple and resumes the decoding. Otherwise, if for any reason the transmission is terminated before the decoding is finished, then the decoder will report a failure.

The operations of the decoder as described above are modelled by deletions of the edges of decoding graph. This decoder is hence named the edge deletion decoder. One can however think of the same decoder working by tracking messages over the edges of the same original decoding graph. In this alternative view we may assume that in each iteration edges first transmit zero/one messages from output

nodes to the input nodes and then vice versa with the following rules. Initially all the messages are zero except for the messages from degree-one output nodes transmitted to their input node neighbours. The messages will be updated in each iteration according to the following rules. An edge will take a message one from its output node end to its input node end if and only if the output node end has received a one from any other adjacent edge in the previous iteration. Otherwise it will take a zero message. Consequently, an edge will take a message one from its input node end to its output node end if and only if the input node end has received a one in the first step of this iteration. Otherwise it will take a zero message. It is now easy to see that this decoding algorithm is a version of the so called message passing decoder [23].

One can conclude that sending a message one from an output node to an input node represents that the value of that input node can be deduced by the information available at that output node at the previous iteration, and is equivalent to deleting that edge in the edge deletion version. Similarly, sending a message one from an input node over all of its edges represents that the value of this input node has been deduced and now can be substituted in all the output node. Obviously, now completion of decoding happens when all the input nodes receive at least a one message on some edge or equivalently, messages transmitted over edges in the decoding graph are all one.

Calculating the order of complexity for this decoding algorithm is easier using the edge deletion version. The total number of operations in $GF(q)$ required is equivalent to the number of edges in the graph upon a constant coefficient. Assuming the total number of received symbols is $(1+\varepsilon)k$, where $\varepsilon$ scales with $k$ according to (2.2), the complexity of decoding is of order

$$\mathcal{O}\left(\left(\sqrt{k}+\ln^2\left(\frac{k}{\delta}\right)\right)\sqrt{k}\ln(k)\right)=\mathcal{O}\left(k\ln(k)\right).$$

In the next subsection we will briefly review some of the intuition behind the design of Soliton distribution, and some of its properties.

### 2.5.3 Degree Distribution Design

In this subsection we will review some of the properties of the degree distribution used in the encoding of LT codes. Through these properties, a brief intuition about the design of a good degree distribution could be derived. We will not go into much details since this will be discussed in more details in the next chapters, where we basically propose our modifications and contributions as well. There has been a large amount of research done on the design of good degree distributions for Fountain codes with different properties over the last years. For a detailed discussion on the design of Soliton degree distribution, one can see [4, 6, 17].

Assume the receiver has received $(1+\varepsilon)k$ symbols. For a randomly chosen information symbol, the necessary condition to be recoverable by the decoder is that it should have been participated with a non-zero coefficient in at least one of the received symbols. In other words, the receiver will not be able to recover the value of an information symbol if its value has not appeared in any of the linear equations corresponding to the received symbols. To put it in the graph theory context used in the description of the decoding process, any input node needs to be connected to at least one of the output nodes in the decoding graph. Now, knowing that the neighbours of any output node are selected uniformly at random at the transmitter, we will derive a necessary condition on the average degree of a good degree distribution. The total number of edges in the decoding graph is equivalent to

$$(1+\varepsilon)k\bar{d},$$

where $\bar{d}$ denotes the average degree in the degree distribution. For a randomly chosen input node, according to the uniformly random selection of information symbols at the transmitter, the probability of being connected to an output node of degree $d$ is $(1-\frac{d}{k})$. Now assuming the independence of connection to any of the edges we can derive a good lower bound on the probability of having degree zero (i.e. not connected to any of these edges). This probability can be lower bounded

as

$$(1-\frac{1}{k})^{(1+\varepsilon)k\bar{d}},$$

which can be well approximated as

$$e^{-\bar{d}(1+\varepsilon)}. \tag{2.4}$$

Therefore, its a reasonable goal in the design of degree distribution to keep this probability bounded by a function of $k$, converging fast enough to zero as the number of information symbols $k$ grows, e.g., as $\frac{1}{k}$. Now to do so, the average degree $\bar{d}$ needs to be of the order $\mathcal{O}(\ln(k))$.

Rather than the average degree, obviously the ratio between different degrees in the degree distribution used for encoding will affect the dynamics of the decoded portion of information symbols over the iterations of decoding process. The dynamics of decoding process along with its dependency on the degree distribution structure was well studied in the context of "*AND-OR tree analysis*" [24] and also in the context of "*hyper-graph collapse process*" in [25]. The relations between these two approaches have been noted by several researcher such as in [26]. Here we will briefly explain this using the first approach. To do so it is beneficial to represent the degree distribution using a polynomial named the "*generating polynomial*," $\Omega(x) = \sum_{i=1}^{D} \Omega_i x^i$. In this representation, $\Omega_i$ will denote the probability of generating (or equivalently receiving) an output symbol of degree $i$. Hence, one can easily see that $\Omega_i \geq 0,\ \forall i$, and $\sum_{i=1}^{D} \Omega_i = 1$.

Similarly, the degree distribution of the input nodes can also been represented by a generating polynomial. As shown in [4], the degree distribution of the input nodes in the decoding graph can be derived as

$$\iota(x) = \left(1 - \frac{\bar{d}(1-x)}{k}\right)^{(1+\varepsilon)k}. \tag{2.5}$$

Now assume that at an arbitrary iteration $i$ of the decoding process, the portion of recovered information symbols is $x_i$ for some $1 \geq x_i \geq 0$. According to the message update rules described in the previous subsection, one can easily deduce

that the average portion of recovered information symbols in the next iteration will be derived as

$$x_{i+1} = \iota\left(1 - \frac{\Omega'(1-x_i)}{\Omega'(1)}\right), \tag{2.6}$$

where $\Omega'(x)$ denotes the derivative of $\Omega(x)$. Now from (2.5), we have

$$\iota\left(1 - \frac{\Omega'(1-x_i)}{\Omega'(1)}\right) \leq e^{(1+\varepsilon)\Omega'(1-x)}.$$

As a result of (2.6) and (2.5), a sufficient condition for the decoding to start with a non-zero portion of degree-one output nodes $x_0$ and accomplish the decoding successfully could be derived as follows:

$$e^{(1+\varepsilon)\Omega'(1-x)} \leq x,\ x \in [x_0, 1].$$

This is in turn equivalent to

$$-\ln(1-x) \leq \Omega'(x),\ x \in [0, 1-x_0].$$

Now integrating both sides, and using the Taylor series expansion of the right side at the equality, results in

$$\Omega(x) = \sum_{i=2}^{\infty} \frac{1}{i(i-1)} x^i.$$

This is called the "*ideal Soliton*" distribution. It has been first shown by Luby in [17] that this distribution will theoretically represent all the required properties for the LT codes. However, it has some practical drawbacks. First, it is an unbounded degree distribution, while for any finite number of information symbols, $k$, the maximum possible degree for output nodes will be $k$. Moreover, it has been shown that the good property of ideal Soliton in refilling the ripple is very sensitive to the variance. In other words, a small variation in the behaviour of the decoder from the theoretical average case can lead to an empty ripple. And finally, the ideal Soliton does not produce any degree-one output symbol, and hence the decoding will never start.

To cure the drawbacks of ideal Soliton, Luby proposes to perform the following modifications:

i) Truncate the distribution at some specific point so that it will be practical for finite number of information symbols.

ii) Add some more weight to the probability of smaller degrees so that it will provide more reduced degree-one output nodes as the decoding starts and produce a larger ripple.

iii) Add a relatively notable weight to some high degree to keep the average degree and the coverage at the required point.

iv) Add a non-zero probability to the choice of producing degree-one output symbols to provide a non-empty initial ripple.

The result makes the decoding process more robust to small variations from the average behaviour. This new degree distribution is then called the "*robust Soliton*" distribution. Putting it in the mathematical language, the robust Soliton distribution is formulated as follows:

For two parameters $c$, and $\delta$, take

$$s = c \ln(\frac{k}{\delta})\sqrt{k},$$

where $c$, and $\delta$ are chosen such that $\frac{k}{s}$ is an integer.

Then the robust Soliton degree distribution is given by

$$\Omega_i = \frac{\Theta_i}{\sum_{i=1}^{k} \Theta_i},\ i = 1, \ldots, k\ ,$$

where $\Theta_i$ is given by

$$\Theta_i = \begin{cases} \frac{1+s}{k} & i = 1 \\ \frac{1}{i(i-1)} + \frac{s}{ik} & i = 2, \ldots, \frac{k}{s} - 1 \\ \frac{1}{i(i-1)} + \frac{s}{k} \ln(\frac{s}{\delta}) & i = \frac{k}{s} \\ \frac{1}{i(i-1)} & i = \frac{k}{s} + 1, \ldots, k \end{cases} .$$

In the next section we will briefly review the main idea behind the design of Raptor codes. We will also look through the basic advantages of Raptor codes over LT codes, but we leave the details of their design to the later chapters where we also present some new contributions to their design.

## 2.6 Raptor Codes

LT codes, already reviewed in the previous section, provide a lot of interesting properties. The main advantage of LT codes over the dense random Fountain codes introduced in section 2.4 is to achieve almost all the good properties of Fountain codes (except for the fact that they need a slightly larger overhead) while keeping the encoding and decoding complexity as efficient as of $\mathcal{O}(k\ln(k))$. However, the complexity-overhead trade-off achieved by the LT codes is not the best possible. To make it more clear, lets take a look at a typical average intermediate performance of LT codes. Figures 2.2 and 2.3 show the progress in the number of recovered information symbols versus the number of received symbols for an LT code with $k = 65536$.

These figures, as a typical examples, reveal a fact about the decoding of LT codes. Although a significant portion of input symbols become recoverable with a very small overhead, but, there are always a few rare input nodes which remain disjoint from all the received symbols and will hence remain unrecoverable until a significant overhead is received. In order to avoid this significant overhead, Shokrollahi introduced the brilliant idea of using an outer code [4, 5]. A similar idea was also independently introduced by Maymounkov in [18].

Based on the design proposed in Raptor codes, the transmitter first uses a linear-complexity erasure correcting outer-code of rate $R$ to encode the $k$ information symbols to $k' = kR^{-1}$ "*intermediate symbols*". It is assumed that both the encoder and decoder share the complete information about the structure of the outer-code in use. As a result, decoder will be able to use the redundancy implemented by the outer-code to help the recovery of rare information symbols, which do not show up in the linear equations corresponding to the received symbols.

The choice of outer-code could vary and will not affect the design, but the original design proposes the use of capacity approaching erasure correcting LDPC codes such as those introduced in [27–29].

