Unequal Error Protection Raptor Codes

Hussein Fadhel

Unequal Error Protection Raptor Codes

A Thesis
presented in partial fulfillment of requirements
for the degree of Master of Science
in the Department of Electrical Engineering
The University of Mississippi

by

Hussein Nadhem Fadhel

May 2012

ABSTRACT

We design Unequal Error Protection (UEP) Raptor codes with the UEP property provided by the precode part of Raptor codes which is usually a Low Density Parity Check (LDPC) code. Existing UEP Raptor codes apply the UEP property on the Luby transform (LT) code part of Raptor codes. This approach lowers the bit erasure rate (BER) of the more important bits (MIB) of the data decoded by the LT part of the decoder of Raptor code at the expense of degrading the BER performance of Less Important Bits (LIB), and hence the overall BER of the data passed from the LT part to the LDPC part of the decoder is higher compared to the case of using an Equal Error Protection (EEP) LT code.

The proposed UEP Raptor code design has the structure of UEP LDPC code and EEP LT code so that it has the advantage of passing data blocks with lower BER from the LT code part to the LDPC code part of the decoder. This advantage is translated into improved performance in terms of required overhead and achieved BER on both the MIB bits and LIB bits of the decoded data compared to UEP Raptor codes applying the UEP property on the LT part.

We propose two design schemes. The first combines a partially regular LDPC code which has UEP properties with an EEP LT code, and the second scheme uses two LDPC codes with different code rates in the precode part such that the MIB bits are encoded using the LDPC code with lower rate and the LT part is EEP. Simulations of both designs exhibit improved BER performance on both the MIB bits and LIB bits while consuming smaller overheads. The second design can be used to provide unequal protection for cases where the MIB bits comprise a fraction of more than 0.4 of the source data which is a case where UEP Raptor codes with UEP LT codes perform poorly.

This work is dedicated to my family,
for all their support and encouragement.

ACKNOWLEDGEMENTS

First and most I would like to thank Dr. Lei Cao for his guidance and much appreciated support throughout the program. His knowledge and intelligence always shed light on the path in my journey to learning. For his kindness, generous guidance and patient encouragement I am genuinely grateful.

I would like thank my committee members, Dr. Allen Glisson and Dr. Mustafa Matalgah for their time and patience.

For the opportunity to be part of such a prestigious program and all the support and help, I would like to express my heartfelt gratitude and appreciation to the Fulbright program.

Also I would like to thank my friends, colleagues and many other people for their support and encouragement.

Last but not the least I am grateful for the bliss of, and to my family for numerous reasons to count.

University, Mississippi Hussein Fadhel
May 2012

LIST OF ABBREVIATIONS

APP	a posteriori probability
BEC	binary erasure channel
BER	bit erasure rate
BI-AWGN	binary-input additive white Gaussian noise
BPA	belief propagation algorithm
BSC	binary symmetric channel
CN	Check Node
DE	Density Evolution
EEP	Equal Error Protection
LDPC	Low Density Parity Check
LIB	Less Important Bits
LR	likelihood ratio
LLR	log-likelihood ratio
MAP	maximum-a posteriori
MIB	more important bits
SPA	sum-product algorithm
SPC	single parity-check
UEP	Unequal Error Protection
VN	Variable Node

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

1. INTRODUCTION

Raptor codes [1] are a class of Fountain codes [2], and they are an extension of LT codes [3], which are the first practical realization of Fountain codes. LT codes have nonlinear encoding and decoding cost while Raptor codes overcome this issue by using a linear block code as a precode to encode source symbols before the LT code and use the output symbols of the precode as input symbols to the LT code. Fountain codes have the ability to adapt to changes in the channel characteristics that block codes may fail in adapting to because of the rateless nature of Fountain codes. Raptor codes combine the advantages of both types of codes, namely block codes and Fountain codes, to produce a class of Fountain codes with linear encoding and decoding times and the flexibility of Fountain codes to channel conditions. Raptor codes use as the precode a prominent class of linear block codes known as Low Density Parity Check (LDPC) codes [4] [5].