After performing the outer-encoding in the transmitter, the rest of transmission process will be totally equivalent to performing an LT coding over the $k'$ interme-

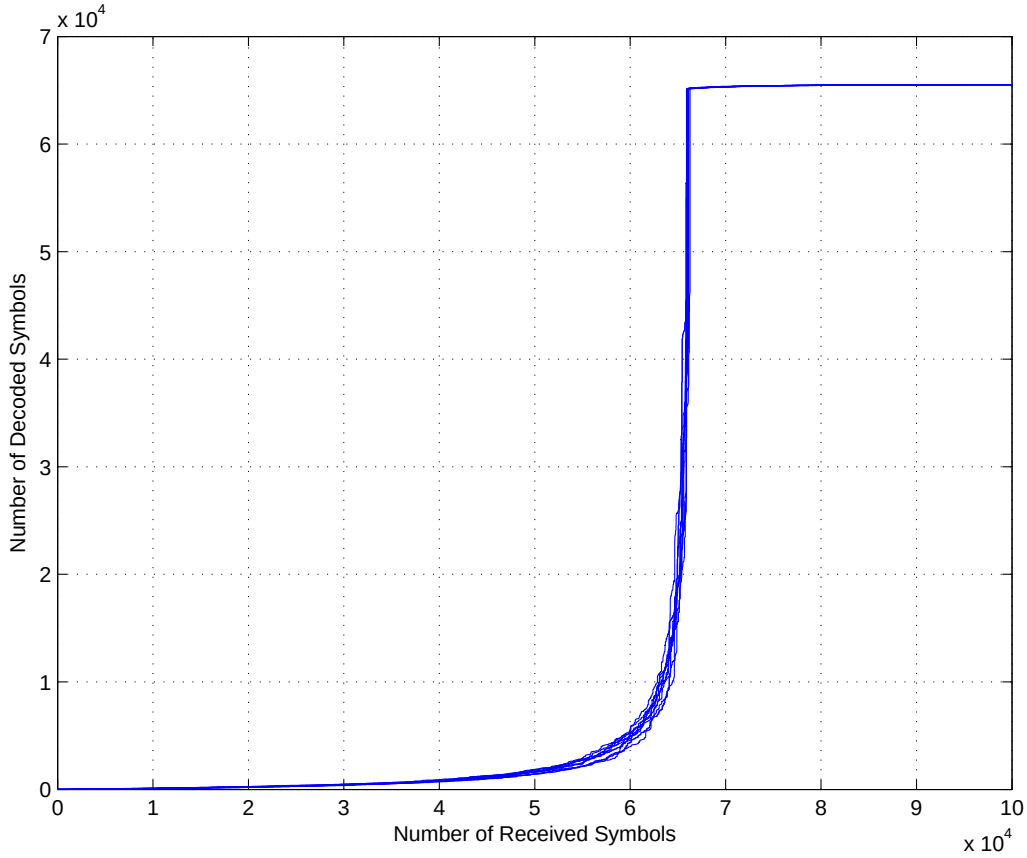



Figure 2.2: Samples of decoding progress in terms of the number of decoded symbols with respect to the number of received symbols for LT codes for $k = 65536$

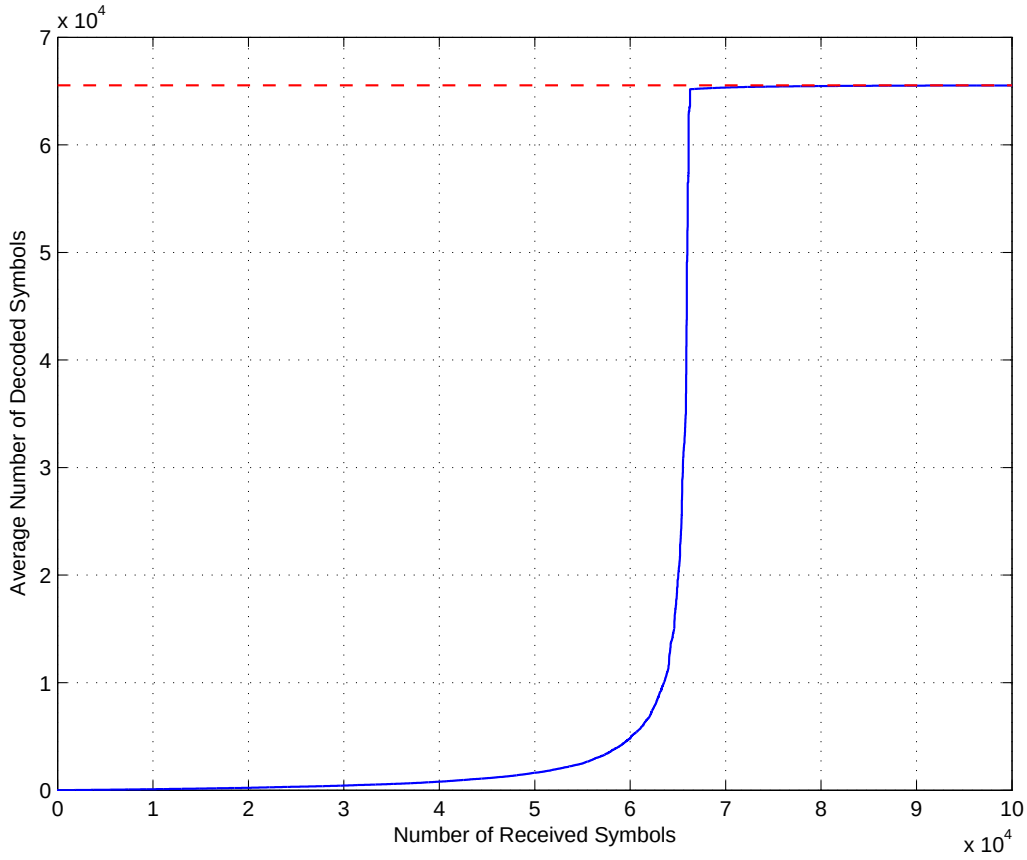



Figure 2.3: Average decoding progress in terms of the number of decoded symbols with respect to the number of received symbols for LT codes for $k = 65536$. The dashed red line shows the total number of input symbols.

diate symbols. The only difference is in the choice of degree distribution. As we will discuss later in this section, the degree distribution used for the Raptor codes can benefit from having a finite average degree not depending on the number of symbols to be encoded (i.e. $k$ or $k'$). This will result in a finite average number of operations in $GF(q)$ per output symbol for encoding. In other words the total complexity of encoding for Raptor codes then will be of the order $\mathcal{O}((1+\varepsilon)k)$, which is linear in the information block size $k$, rather than the super linear total complexity of encoding for LT codes derived as (2.3).

In the receiver, the decoding will not have any changes compared to that of LT codes except for the fact that as soon as a portion of size $(1+\sigma)k = (1+\sigma)Rk'$ of the intermediate symbols are recovered, the receiver will not need to continue receiving new symbols. At this point, using the decoder of the outer-code, the receiver will be able to recover the few last missing intermediate symbols exploiting the dependences implemented among the intermediate symbols through outer-coding. Recovering the $k$ original information symbols will be easy at this point using the bijective mapping of the outer-coding rule.

The complexity of this decoding scheme is easy to evaluate following a discussion similar to that of Subsection 2.5.2. Again the complexity of the first phase is scaled by the number of edges in the decoding graph. Due to the finite average degree of the degree distribution in use, this complexity is of order $\mathcal{O}(1+\varepsilon)k$. Moreover, the complexity of the remaining phases are linear in $k$ as the outer-code in use has a linear complexity of decoding. Hence, the total decoding complexity remains linear in the information block size $k$.

Finally we briefly review how does a finite average degree suffices in the case of Raptor codes. Assume we have $(1+\varepsilon)k$ received symbols. Using (2.4), it is easy to find the probability of leaving a randomly chosen intermediate symbol isolated in the decoding graph. Therefore, the expected value of the number of isolated intermediate symbols can be easily derived as

$$k'e^{-(1+\varepsilon)\bar{d}}.$$

Hence, in order to have an average coverage rate of $(1+\sigma)R$ over the intermediate

symbols in the decoding graph, it is enough to set $\bar{d}$ as follows

$$\bar{d} = \frac{-\ln(1-(1+\sigma)R)}{1+\varepsilon}. \tag{2.7}$$

As one can see from (2.7), the required average degree in Raptor codes then is a finite value which does not scale with the information block size $k$.

# Chapter 3

# Annotated Raptor Codes

In this chapter, we introduce a variation of Raptor codes called the annotated Raptor codes which reduce the overhead of conventional Raptor codes while keeping the encoding and decoding complexity linear. Although design of these codes is out of the scope of this chapter, numerical examples are provided to demonstrate their lower overhead even without a fine optimization.

After a quick review of conventional Raptor codes and introducing our notations in Section 3.2, in Section 3.3 we provide our main idea for reducing the overhead while keeping the complexity unchanged. In Section 3.4, we describe the encoding and decoding of the proposed annotated Raptor codes which is continued by some general comments on the design of code parameters in Section 3.5. Finally in Section 3.6, a numerical example on a benchmark block length is presented. The chapter is summarized and concluded in Section 3.7.

## 3.1 Introduction

Fountain codes such as LT code [17] and Raptor codes [4] were originally designed to achieve the capacity on any binary erasure channel (BEC) with no channel information and at very low complexity. The decoding complexity of Raptor codes under edge deletion (ED) decoding [4, 17] is linear with the block length. Therefore, these codes are the natural choice for data transmission over channels with unknown or very fast changing properties. Raptor codes preserve many of their interesting properties over other channels such as the binary symmetric channel, additive

white Gaussian noise and fading channels [30–33]. These codes have been already adapted as the forward error correction code for multimedia broadcast/multicast services (MBMS) by the 3rd Generation Partnership Project (3GPP) [34].

In practice, it is well known that ED needs a small overhead for successful decoding. Specifically, to decode $k$ information bits, $k(1+\varepsilon)$ received bits are needed at the decoder, where $\varepsilon$ is referred to as the overhead. More specifically, even with the highly optimized designs a non-zero overhead is needed if using the low complexity ED decoding. In order to avoid this overhead or reduce it to a negligible amount, a more complex decoding algorithm is introduced for Raptor codes called the inactivation decoding [8], but this algorithm is practical only for small block lengths due to its non-linear complexity.

## 3.2 Background and Notations

In this section, we briefly review the encoding and decoding of conventional Raptor codes. Unlike what is most common in the literature of rateless codes, we use the matrix form rather than the graph representation. The matrix form is more suitable for explaining annotated Raptor codes later. In this section, we also introduce the notations and definitions that will be used later.

### 3.2.1 Encoding

The encoding starts with a fixed rate outer code of rate $R$ and a parity check matrix $\mathbf{H}_{(n-k)\times n}$ which encodes an information block of $k$ input bits into a block of $n = \frac{k}{R}$ encoded bits $b_1, \ldots, b_n$, called the intermediate bits. To produce an output bit, first, the encoder randomly samples an integer $m \in \{1, \ldots, D\},\ D \leq n$ from a probability distribution. This distribution is characterized by a generating polynomial

$$\Omega(x) = \sum_{i=1}^{D} \Omega_i x^i,$$

where $\Omega_i$ is the probability that $m = i$. The encoder then uniformly at random chooses a set of $m$ intermediate bits and produces an output bit by XORing them.

Output bits are produced and transmitted until enough bits are received by the decoder to recover the information bits successfully.

Each output bit can be viewed as a parity check equation on a subset of intermediate bits, where the parity value is transmitted on the channel. The outer code can also be viewed as a set of parity check equations on intermediate bits. Unlike before, these parity values are always zero, thus they need not be transmitted on the channel. The decoder can use all the outer code equations and any received output bit equation to form an equation system from which all the intermediate bits are recovered. The information bits are then obtained through a linear mapping from intermediate bits according to the outer code.

### 3.2.2 Edge Deletion Decoding

The decoder starts with a linear equation system consisting of the parity check equations of the outer code

$$\mathbf{HX} = \mathbf{O}_{(n-k)\times 1},$$

where $\mathbf{O}_{\ell_1\times\ell_2}$ represents an $\ell_1 \times \ell_2$ all zero matrix. At this point the set of recovered intermediate bits is still empty. Assuming the BEC with erasure rate $\delta$, with probability $1-\delta$ an output bit is received. Receiving each output bit $b_i$ enables the decoder to use $b_i$'s corresponding parity check equation.

Upon receiving an equation, the decoder will substitutes any recovered intermediate bits, and then adds the reduced equation to its equation system. Whenever a reduced equation is of weight one, the equation is put in a set called the ripple. For any equation in the ripple, the value of the intermediate bit is immediately known and can be substituted in every other equation. This procedure is called the elimination process. It is easy to check that the order of using ripple elements have no effect on the performance of the decoder. Note that during the elimination process, the weight of some of the rows of the coefficient matrix is reduced which could in turn result in achieving new equations of weight one, and refilling the ripple. If the ripple gets empty before all the intermediate bits are recovered, receiver will listen to the channel to receive more equations to refill the ripple.

After receiving enough bits for a successful ED decoding, we have the following linear equation system.

$$\begin{bmatrix} \mathbf{H}_{(n-k)\times n} \\ \mathbf{C}_{(1+\varepsilon)k\times n} \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathbf{O}_{(n-k)\times 1} \\ \mathbf{R}_{(1+\varepsilon)k\times 1} \end{bmatrix}, \tag{3.1}$$

where, $\mathbf{C}$ is the coefficient matrix of the parity check equations corresponding to the received bits, $\varepsilon$ is the overhead, $\mathbf{R}$ is the vector containing the value of the received bits and $\mathbf{X}$ is the set of unknown intermediate variables. After successfully finishing ED decoding, upon reordering the rows, we obtain the following matrix equation.

$$\begin{bmatrix} \mathbf{I}_n \\ \mathbf{O}_{\varepsilon k\times n} \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathbf{B}_{n\times 1} \\ \mathbf{O}_{\varepsilon k\times 1} \end{bmatrix}. \tag{3.2}$$

Here, $\mathbf{B} = [b_1, \ldots, b_n]^T$ is a vector, containing the recovered values of the intermediate bits. Note that the reordering is just required for simplifying the representation. In the real implementation, this reordering is not needed.

## 3.3 Main Idea

According to [4], even using highly optimized Raptor codes with very large block lengths, the overhead is nonzero. This means that some of the received bits will be useless. These received bits are represented as all zero rows at the end of ED decoding (see Eq. (3.2)). In other words, these rows contain no new information about the intermediate bits given the rest of equations, thus we call them the "overhead rows". Although, it is not possible to avoid the overhead rows, interestingly, we will see that it is still possible to embed new information bits in them.

To embed new information bits in overhead rows, we first add an auxiliary set of variables $a_1, \ldots, a_{n_a}$ to the binary equation system and extend the columns of the coefficients matrix. We refer to these auxiliary columns of the coefficient matrix and their corresponding set of variables as "A-columns" and "A-variables" respectively. Clearly, the A-columns are not all zeros. Thus, some of the output bits are now XORed with bits from the A-variables. We refer to this operation as "annotation". The details of this operation is presented in Section 3.4.1. As we will explain later, the A-variables themselves must be protected by a low-rate outer code. Let us

denote the $(n_a - k_a) \times n_a$ parity check matrix of this outer code by $\mathbf{H}^{(a)}$ and the encoded block by $\mathbf{A} = [a_1, \ldots, a_{n_a}]^T$.

As a result, in the decoding process the initial matrix form represented in Eq. (3.1) changes to

$$\left[\begin{array}{c|c} \mathbf{H}_{(n-k)\times n} & \mathbf{O}_{(n-k)\times n_a} \\ \mathbf{O}_{(n_a-k_a)\times n} & \mathbf{H}^{(a)}_{(n_a-k_a)\times n_a} \\ \mathbf{C}_{(1+\varepsilon')(k+k_a)\times n} & \mathbf{C}^{(a)}_{(1+\varepsilon')(k+k_a)\times n_a} \end{array}\right] \left[\begin{array}{c} \mathbf{X}_{n\times 1} \\ \hline \mathbf{X}^{(a)}_{n_a\times 1} \end{array}\right] = \left[\begin{array}{c} \mathbf{O}_{(n+n_a-(k+k_a))\times 1} \\ \mathbf{R}_{(1+\varepsilon')(k+k_a)\times 1} \end{array}\right].$$

In the above equations $[\mathbf{C}|\mathbf{C}^{(a)}]$ is the coefficient matrix of the parity check equations corresponding to the received bits where, $\mathbf{C}$ part represents the coefficients of the intermediate bits and $\mathbf{C}^{(a)}$ represents the coefficients of the annotation bits. Notice that $n_a$ extra intermediate bits, which carry $k_a$ new information bits, are now added to the system. Thus, $\varepsilon'$ represents the new overhead. Finally upon reordering of rows the final form after successful ED decoding is

$$\left[\begin{array}{c|c} \mathbf{I}_n & \mathbf{O}_{n\times n_a} \\ \mathbf{O}_{n_a\times n} & \mathbf{I}_{n_a} \\ \hline \mathbf{O}_{\varepsilon'(k+k_a)\times(n+n_a)} & \end{array}\right] \left[\begin{array}{c} \mathbf{X}_{n\times 1} \\ \hline \mathbf{X}^{(a)}_{n_a\times 1} \end{array}\right] = \left[\begin{array}{c} \mathbf{B}_{n\times 1} \\ \mathbf{A}_{n_a\times 1} \\ \mathbf{O}_{\varepsilon'(k+k_a)\times 1} \end{array}\right].$$

The details of ED decoding for an annotated equation system is provided in Section 3.4.2. Here, to make the main idea more clear, we present a toy example. Assume that we have a block of three bits $x_1, x_2, x_3$, and we produce output symbols of degrees 1 to 3 with equal probabilities. Now, if for example the receiver receives $r_1 = x_1 \oplus x_2,\ r_2 = x_1 \oplus x_2 \oplus x_3,\ r_3 = x_2 \oplus x_3$, and $r_4 = x_1$, then the ED decoding of intermediate bits will not perform any elimination process before receiving $r_4$. When $r_4$ is received, it goes to the ripple and ED decoding starts recovering the values of intermediate bits. The received equation system before performing elimination is

$$\left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array}\right] = \left[\begin{array}{c} r_1 \\ r_2 \\ r_3 \\ r_4 \end{array}\right].$$

It is easy to check that ED decoding will recover all the intermediate bits with this equation system and after ED decoding we have

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} r_1 \oplus r_4 \\ r_1 \oplus r_2 \\ r_2 \oplus r_3 \oplus r_4 \\ r_4 \end{bmatrix}.$$

In the above, obviously, the third row is an overhead row and contains no new information about the intermediate bits given all the other rows. But if we annotate some of the transmitted bits (say $r_2$, $r_3$ and $r_4$) with a single A-variable $a$, then the representation of equation system after receiving $r_4$ is

$$\left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline a \end{array}\right] = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}.$$

The ED decoding can start the elimination and recovery procedure at this point, if we perform the decoding only based on the intermediate bits and in terms of the annotated variable $a$. As a result, when the ED decoding of intermediate bits finishes, the resulting equation system has the following form

$$\left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline a \end{array}\right] = \begin{bmatrix} r_1 \oplus r_4 \\ r_1 \oplus r_2 \\ r_2 \oplus r_3 \oplus r_4 \\ r_4 \end{bmatrix}.$$

Notice that still the third equation does not play any role in the recovery of the intermediate bits, but this row can be used to recover the value of the A-variable $a$ as $a = r_2 \oplus r_3 \oplus r_4$. The A-variable can in turn be used to recover any intermediate bit which was computed in terms of the A-variable (in this case $x_1$ in the fourth row). This example shows that with the same number of received bits, it is possible to recover more intermediate bits using annotation. Of course this was a highly fabricated example, in which the overhead was reduced to zero. Clearly, we do not expect zero overhead in a practical setup. However, as will be seen, the annotation idea retrieves a portion of the overhead at no extra cost. In fact, in order to keep the decoding complexity unchanged per information bit, we will see that the decoding procedure used in this toy example is not desirable. In Section 3.4.2 we propose a revised version of ED decoding for annotated Raptor codes.

## 3.4 Annotated Raptor Codes

Ideally we prefer to perform the annotation such that it will not affect any of the desirable properties of the original Raptor codes. More specifically, we do not want to increase the complexity per bit (neither at the encoder nor at the decoder). Achieving this goal, however, requires careful annotation and decoding. To see why the trivial approach (similar to the one in the toy example above) may fail, note that when the ED decoder uses annotated rows as pivots in row operations, extra complexity is resulted from the 1's in the corresponding rows of the A-columns. Thus a high-density of 1's in the A-columns is against the goal of a low-complexity design. Unfortunately, even starting with sparse A-columns, the density of 1's in the A-columns gradually increases as ED decoding progresses. Our numerical simulation shows that the complexity will grow super linear with an approximate exponent of $1.3$. In the following, we briefly outline an annotation method that achieves linear complexity.

Let us assume that we could know beforehand which transmissions end up as overhead rows. If this knowledge existed, we could annotate only these transmissions. Although such a knowledge cannot exist in a real setup, we can annotate a small portion of rows and pretend that they will end up being the overhead rows. Thus, the decoding will start from the non-annotated rows. Our interesting observation is that if annotated rows are selected carefully, ED decoding of non-annotated rows will recover a large portion of intermediate bits. In other words, assume in a conventional Raptor code, we carefully select and mark a $\sigma_0$ portion of the transmissions for annotation. Then, in the receiver, we first exclude the marked received bits and perform ED decoding on the unmarked received equations and the parity check equations of the outer code. When the total number of received bits is close to the number of input bits, we observe that the decoder recovers a $(1-\delta_0)$ portion of the intermediate bits. Typically for $\sigma_0 = 0.05,$ we have $\delta_0 = 0.3.$

After recovery of $(1-\delta_0)$ portion of the intermediate bits, it is easy to see that with probability $(1-\delta_0)^i$, an annotated equation which originally contains $i$ intermediate bits, is reduced to an equation based only on the A-variables. We

call these reduced equations the "A-equations". From these A-equations a fixed portion of A-variables will be recoverable. Now, if the rate of the outer code of the A-variables is selected properly, it will be possible to decode all the A-variables. Consequently, it will be possible to de-annotate all the annotated rows in linear complexity. In fact to keep the complexity of this de-annotation at its absolute minimum, in this work, each annotated row has a single A-variable in it. This also keeps the encoding complexity linear.

Finally, after de-annotation, the rest of the intermediate bits will be recovered using ED decoding. In terms of ED decoding of the intermediate bits, the only difference between an annotated Raptor code and a conventional one is that here we have changed the order of using the received equations. We use some of the equations at first and postpone using the others (the annotated ones) for a while. Between these two phases, we recover some information bits that are embedded in the annotation.

In the next section we will go through more details of the encoding and decoding algorithms for the annotated Raptor codes.