Today we have almost the entire world connected by a network of data communicating devices with networks of different sizes and complexities, and data transmission also occurs within the communicating devices themselves. Communicated data is generally received with errors inflicted by the channel, and various parts of the source information may have different sensitivities to errors, e.g., errors in the header of a packet has more effect than errors in the payload, so the need for Unequal Error Protection (UEP) channel codes is evident.

UEP in the terminology of channel codes refers to the process of providing certain parts of the source data called more important bits (MIB) with higher protection than the rest of the data. The effect of UEP can be observed in the bit erasure rate (BER) of decoded data where the BER of the MIB bits is considerably lower than the BER of the rest of the decoded bits.

MPEG streams where I-frames need higher protection than P-frames and packets transmitted through optical networks where errors in the header cause more damage than errors in the payload are two examples where unequal error protection can be applied as an efficient solution compared to increasing the overall error protection capability. Looking at these two examples we can also notice that different types of source information demand UEP codes with different designs.

Raptor codes are a concatenation of an LT code and a linear block precode such as LDPC codes. The UEP property can be inserted into Raptor codes by using a UEP LT code, UEP LDPC code, or both. UEP LT codes were first studied in [6] and [7]. Codes with different designs have been developed since then, and many approaches have been proposed for UEP LDPC codes such as [8], [9]and [10].

Raptor codes first encode source data of k bits with a precode such as LDPC codes. The n output bits of this process are called the intermediate symbols and are used as the input to the LT code part of Raptor codes. The decoder of Raptor codes consists of two parts, namely the LDPC part and the LT part. The received data, say of m bits are first decoded by the LT code part to retrive the n intermediate bits and then the length-n block is passed to the LDPC code part. The performance of the LDPC code partly depends on the performance of the LT part and the BER of the data passed to the LDPC decoder from the LT decoder, particularly considering that LDPC codes adopt the belief propagation algorithm (BPA) for the decoding where as the errors in the input data decreases less erroneous information propagates and more bits will be decoded correctly.

LT codes in general have higher error floors compared to LDPC codes; in fact LDPC codes outperform even the state of the art Turbo codes as was shown in [11]. Also a key issue to consider is that UEP LT codes have a higher overall BER for some overhead range compared to EEP LT codes with the same output degree distribution and code length over the same overhead range. Combining the arguments stated above we can notice the disadvantage of using UEP LT codes to design UEP Raptor codes compared to using EEP LT code in the sense that UEP LT codes will pass a higher overall BER to the LDPC part of the decoder of Raptor codes which affects its performance negatively. Existing UEP Raptor codes are constructed with EEP LDPC codes as precode and an UEP LT code e.g., [7], and [12]. The work of [13] is the only exception we came across where UEP codes are used on both the LDPC and LT parts.

1.1 THESIS CONTRIBUTION

We study the effect of applying the UEP on the precode part of UEP Raptor codes. From the discussion above we can infer that using Equal Error Protection (EEP) LT code and applying the UEP in the precode or LDPC code part in the design of UEP raptor codes gives the advantage of passing less errors in the data decoded by the LT code and which will be decoded by the LDPC code next. LDPC codes use belief propagation algorithm for decoding where as more correct information propagate through the decoding graph more bits will be decoded correctly in the different parts of the codeword and also LDPC codes have much lower error floors than LT codes. All these factors lead us to suspect that UEP Raptor codes with UEP precode will have a better performance than UEP Raptor codes with UEP LT code. In order to put this to the test we propose two designs: The first has the structure of a partially regular LDPC code as precode, which provides UEP by optimizing the weight distribution of the columns of the parity check matrix, and EEP LT code. The second design has the structure of two EEP LDPC codes of different rate in the precode part and EEP LT code. The second design has the flexibility to provide UEP for cases where the MIB part comprises a fraction $\alpha \geq 0.4$ of the source data while UEP Raptor codes with

UEP LT code exhibit poor performance in such cases. We run simulations for both designs and the results show that both code designs achieve improved performances in terms of the BER of decoded data on both MIB and LIB parts while consuming less overhead compared to UEP Raptor codes with UEP LT codes.