### 3.4.1 Encoding

The encoding process in annotated Raptor codes has two separate steps. In the first step, two information blocks of length $k$ and $k_a$ are coded into two encoded blocks (i.e., the intermediate variables and the A-variables), using fixed rate outer codes with parity check matrices $\mathbf{H}_{(n-k)\times n}$, and $\mathbf{H}^{(a)}_{(n_a-k_a)\times n_a}$.

The second step, which contains two phases, will generate an output bit. First an integer $m \in \{1, \ldots, D\},\ D \leq n$ will be sampled based on a probability distribution represented by its generating polynomial

$$\Theta(x) = \sum_{i=1}^{D} (\Phi_i + \Psi_i) x^i.$$

Here $m = i$ happens with probability $(\Phi_i + \Psi_i)$. Consequently, based on the selected value of $m$, encoder samples another random variable $b \sim \mathcal{B}\left(\frac{\Psi_m}{\Phi_m + \Psi_m}\right)$, where $\mathcal{B}(p)$ represents the Bernoulli distribution with probability of success equal to $p$.

The encoder then chooses $m$ intermediate bits uniformly at random. If the Bernoulli outcome is success, a single A-variable bit is also selected uniformly at random. Finally, the XOR of all the selected bits forms an output bit for transmission. Output bits are generated and transmitted iteratively, until successful transmission of the whole data block.

### 3.4.2 Decoding

The decoding procedure has already been described earlier in this section. Here we summarize the procedure. Two separate edge deletion decoders are used. The first one decodes the intermediate bits, using the non-annotated equations and the rows of matrix $\mathbf{H}$. The second one decodes the A-variables using any row whose first $n$ elements are all zeros including the rows of matrix $\mathbf{H}^{(a)}$. Obviously, as the decoders recover some of the intermediate bits and A-variables they remove them from all the equations and hence each decoder may provide the other with some new equations to be used in the rest of the decoding process. When both the decoders run out of ripple, receiver listens to the channel to receive new equations and refill at least one of the ripples again.

## 3.5 Some Comments on Design

Assume that the decoder has already received $n = (1+\varepsilon)k$ bits. Moreover, assume that through numerical search we have obtained the probability distribution $\Theta(x)$ for which, excluding the annotated received bits, ED decoding is able to recover a $\delta_0$ portion of the intermediate bits. Based on the discussions in the previous section, the probability that a randomly selected row be reduced to an A-equation is

$$P^* = \sum_{i=1}^{D} \Psi_i (\delta_0)^i.$$

Therefore, the average number of A-equations released by ED decoding of intermediate bits excluding annotated equations, is $(1+\varepsilon)kP^*$. According to the single-bit annotation strategy taken in this thesis, the probability that a randomly selected

A-variable is not covered in the released A-equations is

$$(1-\frac{1}{n_a})^{((1+\varepsilon)kP^*)} \simeq e^{(\frac{-(1+\varepsilon)kP^*}{n_a})}.$$

Hence, the average number of A-variables which are now recovered is approximately

$$m_a = n_a\left(1-e^{\frac{-(1+\varepsilon)kP^*}{n_a}}\right). \tag{3.3}$$

It is seen from (3.3) that $m_a$ is an increasing function of $n_a$ and that $m_a < (1+\varepsilon)kP^*$. Therefore, the new overhead $\varepsilon'$ can be found as

$$\varepsilon' = \frac{\varepsilon k - m_a}{k+m_a}. \tag{3.4}$$

It is easily seen that $\varepsilon' < \varepsilon$ as long as $m_a > 0$ (i.e., $n_a > 0$). Moreover, $\varepsilon'$ is a strictly decreasing function of $n_a$. It means that as the number of A-variables increases, more information bits can be transmitted using annotation, and thus, a larger portion of the overhead can be retrieved. As a result the lower the rate of the outer code for A-variables, the smaller the overhead will be. The improvement in the overhead, however, is bounded because $m_a$ saturates as a function of $n_a$ (see Eq. (3.3)).

A very low rate outer code, however, introduces a significant source of complexity. Although there exist very good low rate codes with linear complexity such as LDPC codes designed for erasure channels [27–29], when the rate of these codes tend to zero, the coefficient of the linear complexity tends to infinity. Figure 3.1 depicts complexity per information bit, measured as the number of XORs needed for encoding/decoding of LDPC codes designed in [27]. This figure is based on codes that achieve 95% of the channel capacity.

To keep the complexity of annotated Raptor codes equal to that of conventional Raptor codes, we must use an outer code for the A-variables whose complexity per information bit is the same as conventional Raptor codes. The complexity per information bit of a conventional Raptor code is equal to the average weight of its output bits which is typically at least eight (considering the complexity of the

high-rate outer code). Thus, Fig. 3.1 suggests that the A-variables must be encoded using an outer code of rate around 0.25. Obviously, lower rate codes can be used to retrieve a higher portion of the overhead, but at the cost of a higher complexity per information bit. This extra complexity, however, is quite small since it affects only the parity check equations of the A-variables, which represent a small fraction of all equations (typically less than 4%). Nonetheless, for any fixed rate outer code, the complexity remains linear.

Now assume we have selected an outer code of rate $R_a$ for A-variables which guarantees successful decoding of A-variables for erasure rates less than $1 - R_a$ with high probability. According to the above discussions, we can now select the number of information bits $k_a$ to be encoded to $n_a$ A-variables as $k_a = n_a R_a$, where $n_a$ must satisfy

$$R_a < 1 - e^{(\frac{-(1+\varepsilon)kP^*}{n_a})}.$$

Thus we have

$$k_a < R_a \frac{-(1+\varepsilon)kP^*}{\ln(1-R_a)}. \tag{3.5}$$

This equation can be used to choose the number of information bits to be encoded by the rate $R_a$ outer code and be used as A-variables for annotations.

## 3.6 Example Code and Numerical Results

This section provides a numerical example of an annotated Raptor code. As the optimization of the code is out of the scope of this thesis, our example here does not represent an optimal design. Indeed, in order to better justify the benefits of annotated Raptor codes, we focus on the impact of annotation on an existing probability distribution optimized for conventional Raptor code. Clearly, we expect even better results through optimizing a probability distribution for annotated Raptor codes.

Our focus in this example is on the highly optimized probability distribution $\Omega(x)$ presented in [4] for a Raptor code with an information block of $k$ =64,520 bits and an outer code of rate $R = 0.9845$ to produce a block of $n$ =65,536 intermediate

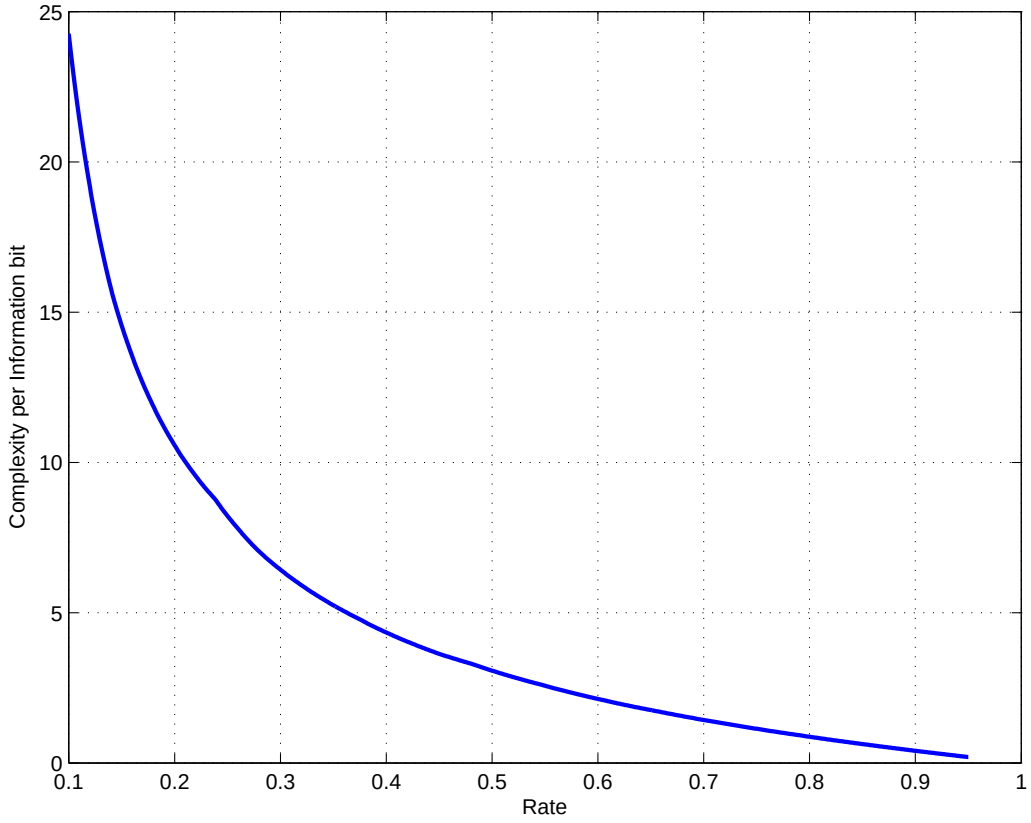



Figure 3.1: Complexity per information bit vs. rate for capacity approaching sequences of LDPC codes designed for the BEC.

| $i$ | $\Phi_i$ | $\Psi_i$ | $\Omega_i = \Phi_i + \Psi_i$ |
|---|---|---|---|
| 1 | 0.007969 | 0 | 0.007969 |
| 2 | 0.478570 | 0.015 | 0.493570 |
| 3 | 0.161220 | 0.005 | 0.166220 |
| 4 | 0.072646 | 0 | 0.072646 |
| 5 | 0.082558 | 0 | 0.082558 |
| 8 | 0.056058 | 0 | 0.056058 |
| 9 | 0.037229 | 0 | 0.037229 |
| 19 | 0.055590 | 0 | 0.055590 |
| 65 | 0.025023 | 0 | 0.025023 |
| 66 | 0.003135 | 0 | 0.003135 |

Table 3.1: Example Code with $k = 64520$ and $k_a = 800$.

bits. As mentioned before, we use a single bit annotation for the output bits that are selected to be annotated. This represents the simplest form of annotation. One may consider a degree distribution for the A-variables and optimize it for improved performance. Such optimizations, however, are out of the capacity of this chapter.

Based on a set of numerical experiments we selected the probability distribution presented in Table 3.1 for this example. Please notice that the third column represents the probability distribution of the Raptor code presented in [4]. The rate of the outer code of the A-variable is selected to be 0.25 to encode $k_a = 800$ information bits into $n_a = 3200$ A-variables. These A-variables are annotated to the 65,536 intermediate bits of the above mentioned Raptor code. Simulations show that the average overhead based on the annotation method introduced in this chapter is 3.4%. This amounts to 10% overhead reduction compared to the average 3.8% overhead of the original Raptor code. We emphasize that the complexity per information bit is exactly the same for both codes. It is worthwhile to mention that by using conventional Raptor codes, an overhead of 3.4% could not be achieved for block lengths less than 80,000 bits [4], which would involve a much more memory complexity.

## 3.7 Summery

Since Raptor codes need a reception overhead to be able to recover the information bits, some of the received bits are indeed never used in the process of decoding. In this chapter, we presented an extension of the well known Raptor codes showing that extra information bits can be embedded through careful annotation of a subset of transmissions. We then detailed the encoding and the decoding process of the proposed codes based on the changes made in the design of the original Raptor codes. Finally, we provided a numerical example verifying the improved performance even without optimization a probability distribution for the annotated Raptor codes. Finding the optimal probability distribution for the new encoding/decoding structure will reveal its full potentials.

# Chapter 4

# On Raptor Code Design for Inactivation Decoding

In this chapter, we study the design of Raptor codes for the BEC when inactivation decoding (ID) [8] is used. An ID decoder is essentially a maximum likelihood decoder with controlled complexity, which can accomplish the decoding with a smaller number of received symbols than any other decoder requires. Hence, ID is incorporated in 3GPP as a practical decoder [34]. Despite the rich literature on code design for the conventional edge deletion decoding (e.g., [4, 39, 40]), code design for ID has not yet received much attention.

In the remainder of this chapter, we first briefly review the encoding and decoding of Raptor codes, focusing on ID. In Section III, we introduce our code design, by proposing a new design criterion, and then we use this criterion for an analytical design. The numerical comparisons between the code used by the 3GPP and our proposed code are presented in Section IV.

## 4.1 Introduction

LT codes were originally introduced as the first practical Fountain codes in [17]. As such, LT codes are designed to transmit a theoretically endless stream of symbols until the receiver has enough symbols to decode all the information bits. Raptor codes [4], an extension of LT codes, employ an outer code to enable the receiver to recover the whole information stream from any sufficiently large subset of re-

covered intermediate symbols. This idea significantly improves the performance of LT codes, as the recovery of the last few percentages of the information bits, which could be very slow, is now done by using the outer code.

Raptor codes are able to asymptotically achieve the channel capacity on any binary erasure channel (BEC) without any channel state information at the transmitter or the receiver. This universal capacity-achieving property enables optimal performance even in time-varying channels. Accordingly, these codes are the natural choice for broadcasting/multicasting to a group of receivers with different and even unknown channel qualities. As a result Raptor codes have been adopted by the 3rd Generation Group Partnership Project (3GPP) to be used in multimedia broadcast/multicast services (MBMS) for forward error correction [34] and digital video broadcast-handheld (DVB-H) [35]. The desirable properties of Raptor codes have motivated many researchers to study their performance and design for other channels [30–32, 36]. Decoder design for Raptor codes has also been an active research area [8, 37, 38].

## 4.2 Encoding and Decoding of Raptor Codes

Encoding and decoding of Raptor codes have been discussed in the previous chapter in Subsections 3.2.1 and 3.2.2. Accordingly, we will not go through the details of the encoding process here, while we will use the same set of notations and definitions in this chapter. However, it worth to review the decoding process again here This time we will look at this process in the context of graph theory rather using the matrix form representation.

Decoding of the Raptor codes is performed in two separate steps. First the LT code is decoded, and then the outer code is decoded in the second step. Assuming that the outer code can recover the whole information block from any subset of $n(R+\sigma),\ \sigma > 0$ recovered intermediate bits, we focus our discussion on the LT decoder.

For LT decoding, a decoding graph [4] is formed based on the set of received symbols. The decoding graph is a bipartite graph with one vertex set corresponding

to the set of all intermediate bits, and the other set corresponding to the output bits (output nodes). Initially, each output node is adjacent to the group of intermediate nodes forming the corresponding received bits.

Various decoding solutions can be used. Gaussian elimination, although optimal, is typically too complex. A modified version of the belief propagation algorithm, called edge deletion decoding (EDD) [17], is an efficient alternative when an appropriate design of $\Omega(x)$ is performed. EDD requires a small overhead in the number of received symbols for successful decoding [41]. This algorithm uses degree one output nodes in the decoding graph to deduce the value of their neighbouring intermediate nodes, and then removes the recovered intermediate nodes to achieve new degree-one output nodes iteratively.

## Inactivation Decoding

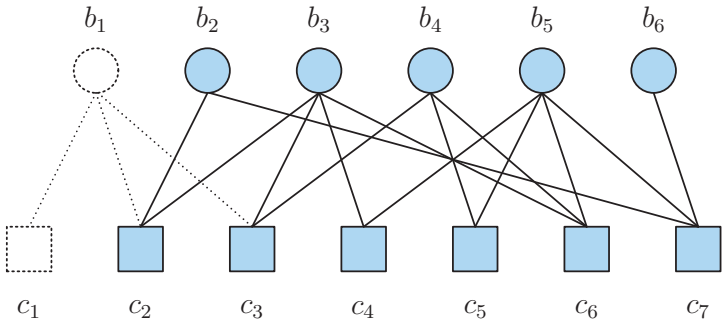
For moderate block lengths ($1024$ to $8192$ bits), which are of interest in applications supported by the 3GPP standard, a modified version of EDD, called inactivation decoding (ID), was introduced in [8]. The main difference between ID and EDD occurs when the set of degree one output nodes, called the ripple, becomes empty. In this case, the EDD stops until the ripple is refilled by receiving more symbols from the channel. ID, however, instead of waiting for more symbols, selects some unrecovered intermediate nodes in the remaining decoding graph and temporarily excludes them from the graph. This process is called inactivation. By inactivating some of the intermediate nodes, their edges will also be excluded temporarily, reducing the degree of some of the remaining output nodes. Thus, the decoder extracts some *reduced* degree-one output nodes, whose values can be found in terms of the inactivated bits. The decoder can now recover more intermediate bits (albeit, in terms of the inactivated bits) until the ripple is empty again, and another inactivation can be performed. Finally, the decoder uses Gaussian elimination for the inactivated bits and finishes the decoding by using a back filling process, which evaluates all the intermediate bits which have been recovered in terms of the inactivated bits.

In order to solve the subsystem of linear equations formed by the inactivated bits, this subsystem should be full rank. The subsystem has a high probability of being full rank because it is dense. However, if it is not full rank, the receiver receives more symbols to obtain more equations and remove the rank deficiency. This process results in a very small average reception overhead, which has been shown to be less than one percent in practice [6, 38].

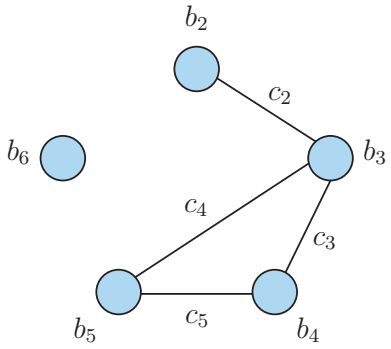
For selecting a node to be inactivated, many different strategies can be used [8]. One trivial choice is to randomly select an unrecovered intermediate node connected to a reduced degree two output node. We will refer to this strategy as "Random ID". Another strategy, introduced in [8], is to inactivate one of the nodes in the maximum connected component of $G$, where $G$ is the *degree-two induced subgraph* [6, 42] of the remaining decoding graph (see Fig. 1). Inactivating any of the bits in a connected component of $G$ causes the immediate recovery of all the other bits in that component. Hence, the second strategy, which we refer to as "Maximum Component ID," performs better than Random ID. Since $G$ is subject to change during the decoding process, the search for the largest connected component must be repeated during each inactivation step. Thus, Max-Component ID is considerably more complex than Random ID.

Degree distribution design for ID is considered in [6,42], where Max-Component ID is assumed, and the design criterion is to statistically guarantee the existence of a giant connected component in $G$ at each inactivation step. Having a giant connected component guarantees the recovery of a large portion of nodes at each inactivation step.

Accordingly, [6, 42] introduced a procedure that takes an $\Omega(x)$ and determines on average for how many inactivations a giant component will almost surely be present in $G$. The design problem is then to find a generating polynomial $\Omega(x)$, which will guarantee the existence of a giant connected component until the desired portion of the intermediate bits is recovered. This design criterion, in addition to a mixture of optimization methods, has been used to design a degree distribution which the 3GPP has adopted [34].




(a)




(b)

Figure 4.1: (a) The decoding graph with output nodes $c_1$ to $c_7$ and intermediate bits $b_1$ to $b_6$. The ripple is initiated with $c_1$, which recovers $b_1$ and then becomes empty. (b) The reduced degree two induced subgraph $G$ based on the remaining effective decoding graph. In $G$, every reduced degree two output node will represent an edge. Here, $G$ contains two connected components. The maximum component contains four nodes. Max-Component ID may, for example, choose $b_4$ as a node in the maximum component of $G$ for inactivation. This choice will refill the ripple with $c_3, c_5$, which, in turn, recover $b_3$ and $b_5$ in terms of $b_4$.

## 4.3 Degree Distribution Design

In this section, a new design criterion is proposed from which a more efficient $\Omega(x)$ for ID is designed. The basic difference between our design and that of [6, 42] is that [6, 42] aims to increase the portion of bits that are *guaranteed* to be recovered after each inactivation, whereas our design aims to increase the *average* portion of recovered bits after each inactivation. Notice that the actual number of recovered bits is usually more than the *guaranteed* portion. Thus, it appears reasonable to aim at increasing the average recovery.

From the discussion in Section 4.2, it is obvious that for a fixed decoder structure and with a constant performance for the outer code, all the properties of a Raptor code are characterized by the generating polynomial $\Omega(x)$. Similar to the case of design for the EDD, an infinite block length assumption is made for the analytical design of $\Omega(x)$. However, the performance of the finite length case is evaluated through simulations.

For a Raptor code under EDD, the main performance measure is the overhead. Under ID, however, the overhead may not be as meaningful because ID performs an inactivation instead of receiving extra symbols. As a result, a good measure of performance appears to be the number of required inactivations [38] and [6], which directly affects the decoding complexity. Therefore, our design goal is to reduce the number of required inactivations.

### 4.3.1 Evolution of $\Omega(x)$ During ID

In ID, after each inactivation, the remaining degree distribution changes. As a result, to study the average performance analytically, we need the remaining degree distribution, based on the original $\Omega(x)$ and the portion of the recovered intermediate bits $\delta$. Denoting the new degree distribution as $\Omega_\delta(x)$, we have $\Omega_0(x) = \Omega(x)$, and $\Omega_1(x) = 1$. Also, since the selection of the intermediate bits in the encoding is uniformly random, recovering a $\delta$ portion of intermediate bits is equivalent to deleting a randomly chosen $\delta$ portion of intermediate nodes in the decoding graph. Hence, we can assume that a random $\delta$ portion of the edges of the decoding graph

is also deleted. Therefore, for randomly chosen output node of initial degree $j$, will be of degree $i = 1, \cdots, j$ with probability $\binom{j}{i}(1-\delta)^i\delta^{j-i}$. Then, the average degree distribution of the output symbols in the remaining graph will be

$$\begin{aligned}\Omega_\delta(x) = \sum_{i=1}^{D} \Omega_{\delta,i} x^i &= \sum_{i=1}^{D} \left( \sum_{j=0}^{D-i} \Omega_{i+j} \binom{i+j}{j} (1-\delta)^i \delta^j \right) x^i \\ &= \Omega((1-\delta)x + \delta).\end{aligned}$$

As a result,

$$\Omega_{\delta,i} = \sum_{j=0}^{D-i} \Omega_{i+j} \binom{i+j}{j} (1-\delta)^i \delta^j. \tag{4.1}$$

### 4.3.2 Previous Results

Degree distribution design for ID is considered in [6, 42], where Max-Component ID is assumed and the goal of design is to statistically guarantee the existence of a giant connected component in the degree-two induced subgraph $G$ at each inactivation step. Having a giant connected component could guarantee the recovery of a large portion of nodes in each inactivation step. Obviously recovering one bit of any connected component in $G$ results in the recovery of the whole component.