1.2 ORGANIZATION OF THESIS

In Chapter 2 we introduce Fountain codes and discuss LT codes as a practical realization in terms of the mechanism in which Fountain codes work, encoding, decoding and the significance of output degree distribution in the success or failure of LT codes. In Chapter 3 LDPC codes are introduced, the matrix and graphical representations are reviewed, and the encoding and decoding processes are discussed with some detail. In Chapter 4 we introduce Raptor codes and as we go through the chapter the importance of Chapters 2 and 3 will become evident, as Raptor codes are a concatenation of LT and LDPC codes. We discuss the design of Raptor codes, the advantages they have over LT codes, the encoding and decoding processes, and the degree distribution developed for Raptor codes. In Chapter 5 we introduce unequal error protection codes and how the UEP property is embedded into LT, LDPC, and eventually Raptor codes. We discuss the design and implementation of the UEP Raptor codes we propose, i.e., applying the UEP property in the precode part of Raptor codes. Following that simulations results are given. Conclusions and points we believe to be worthy for future study are the subject of Chapter 6.

Chapter 2

2. FOUNTAIN CODES AND LT CODES

Fountain codes [2] are a class of rateless codes. For a given block of data of k input symbols the code can generate a potentially limitless number of encoding symbols n. These input and source symbols can be bits or blocks of data of equal size. Fountain codes have simple encoding and decoding algorithms based on the sparse nature of their code graphs and the decoder can recover the original data after receiving any set of encoding symbols with cardinality slightly higher than the size of the source block k.

The nomenclature comes as Fountain codes can be visualized as a fountain with endless water drops (encoded symbols) and a receiving end can continue collecting these drops until the original file is recovered [14] . Fountain codes are universally near optimal, i.e., they approach capacity for any erasure channel.

Traditional block codes can be inefficient in terms of their use of the channel resources. If the communicating devices are unable to identify the channel error probability, then the code rate used can prove inefficient either for using unnecessary redundancy or using a higher rate than what is required to achieve reliable transmission. This can be more evident with large scale communication such as multicast or satellite communications, where monitoring the condition of every channel to assign the proper rate is unrealistic and therefore the worst case error rate is adopted, which may be wasteful on some channels and still may fail on some other channels due to sudden changes in the channel characteristics. Fountain codes can overcome such issues with their rateless nature.

Fountain codes also have the potential to replace the transmission control protocol TCP used on the Internet in different cases by taking advantage of the fact that the Internet can be well modeled by the binary erasure channel BEC [15]. TCP uses Automatic Repeat Request (ARQ) for error control and this approach can be highly inefficient and wasteful of channel resources in some cases, e.g., in multicast situations where different receivers may demand retransmission of different packets. Fountain codes provide an alternative utilizing their fountain-like nature that allows any receiving node to reconstruct the source data from any set of encoding symbols.

LT codes proposed by Michael Luby in 1998 are the first practical realization of Fountain codes. Below we introduce LT codes in order to see the details of encoding and decoding of LT codes and Fountain codes in general.

LT codes [3] can be used to encode source data of size k symbols, where each symbol can be one bit or an arbitrary $l-$bit long symbol. "Each encoding symbol can be generated, independently of all other encoding symbols, on average by $O(ln(k/\delta))$ symbol operations, and the k original input symbols can be recovered from any $k + O(\sqrt{k} \cdot \ln^2(k/\delta))$ encoding symbols with probability $1 - \delta$ by on average $O(k \cdot ln(k/\delta))$ symbol operations" [3], where δ is the failure probability of the decoder to recover the original data from any K $> k$ encoding symbols, and a symbol operation is either an exclusive-or operation between two symbols or copying one symbol to another.