Accordingly, [6, 42] introduces a procedure that takes an $\Omega(x)$ and determines on average for how many inactivations, almost surely there will be a giant component in $G$. Then the design problem is to find a generating polynomial $\Omega(x)$, which guarantees the existence of a giant connected component until the desired portion of the intermediate bits are recovered. This design criterion in addition to a mixture of optimization methods have been used to design a degree distribution which has also been adopted by 3GPP [34]. Assume that the initial generating polynomial is $\Omega(x)$, the receiver has received $(1+\varepsilon)k$ symbols, and an arbitrary $\delta$ portion of the intermediate bits have been recovered. Using a similar argument as in the previous subsection, the probability of having a reduced degree-two output node is

$$\sum_{i=2}^{D} \Omega_i \frac{i(i-1)}{2} (1-\delta)^2 \delta^{i-2} = \frac{(1-\delta)^2}{2} \Omega''(\delta).$$

Therefore, because every output symbol of reduced degree two results in one edge in $G$, the average sum of degrees in $G$ is $(1+\varepsilon)(1-\delta)^2\Omega''(\delta)$. Then the average degree of $G$ is

$$\bar{d}_{\Omega(x)}(\delta,\varepsilon) = \frac{(1+\varepsilon)k(1-\delta)^2\Omega''(\delta)}{(1-\delta)k} = (1+\varepsilon)(1-\delta)\Omega''(\delta). \tag{4.2}$$

It has been shown in [43] that in order to almost surely have a unique giant connected component in a random graph, it is necessary that its average degree is strictly greater than one. In addition, when the expected degree of all the vertices are equal, the size of this unique giant connected component is concentrated around the unique solution of the equation $1 - x - e^{-\bar{d}x} = 0$, where $\bar{d}$ is the expected average degree [44, 45].

Accordingly, [6, 42] introduces a procedure that takes an $\Omega(x)$ and determines on average for how many inactivations, almost surely there will be a giant component in $G$. The procedure is as follows: start with $\delta_0 = 0$, and iteratively for $i > 0$ set $\delta^{(i)} = 1 - \prod_{j=0}^{i-1}(1-\delta_j)$, then while $\bar{d}_{\Omega(x)}(\delta^{(i)},\varepsilon) > 1$ set $\delta_i$ the root of $1 - x - e^{-\bar{d}_{\Omega(x)}(\delta^{(i)},\varepsilon)x} = 0$, where $\bar{d}_{\Omega(x)}(\delta^{(i)},\varepsilon)$ is introduced in (4.2). Therefore, when $\bar{d}_{\Omega(x)}(\delta^{(i)},\varepsilon) \leq 1$, ID cannot guarantee recovering a significant portion of the remaining bits.

Then the design problem is to find a generating polynomial $\Omega(x)$, which is able to keep the procedure running until the desired portion of the intermediate bits are recovered. This design criterion in addition to a mixture of optimization methods have been used to design a degree distribution which has also been adopted by 3GPP [34].

### 4.3.3 A New Design Criterion

The new design is based on a new insight into the ID process. As mentioned in Section 4.2, ID starts with an EDD phase and works until the ripple is empty. At this point, inactivation is performed, and another phase of EDD is started. Thus, one can think of ID as a series of EDDs, each applied to a portion of the unrecovered bits. According to this view, we need a degree distribution that will perform well

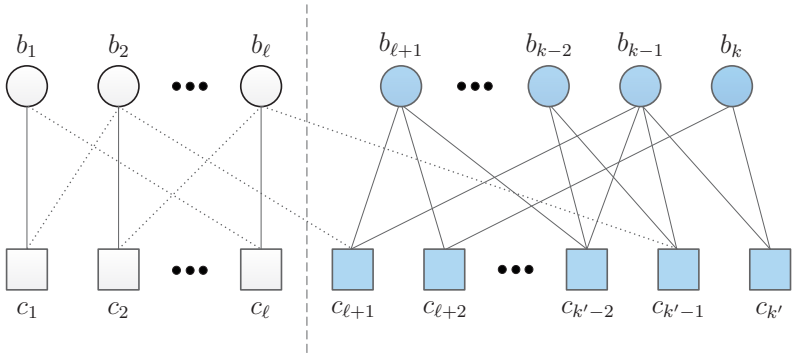



Figure 4.2: The decoding graph at some intermediate step of decoding. The receiver started the decoding after receiving $k' = (1+\varepsilon)k$ output symbols, where $\varepsilon$ is a very small positive number. Up to this point, a $\delta = \frac{\ell}{k}$ of the intermediate bits have been recovered. The left part shows the non-effective part of the decoding graph at this moment, which could not participate in the decoding of the remaining bits. The right part is still effective and contains $(1-\delta)k$ intermediate bits and approximately $(1-\delta)k'$ output nodes as well.

under a series of EDDs despite the recovery of any portion of bits. Designing such a distribution is a challenging task because the degree distribution for each EDD step may be different. Thus, the performance may differ in each step. A degree distribution which remains close to optimal in all EDD steps is, therefore, desired. Accordingly, we first investigate another effect of recovering a $\delta$ portion of the intermediate bits on the degree distribution of the output nodes.

An intermediate bit $b_i$ is recovered when the degree of an output node $c_{i'}$, connected to $b_i$, is reduced to one. In fact, the last edge of $c_{i'}$ connects it to $b_i$. After recovering $b_i$, the output node $c_{i'}$ can no longer be effective in the decoding process. Now, assume that the receiver has originally received $(1+\varepsilon)k$ symbols for a very small $\varepsilon > 0$ (in practice $\varepsilon < 0.01$). Therefore, after recovering a $\delta$ portion of intermediate bits, a $\delta$ portion of the output nodes will not be effective for the rest of the decoding process (see Fig. 2). In other words, decoding continues by performing EDD on the remaining decoding subgraph containing $(1-\delta)$ portion of output nodes and the unrecovered intermediate nodes.

Now, recall that $\Omega_{\delta,j}$ represents the fraction of reduced-degree $j$, $j \geq 2$ output nodes after recovering a $\delta$ portion of intermediate nodes. Accordingly, starting with

a close-to-optimal $\Omega(x)$, if for any $0 < \delta < 1$, $\Omega(x)$ satisfies

$$\forall j \geq 2, \;\; \Omega_{\delta,j} = (1-\delta)\Omega_j, \tag{4.3}$$

then the close-to-optimal performance for the next EDD step is preserved. This way, the code recovers a large portion of bits in each EDD step. Thus, we use (4.3) as a design criterion.

### 4.3.4 The Proposed Code Design

According to (4.1),

$$\Omega_{\delta,i} = \frac{(1-\delta)^i}{i!} \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!\delta^j}{j!}.$$

Now, let us define

$$f_{\delta,i} \triangleq \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!(\delta)^j}{j!}. \tag{4.4}$$

Then we obtain $\Omega_{\delta,i} = \frac{(1-\delta)^i}{i!} f_{\delta,i}$. In addition, according to (4.3), for all $i \geq 2$ it is desired to have $\Omega_{\delta,i} = (1-\delta)\Omega_i$. Thus, we can formulate part of the design criterion as $f_{\delta,i} = i!(1-\delta)^{-(i-1)}\Omega_i$, or equivalently,

$$\begin{aligned} \forall i \geq 2, \;\; \sum_{j=0}^{D-i} \frac{\Omega_{i+j}(i+j)!\delta^j}{j!} &= i!(1-\delta)^{-(i-1)}\Omega_i \\ &= i(i-1)\Omega_i \sum_{j=0}^{\infty} \frac{(i+j-2)!\delta^j}{j!}. \end{aligned} \tag{4.5}$$

In (4.5), the summation on the right-hand side is derived by evaluating the Taylor series expansion of the function $(i-2)!x^{-(i-1)}$ centred at $x_0 = 1$ for $x = 1-\delta$. Assuming the maximum degree $D$ could be infinite, equation (4.5) suggests the following solution:

$$\Omega(x) = \sum_{i=2}^{\infty} \frac{1}{i(i-1)} x^i. \tag{4.6}$$

Surprisingly, this solution is the well known ideal Soliton distribution [17]. This, however, is an infinite degree distribution which cannot be used in a practical setup.

With a finite allowed maximum degree $D$, (4.6) must be modified. In the next subsection, we provide a finite approximation of (4.6) that remains close to optimal throughout the decoding process by satisfying (4.3) with a good approximation.

### 4.3.5 Finite Maximum Degree Design

Recall that for the outer code to finish the decoding successfully, the LT code requires a recovery rate greater than $R+\sigma$. Also, for a finite maximum degree design, to satisfy (4.3) and motivated by (4.6), we seek an $\Omega(x)$ approximately in the form of

$$\Omega(x) = \sum_{i=2}^{D} \frac{c}{i(i-1)} x^i.$$

As was first mentioned in [26], the hypergraph collapse process studied in [25] is identical to the EDD process. Now, let $r$ be a positive real number less than or equal to the smallest positive root of $(1+\varepsilon)\Omega'(x) + \ln(1-x) = 0$. Then, as $k \to \infty$, under EDD, $rk$ intermediate bits are recoverable with a high probability from any set of $(1+\varepsilon)k$ received bits [25]. Similar results were also obtained in [4], based on the And-Or tree analysis [24]. This result was used in [40] to study the performance of EDD for recovery of a less-than-one portion of the message bits as needed in Raptor codes. The average recoverable portion of bits for a given degree distribution, therefore, is equal to the smallest positive root of $(1+\varepsilon)\Omega'(x)+\ln(1-x) = 0$.

Thus, to achieve a recovery rate of $R+\sigma$, we need $\Omega'(x) > \frac{-\ln(1-x)}{(1+\varepsilon)}$ for all $x \in (0, R+\sigma)$. By using the Taylor series expansion $-\ln(1-x) = \sum_{i=1}^{\infty} \frac{x^i}{i}$, a necessary condition for all $x \in (0, R+\sigma)$ can be derived as

$$\begin{aligned}
\Omega(x) = \int_0^x \Omega'(t)dt &\geq \int_0^x \frac{-\ln(1-t)}{(1+\varepsilon)dt} \\
&= \frac{x(1-\ln(1-x)) + \ln(1-x)}{(1+\varepsilon)} \\
&= \sum_{i=2}^{\infty} \frac{x^i}{(1+\varepsilon)i(i-1)}.
\end{aligned}$$

Among all the terms of the form $\omega_i(x) \triangleq \frac{x^i}{i(i-1)}$, the term $\omega_j(x)$ has the maximum derivative in the interval $I_j \triangleq (\frac{j-1}{j}, \frac{j}{j+1})$. Also, $\forall i > j \geq \ell \geq 2$, $\frac{d}{dx}\omega_j(x) > \frac{d}{dx}\omega_i(x)$ for any $x \in I_\ell$. Therefore, to have

$$\forall x \in (0, R+\sigma)\ \ \Omega(x) > \frac{1}{(1+\varepsilon)} \sum_{i=2}^{\infty} \omega_i(x), \tag{4.7}$$

for a given $\varepsilon$, it is enough to set

$$\Omega(x) = \sum_{i=2}^{m} \frac{c}{i(i-1)} x^i + \left(1 - \frac{c(m-1)}{m}\right) x^{m+1}, \tag{4.8}$$

where $m$ is an integer such that $m \geq m^*$ and $\frac{m^*-1}{m^*} \leq (R+\sigma) \leq \frac{m^*}{m^*+1}$, and $c \geq \frac{1}{1+\varepsilon}$. Clearly, choosing a larger $m$ results in a better approximation to (4.6).

Using (4.7) and (4.8), we obtain

$$\begin{aligned} &\forall x \in (0, (R+\sigma)), \\ &\quad c \geq \frac{x(1-\ln(1-x)) + \ln(1-x) - x^{m+1}(1+\varepsilon)}{(1+\varepsilon)(\sum_{i=2}^{m} \frac{x^i}{i(i-1)} - \frac{(m-1)}{m} x^{m+1})}. \end{aligned} \tag{4.9}$$

The right-hand side of (4.9) is a strictly increasing function of $x$. Thus, we can finish the design by choosing $c$ equal to the value of the right-hand side evaluated at $x = R + \sigma$.

In order for the decoding to start and recover a portion of intermediate bits before the first inactivation, we provide a very small positive $\Omega_1$, as do the existing approach.

Setting $m = m^*$ provides the lowest computational complexity since doing so obtains the smallest average degree of the distribution. However, setting $m = m^*$ also reduces the probability of covering a randomly selected intermediate bit in an output symbol and therefore slightly increases the reception overhead. This slight increase is a side-effect of a decreasing outer code rate in the 3GPP for a smaller block length. At $\ell = 1024$, the outer code rate is reduced to $R = 0.9381$, and setting $m = m^* = 16$ makes the average degree of our $\Omega(x)$ slightly smaller than that of the 3GPP. Moreover, the probability of leaving an intermediate bit uncovered (not involved in any of the equations corresponding to the received bits) is approximately $e^{-\Omega'(1)(1+\varepsilon)}$ [2], where $\Omega'(1)$ represents the average degree. Therefore, for

successful decoding, a slightly higher overhead will be needed. As reported in Table I, this increase is around 0.33% for $\ell = 1024$ and only 0.04% for $\ell = 8192$.

The choice of the outer code rate $R$ in the 3GPP is based on the performance of the adopted $\Omega(x)$. Appropriate outer code selection for our proposed $\Omega(x)$ can be considered. Among other solutions for this slight increase in the overhead, one can either allow $m$ to be larger than $m^*$ or add a term of higher order to prevent the loss of coverage. In Section 4.4, we will compare our design numerically with that of the 3GPP.

## 4.4 Numerical Results

In order to verify the performance of our proposed codes, we compare the performance of the degree distribution adopted in 3GPP with the proposed degree distribution introduced in (4.8), where $m$ is chosen to be equal to $m^*$. In each case, similar to [6, 38, 42] we assume that the receiver receives enough overhead to form a full rank equation system in terms of the intermediate variables. As ID is a version of ML decoding, having a full rank equation system is sufficient for successful decoding. Hence, the decoding success rate is always one.

Figures 4.3 and 4.4 depict the average performance of both degree distributions for different block lengths $\ell$, and the different strategies used for selecting a node for inactivation. As discussed in Section 4.3, the basis of our comparison is the number of required inactivations. Thus, figures 4.3 and 4.4 provide the cumulative distribution function (CDF) for a normalized number of inactivations (i.e., the number of inactivated nodes required for successful decoding divided by the block length). Figure 4.3 compares the performance of our proposed degree distribution with that of the 3GPP codes under two different selection strategies for a block length of $\ell = 1024$. figure 4.4 repeats the same comparison for $\ell = 8192$. In both figures, the rate of the outer code is chosen according to 3GPP guidelines. This rate for $\ell = 8192$ is equal to $R = 0.9834$ and for $\ell = 1024$ is $R = 0.9381$. As figures 4.3 and 4.4 reveal, for all cases, the performance of our proposed degree distribution is superior to the degree distribution of the 3GPP.

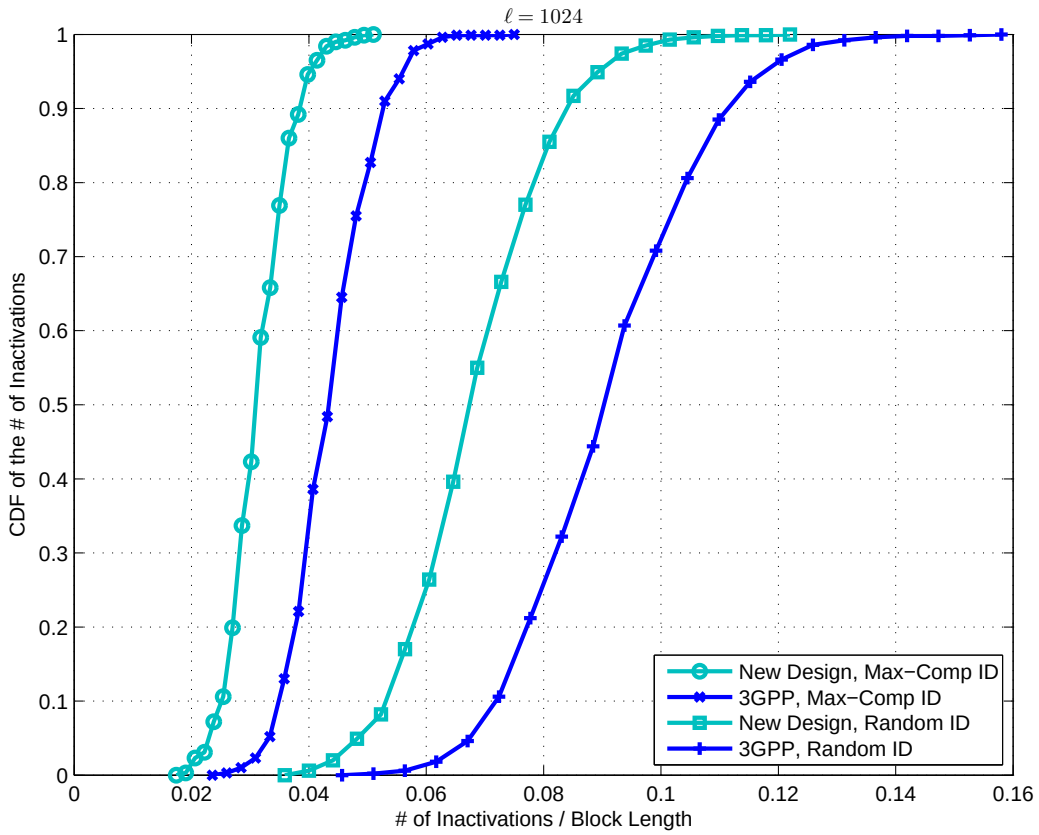



Figure 4.3: CDF of the normalized number of inactivations required for successful decoding, $\ell = 1024$. Simulations are performed based for $10^4$ iterations. In each iteration a complete block of information is transmitted and its overhead is calculated.

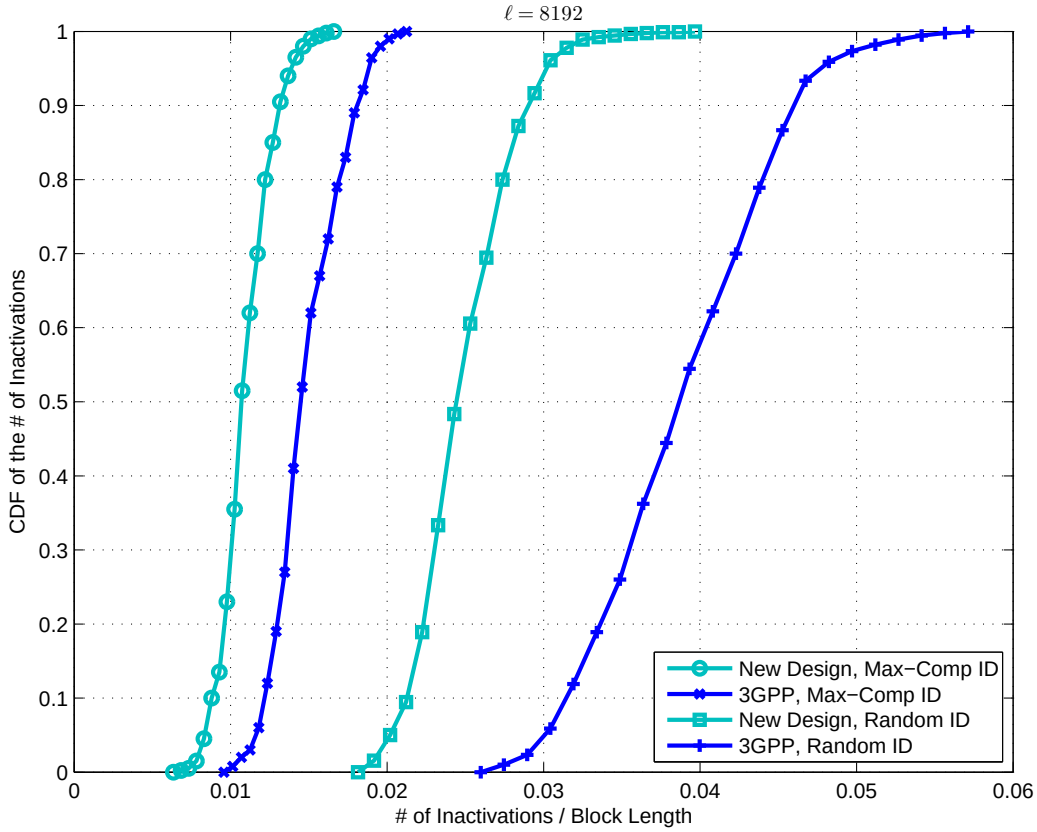



Figure 4.4: CDF of the normalized number of inactivations required for successful decoding, $\ell = 8192$. Simulations are performed based for $10^4$ iterations. In each iteration a complete block of information is transmitted and its overhead is calculated.

Table 4.4 shows the average performance measures for our simulations, which are again based on $m = m^*$. In this table, $\bar{d} = \Omega'(1)$ is the average degree of the distributions, $\bar{\varepsilon}$ represents the average reception overhead. Also, $\bar{I}_{Max-Comp}$, and $\bar{I}_{Rand}$ denote the average normalized number of inactivations when the selection of nodes for inactivation has been performed based on using the Max-Component ID and Random ID strategies, respectively. Table 4.4 indicates that our codes significantly reduced the number of inactivations at the cost of a slightly higher overhead.

## 4.5 Summary

A new criterion for the design of degree distributions for inactivation decoding was presented. Based on this criterion, a family of degree distributions was found analytically. The suggested family was modified for the practical case of finite maximum degree. The simulation results confirmed the superiority of the proposed codes over existing designs.