2.1 LT ENCODING

The encoding process is fairly easy to describe:

1. Randomly choose a degree d for the encoding symbol from a degree distribution. The degree of an encoding symbol is the number of input symbols participating to produce the encoding symbol.

2. Uniformly at random choose d distinct input symbols.

3. The value of the encoding symbol is the exclusive-or of the d input symbols for (d > 1) and it is simply copying the input symbol to the output symbol for ($d = 1$).

Source data of different sizes require different distributions and the degree distribution sampled to choose the degree of encoding symbols has a major role in the failure or success of the decoding process and its design and effects will be presented in section 2.4.

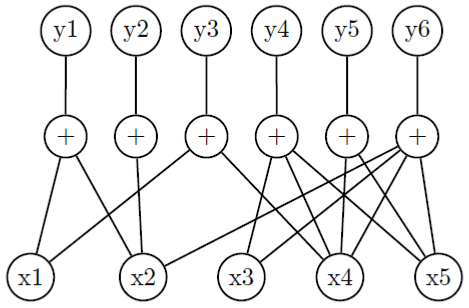

Figure 2.1. LT encoding

Figure 2.1 is a bipartite graph illustrating a toy example of encoding 5 input symbols x1, . . . , x5 to generate 6 encoding symbols y1, . . ,y6. For example, output symbol y1 = x1⊕x2 is a degree-2 encoding symbol connected to two nodes called neighbors. y2 is degree-1 and has only one neighbor y2 = 0⊕x2 = x2.

After transmission the decoder receives a random set of encoded symbols with distortion based on the channel characteristics. In order to start decoding the decoder needs to know the degree and set of neighbors of each encoding symbol. Passing this information to the decoder can be accomplished as follows:

- the degree and set of neighbors are attached as a header to the packet containing the encoding symbol.

- the encoder computes the required information for each encoding symbol implicitly based on the timing of reception, the position of the encoding symbol relative to the other symbols, or through passing a key which can be used as a seed to a pseudo-random generator in order to reconstruct the degree and set of neighbors of each encoding symbol. The software used in our simulations uses the latter method.

When decoding starts, the decoder knows the degree and set of input symbols used to construct each encoding symbol. For a decoder that uses belief propagation MAP algorithm [16], such as the software used in our simulations, this information is passed to the MAP algorithm as extrinsic information and then the decoding objective becomes finding the correct value of each input symbol.

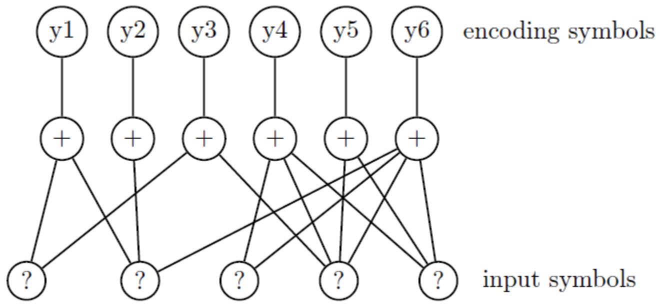

Figure 2.2. LT decoding

2.2 LT DECODING

To understand the mechanism in which LT decoding operates we need to keep in mind that each encoding symbol of degree d $>$ 1 is the XOR-sum (addition on $\mathbb{F}_2$), of its input symbol neighbors, and generating a degree one encoding symbol is simply the process of copying a randomly chosen input symbol to the encoding symbol.

When decoding starts, a degree one encoding symbol is released to cover the input symbol it contains. Now this input symbol can be subtracted from all the encoding symbols that it is a neighbor to and their degree can be decremented by one since the decoder identified the value of this input symbol. The subtraction is done by XORing between the corresponding symbols, subtraction is equivalent to addition on $\mathbb{F}_2$. From the perspective of a bipartite graph of LT codes, decoding is the process of removing edges connecting encoding symbols that have been recovered untill an encoding symbol has one edge left, i.e., it is degree-1, in which case it contains the value of the input symbol it is connected to and can be released to the ripple (the set of encoding symbols reduced to degree one). In Figure 2.2, y2 is a degree one encoding symbol which can be released to cover x2, which in turn can reduce y2 to degree-1. This process continues for every encoding symbol of degree $d > 1$ until it decreases to degree 1 in which case it can be released to cover the input symbol it contains if it has not been covered yet. Having that clarified we can present two definitions from Luby's paper LT Codes [3]

Definition 1 (decoder recovery rule): If there is at least one encoding symbol that has exactly one neighbor then the neighbor can be recovered immediately since it is a copy of the encoding symbol. The value of the recovered input symbol is exclusive-ORed into any remaining encoding symbols that also have that input symbol as a neighbor, the recovered input symbol is removed as a neighbor from each of these encoding symbols and the degree of each such encoding symbol is decreased by one to reflect this removal.

Definition 2 (LT process): All input symbols are initially uncovered. At the first step all encoding symbols with one neighbor are released to cover their unique neighbor. The set of covered input symbols that have not yet been processed is called the ripple, and thus at this point all covered input symbols are in the ripple. At each subsequent step one input symbol in the ripple is processed: it is removed as a neighbor from all encoding symbols which have it as a neighbor and all such encoding symbols that subsequently have exactly one remaining neighbor are released to cover their remaining neighbor. Some of these neighbors may have been previously uncovered, causing the ripple to grow, while others of these neighbors may have already been in the ripple, causing no growth in the ripple. The process ends when the ripple is empty at the end of some step. The process fails if there is at least one uncovered input symbol at the end. The process succeeds if all input symbols are covered by the end.

Definition 2 introduces the term *ripple* which is the set of degree one encoding symbols, or covered input symbols which have not been processed yet, i.e., have not been subtracted from neighboring encoding symbols yet. From the definition we can understand the significance of the ripple in the success of the decoding process. If the ripple is empty before recovering all input symbols the decoding process fails. At the same time it is advantageous to keep the size of the ripple small in order to avoid having redundant input symbols covered in the ripple. These requirements can be satisfied by proper analysis and careful design of the degree distribution.

2.3 DESIGN AND ANALYSIS OF THE OUTPUT DEGREE DISTRIBUTION

Definition 3 (degree distribution): For all d, $\rho(d)$ is the probability that an encoding symbol has degree d [3].

The design of the degree distribution has the following two objectives:

1. Ensure the success of the LT process using as few encoding symbols as possible.

2. Keep the average degree of encoding symbols as small as possible.

Using the degree distribution $\rho(1) = 1$ which corresponds to choosing degree one for all encoding symbols, and encoding is done by simply choosing an input symbol at random and copying its value to the encoding symbol. The probability analysis of this case shows that $k \cdot \ln \frac{k}{\delta}$ encoding symbols are needed to to cover all input symbols at the decoder side with probability $1 - \delta$. From this we can infer that for any distribution the sum of the degrees needed to recover k input symbols must be at least $k \cdot \ln \frac{k}{\delta}$.

Before going into the details of the degree distribution $\rho(i)$ we first analyze the probability of releasing an encoding symbol of degree i when L input symbols remain unprocessed. Below we state the definition of the degree release probability as given in [3].

Definition 4 (degree release probability): Let $q(i, L)$ be the probability that an encoding symbol of degree i is released when L input symbols remain unprocessed, then $q(i, L)$ is :

- $q(1, k) = 1$ for $i = 1$

- For $i = 2, ...k$, for all $L = k - i + 1, ..., 1,$

$$\frac{i(i-1) \cdot L \cdot \prod_{j=0}^{i-3} k-(L+1)-j}{\prod_{j=0}^{i-1} k-j} \tag{2.3.1}$$

- For all other i and L, $q(i, L) = 0$.