Table 4.1: Average performance measures for the 3GPP code and the proposed design with $m = m^*$.

| Code | $\ell$ | $\bar{d} = \Omega'(1)$ | $\bar{\varepsilon}$ | $\bar{I}_{Max-Comp}$ | $\bar{I}_{Rand}$ |
|---|---|---|---|---|---|
| 3GPP | 1024 | 4.6184 | 0.38% | 4.49% | 9.40% |
| | 8192 | 4.6184 | 0.44% | 1.54% | 4.03% |
| New Design | 1024 | 3.9739 | 0.66% | 3.20% | 7.04% |
| | 8192 | 5.0854 | 0.48% | 1.13% | 2.59% |

# Chapter 5

# Conclusions

This thesis focused mainly on rateless codes for erasure channel. The purpose of this research was to study the new techniques for improving the performance of Raptor codes in practical settings. The results presented in this work are used to combat the imperfections rising from the finite length of information block.

In Chapter 3 a new technique for rateless code design, named annotation, was introduced. Annotation provides the possibility of retrieving a portion of overhead in the original design of Raptor codes. Based on this technique, a generalized version of Raptor codes, called annotated Raptor codes, were introduced. We evaluated the performance of the new design for information block length of $k = 64520$, and compared the results with that of the original Raptor codes. The comparison shows that annotated Raptor codes are capable of achieving higher transmission rates in finite lengths compared to the original design of Raptor codes. The reduction in the overhead is shown to be more than $10\%$.

The design of Raptor codes for short block lengths are studied in Chapter 4. In current practical settings the information block is usually around several thousands. In this case a computationally more expensive decoding algorithm, named inactivation decoding, is preferred due to its better performance in terms of overhead. We proposed a new design criterion for Raptor code design under this setting. Based on this criterion an analytical framework for the Raptor code design is presented and the codes designed using this approach are compared with the conventional design. The results show notable improvements in the computational costs of this practical

method. The new design reduces the number of computationally expensive inactivation operations significantly. The amount of required overhead is shown to be decreased in different block lengths. However, for very short block length (e.g., $k = 1000$), a slight increase in the overhead is observed.

## 5.1 Future Research Directions

This thesis work provides a foundation for further research on several interesting topics. The annotation technique presented in Chapter 3 opens a new way of rateless code design. The main advantage of this method is to provide different levels of protection to different sets of information symbols in the coding. The goal in our design was to retrieve a portion of overhead and convert some of the useless received symbols into innovative receptions which transmit new information to the receiver. This potential however can be used in other directions such as non-equal error protection over different sets of information symbols. Setting an analytical framework for the analysis of performance in the new design will help the investigation of the advantages in this generalized design, and optimizing it for different applications.

Among other methods to provide a higher level of protection in annotated Raptor codes is to use the more powerful inactivation decoding algorithm for decoding the annotated part of the message in the receiver. Design of such codes can benefit the analysis provided in Chapter 4 of this thesis. Our initial experiments shows that such codes can reduce the reception overhead around $20\%$ compared to the original Raptor code. However, the optimal design needs more analysis and experiments.

The close connection between rateless codes and network coding promises a significant gain in using the rich analytical background of rateless codes for network coding. This connection has been first noticed and used in [46]. Although the original design of LT codes and Raptor codes do not allow their use in a distributed fashion required for network coding, some notable efforts has already been made to design decomposable versions of these codes [47]. Investigating the capabilities of new design approaches presented in this thesis for extensions to distributed versions

can open a new direction for research in future.

# Bibliography


[1] Cisco Visual Networking Index: Forecast and Methodology 2011-2016, 2012.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege,"A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *Proc. of the ACM SIGCOMM'98* Vancouver, BC, Canada, Sep. 1998, pp. 56–67.

[3] J. W. Byers, M. Luby, and M. Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast," in *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, 2002, pp. 1528–1540.

[4] A. Shokrollahi, "Raptor Codes," in *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[5] A. Shokrollahi, "Raptor Codes," in *IEEE Proc. of the International Symposium on Information Theory*, Chicago, USA, June - July 2004, pp. 36.

[6] A. Shokrollahi, and M. Luby, "Raptor Codes," in *ser. Foundations and Trends in Communications and Information Theory*, vol. 6, No. 3–4, pp. 213–322, 2011.

[7] K. Mahdaviani, M. Ardakani, and C. Tellambura, "Annotated Raptor Codes," in *Proc. of the IEEE Information Theory Workshop (ITW)*, Paraty, Brazil, Oct. 2011, pp. 272–276.

[8] A. Shokrollahi, S. Lassen, and R. Karp, "Systems and Processes for Decoding Chain Reaction Codes Through Inactivation, United States Patent Serial Number 6856263, Feb. 2005.

[9] K. Mahdaviani, M. Ardakani, and C. Tellambura, "On Raptor Code Design for Inactivation Decoding," accepted for publication in *IEEE Transactions on Communications*, Apr. 2012.

[10] P. Elias, "Coding for two noisy channels," in *Proc. of the 3rd London Symposium on Information Theory*, London, UK, Sep. 1955, pp. 61–76.

[11] T. M. Cover, and J. A. Thomas, "Elements of Information Theory," Wiley-Interscience, New York, NY, USA, ISBN:0-471-06259-6, 1991.

[12] D. J. C. MacKay, "Fountain codes," *IEE Proc. Communication*, vol. 152, no. 6, pp. 1062–1068, Dec. 2008.

[13] I. S. Reed, and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, vol. 8, no. 2, pp. 300–304, Jun. 2009.

[14] E. R. Berlekamp, “Nonbinary BCH decoding,” in *Proc. of the IEEE International Symposium on Information Theory*, San Remo, Italy, Sep. 1967.

[15] J. L. Massey, “Shift-register synthesis and BCH decoding,” *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[16] G. D. Jr. Forney, “On Decoding BCH Codes,” *IEEE Transactions on Information Theory*, vol. 11, no. 4, pp. 549–557, Oct. 1965.

[17] M. Luby, “LT Codes,” in *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[18] P. Maymounkov, “Online Codes,” Technical Report TR2002-833, New York University, 2002.

[19] C. Studholme, and I. Blake, “Windowed Erasure Codes,” in *Proc. of the IEEE International Symposium on Information Theory*, Seattle, Washington, USA, July 2006, pp. 509–513.

[20] N. Rahnavard, B.N. Vellambi, and F. Fekri, “Rateless codes with unequal error protection property,” in *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1521–1532, 2007.

[21] D. Sejdinovi, D. Vukobratovi,A. Doufexi, V. enk, and R. J. Piechocki, “Expanding Window Fountain Codes for Unequal Error Protection,” in *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2510–2516, Sept. 2009.

[22] M. C. O. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, “Sliding-Window Digital Fountain Codes for Streaming of Multimedia Contents,” in *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS’07)*, New Orleans, USA, May 2007, pp. 3467–3470.

[23] R. G. Gallager. “Low-Density Parity-Check Codes,” M.I.T. Press, Cambridge, MA, 1963.

[24] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of Random Processes via And-Or Tree Evaluation, in *Proc. of the ACM-SIAM Symposium on Discrete Algorithms SODA’98*, San Francisco, CA, USA, Jan. 1998, pp. 364–373.

[25] R. W. R. Darling, and J. R. Norris,“Structure of Large Random Hypergraphs, in *Annals Applied Probability*, vol. 15, no. 1A, pp. 125-152, Feb., 2005.

[26] E. Maneva, and A. Shokrollahi, “New Model for Rigorous Analysis of LT-Codes, in *Proc. of the IEEE International Symposium on Information Theory*, Seattle, USA, July 2006, pp. 2677-2679.

[27] M. A. Shokrollahi, “New sequences of linear time erasure codes approaching the channel capacity, in *Proc. of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-13)*, London, UK, 1999, pp. 65-76.

[28] P. Oswald, and A. Shokrollahi, “Capacity-Achieving Sequences for the Erasure Channel, in *IEEE Transactions on Information Theory*, vol. 48, no. 12, pp. 3017–3028, Dec. 2002.

[29] H. Saeedi, and A. H. Banihashemi, "New Sequences of Capacity Achieving LDPC Ensembles over the Binary Erasure Channel, in *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6332–6346, Dec. 2010.

[30] O. Etesami, and A. Shokrollahi, "Raptor Codes on Binary Memoryless Symmetric Channels," in *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.

[31] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Rateless Codes on Symmetric Channels," in *Proc. of IEEE International Symposium on Information Theory (ISIT '04)*, Chicago, IL, USA, Jun. 2004, p. 38.

[32] R. Palanki, and J. S. Yedidia, "Rateless Codes on Noisy Channels," in *Proc. of IEEE International Symposium on Information Theory (ISIT 04)*, Chicago, IL, USA, Jun. 2004, p. 37.

[33] J. Castura, and Y. Mao, "Rateless Coding over Fading Channels," in *IEEE Communications Letters*, vol. 10, no. 1, pp. 46-48, Jan. 2006.

[34] "Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Services (MBMS); Protocols and Codecs (Release 6)," 3GPP, Tech. Rep. 3GPP TS 26.346 V6.3.0, 2005.

[35] ETSI, DVB BlueBook A101 digital video broadcasting (DVB); IP datacast over DVB-H: content delivery protocols, Tech. Rep. TS 102 472 V1.2.1, Dec. 2006.

[36] Z. Chen, J. Castura, and Y. Mao, "On the Design of Raptor Codes for Binary-Input Gaussian Channels," in *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3269-3277, Nov. 2009.

[37] A. AbdulHussein, A. Oka, and L. Lampe, "Decoding with Early Termination for Raptor Codes," in *IEEE Communications Letters*, vol. 12, no. 6, pp. 444-446, June, 2008.

[38] S. Kim, S. Lee, and S.-Y. Chung, "An Efficient Algorithm for ML Decoding of Raptor Codes over the Binary Erasure Channel," in *IEEE Communications Letters*, vol. 12, no. 8, pp. 578-580, Aug. 2008.

[39] P. Pakzad, and A. Shokrollahi, "Design Principles for Raptor codes," in *Proc. of IEEE Information Theory Workshop (ITW '06)*, Punta del Este, Uruguay, Mar. 2006, pp. 165–169.

[40] S. Sanghavi, "Intermediate Performance of Rateless Codes," in *Proc. of IEEE Information Theory Workshop (ITW '07)*, Lake Tahoe, CA, USA, Sep. 2007, pp. 478–482.

[41] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efcient Erasure Correcting Codes," in *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[42] A. Shokrollahi, "Fountain Codes," in *the 2nd Tutorial of the IEEE International Symposium on Information Theory*, Seoul, Korea, June/July 2009.

[43] F. Chung, and L. Lu, “Connected Component in Random Graphs with Given Expected Degree Sequences,” in *Annals of Combinatorics*, vol. 6, pp. 125–145, 2002.

[44] P. Erdos, and A. Renyi, “On Random Graphs. I,” in *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[45] F. Chung, and L. Lu, “The Volume of the Giant Component of a Random Graph with Given Expected Degrees,” in *SIAM Journal of Discrete Mathematics*, vol. 20, no. 2, pp. 395–411, 2006.

[46] P. Maymounkov, and N. J. A. Harvey, “Methods for Efficient Network Coding,” in *Proc. of the 44th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA. Sep. 2006, pp. 40–49.

[47] S. Puducheri, J. Kliewer, and T. E. Fuja, “Distributed LT Codes,” in *Proc. of IEEE International Symposium on Information Theory (ISIT ’06)*, Seattle, USA, July 2006, pp. 987–991.