Definition 5 (overall release probability): Let $r(i, L)$ be the probability that an encoding symbol is chosen to be of degree i and is released when L input symbols remain unprocessed, i.e., $r(i, L) = \rho(i)q(i, L)$. Let $r(L)$ be the overall probability that an encoding symbol is released when L input symbols remain unprocessed, i.e., $r(L) = \sum_i r(i, L)$.

Where $\rho(i)$ is the output degree distribution. The significance of the release probability will become evident later when we discuss the importance of the size of the ripple. Also we will be able to value the importance of the degree distribution $\rho(i)$ in the LT process in terms of keeping the ripple at a desirable size.

2.4 IDEAL SOLITON DISTRIBUTION

The name Soliton distribution comes from the analogy to the Soliton wave, a wave that travels at a constant speed while maintaining its amplitude due to its unique property of perfect balance between dispersion and refraction. A similar property is required in the output degree distribution in the sense of keeping the ripple at a desirable size by ensuring that input symbols are added to the ripple at the same rate they are processed.

When designing the degree distribution there are two properties that impose the constraints in light of which the degree distribution is chosen :

- keep the ripple size small enough in order to avoid covering input symbols already in the ripple to avoid unnecessary redundancy.
- keep the ripple size large enough in order to ensure the ripple does not disappear before recovering all input symbols.

Analytically, the ideal Soliton distribution performs ideally in terms of the total number of encoding symbols needed to recover the source data and desirable ripple size, however this distribution performs very poorly in practice for reasons we will be able to understand after some analyses.

Definition 6 (Ideal Soliton distribution): The Ideal Soliton distribution can be expressed as:

- $\rho(1) = 1/k$
- For all $i = 2, ..., k, \rho(i) = \frac{1}{i \cdot (i-1)}$.

Recall that the ripple is of the set of released encoding symbols which have not been processed yet, hence it is directly affected by the release probability of the encoding process, which depends on the degree distribution $\rho(i)$. The release probability is equal to $r(i, L) = \rho(i) \cdot q(i, L)$

Using the argument above we can use the analyses of the release probability as a tool to determine the expected size of the ripple using the Ideal Soliton distribution.

uniform release probability $r(L)$:

For the Ideal Soliton distribution, $r(L) = 1/k$ for all $L = k, \ldots, 1$.

Using the value of the release probability given above we can calculate the expected number of encoding symbols released, i.e., the ripple size, of the LT process using the Ideal Soliton distribution.

$$k \cdot r(L) = k \cdot \sum_{i=1}^{k-L+1} r(i, L) = 1 \tag{2.4.1}$$

Assuming an ideal case, one encoding symbol is released for every input symbol processed and exactly k encoding symbols are sufficient to recover k input symbols. However the ideal case is far from the actual case and the Ideal Soliton distribution performs very poorly in practice because the expected ripple size is one, and any variance can cause the ripple to decrease to zero and cause the decoding process to fail. However the ideal Soliton distribution gives an insight into the type of behavior required from the degree distribution and in fact the Ideal Soliton distribution is modified to produce the Robust Soliton distribution.

2.5 ROBUST SOLITON DISTRIBUTION

The modified distribution is designed based on two objectives: the expected ripple size remains large enough throughout the decoding process and ripple size is not larger than necessary in order to avoid redundancy. The Robust Soliton distribution introduces two new parameters, δ, and c where δ is the allowable failure probability of the decoder to recover the source data for any K encoding symbols and c is a constant $c > 0$.

The Robust Soliton distribution modifies the release probability so that the expected size of the ripple is about $\ln(\frac{k}{\delta}\sqrt{k})$ by analogy to the case of a random walk of length k, where the probability of deviating from the mean by more than $\ln(\frac{k}{\delta}\sqrt{k})$ has an upper bound of δ.

Definition 7 (Robust Soliton distribution): The Robust Soliton distribution $\mu(i)$ is defined as follows. Let $S = c \cdot \ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$. Define

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

To obtain the robust Soliton distribution $\mu(i)$, add $\tau(i)$ to the Ideal Soliton distribution $\rho(i)$ and normalize as follows:

- $\beta = \sum_{i=1}^{k} \rho(i) + \tau(i)$
- for all $i = 1, ..., k, \mu(i) = (\rho(i) + \tau(i))/\beta$.

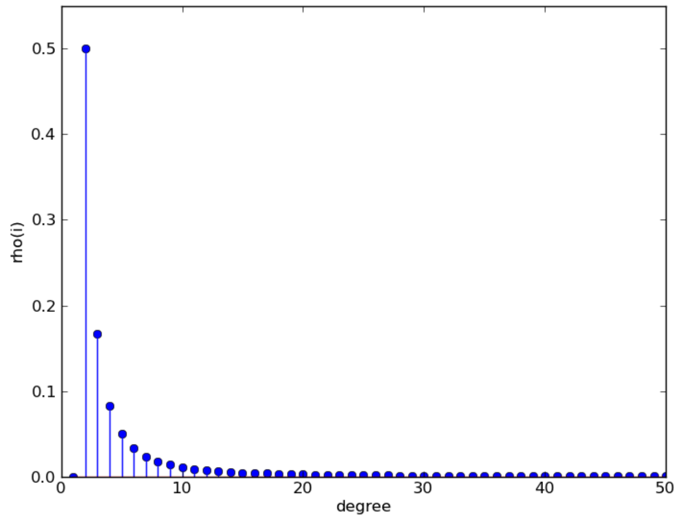

Figure 2.3. Ideal Soliton distribution for $k = 10000$

Figure 2.3 illustrates the probability mass function of the Ideal Soliton distribution for $k = 10000$. The additional parameter $\tau(i)$ is shown in Figure 2.4. The small elevation in the beginning ensures that there are enough degree-1 encoding symbols for decoding to start and the spike at k/S increases the probability that every source symbol is covered. Figure 2.5 illustrates the Robust Soliton distribution for $k = 10000, c = 0.3$, and $\delta = 0.05$.

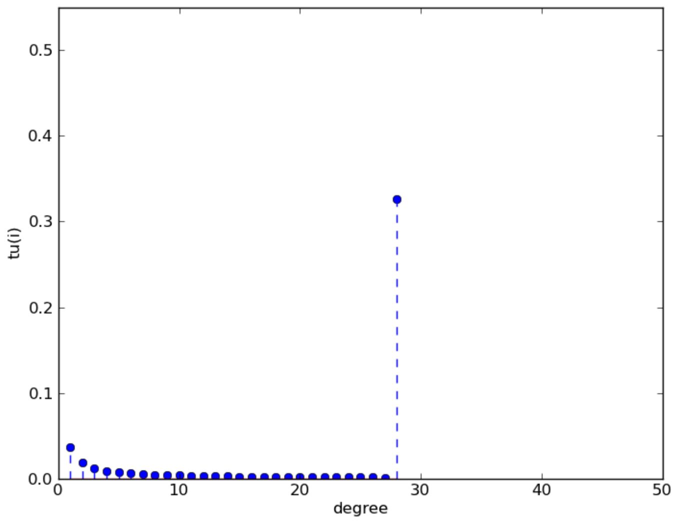

Figure 2.4. $\tau(i)$ for $c = 0.3$ and $\delta = 0.05$

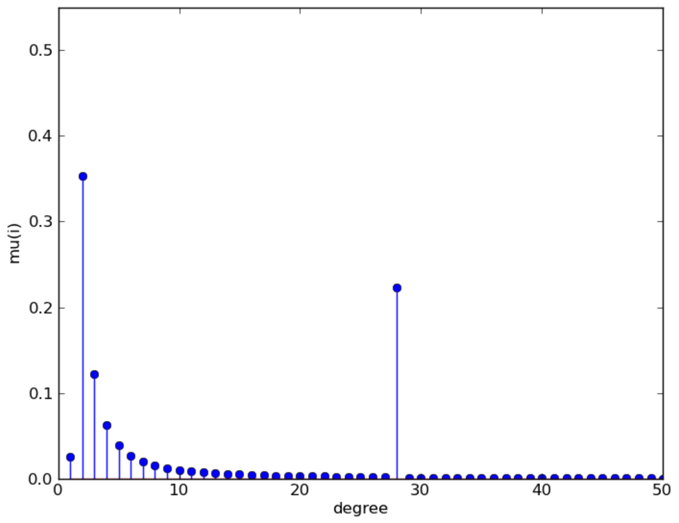

Figure 2.5. Robust Soliton Distribution

Chapter 3

3. LOW-DENSITY PARITY-CHECK CODES

Low-density parity-check (LDPC) codes are a class of linear block codes with near-capacity performance on a large set of data transmission channels, surpassing Turbo codes on many channels [17]. LDPC codes were discovered by Gallager and presented in detail in his doctoral dissertation in 1960 [4] but remained ignored until 1981 when Tanner used bipartite graphs to generalize and introduce a graphical representation of LDPC codes, and this representation was eventually named Tanner graph [18]. LDPC codes were redicovered in the 1990s by MacKay and Neal [19], [5]. The LDPC codes suggested by Mackay and Neal are slightly different than Gallager's, Mackay and Neal reported based on empirical results that Gallager's work is superior [20].

3.1 REPRESENTATIONS OF LDPC CODES

Introducing the representations of LDPC codes can help understand both the mathematical basis and encoding and decoding algorithms used to construct LDPC. Without loss of generality we will consider only binary LDPC codes with arithmetic on $\mathbb{F}_2$.

3.1.1 MATRIX REPRESENTATION

We start by a brief definition of linear codes.

Linear Codes: A linear error-correcting code can be represented by an $K \times N$ binary matrix

G (the generator matrix), such that a K-bit binary message s can be encoded as a N-bit vector $t = sG \, mod \, 2$.

LDPC codes are usually represented by their parity-check matrices H of size $M \times K$ where $M = N - K$ and H is the null space of the $K \times N$ Generator matrix G. A quick look at the dimensions of the two matrices can reveal that their row dimensions sum up to N which is the dimension of the vector space to which the vector subspaces H and G belong. LDPC codes are linear block codes with sparse or low-density parity-check matrices and this property is an important attribute when it comes to finding good implementable decoders. Before going forward we give few definitions first :

Definition 1: The weight of a binary vector or matrix is equal to the number of 1s in it. The overlap between two vectors is the number of 1s in common between them. The density of a source of random bits is the expected fraction of 1's. A vector is considered to be sparse if its density is less than 0.5. A vector is very sparse if its density vanishes as its length increases, for example, if a constant number c of its bits are 1s. Low-density can be a vague term but usually a density of or less than 0.01 would be considered low.

The code rate of the LDPC code is usually calculated as:

$$R = 1 - \frac{M}{N} = \frac{K}{N}$$

A source message s of size $1 \times K$ is encoded as a $1 \times N$ codeword $t = sG \, mod \, 2$ where $G_{(K \times N)}$ is the Generator matrix of the LDPC code. When the codeword t is transmitted, channel noise n is added and the recieved codeword will be:

$$r = (sG + n) \, mod \, 2$$

The task of the decoder is to deduce s given the received codeword r, the assumed noise properties of the channel, and the code structure which is assumed known to both the encoder and decoder. An optimal decoder is supposed to return the message s which maximizes the posteriror probability:

$$\frac{p(r|s,G)p(s|G)}{p(r|G)} = \frac{p(r|s,G)p(s)}{p(r|G)}$$

3.1.2 GRAPHICAL REPRESENTATION

LDPC codes can be graphically represented by Tanner graphs [18]. Tanner graphs are bipartite graphs.
A bipartite graph denoted by $G = (U, V, E)$ is a graph whose vertices (nodes) are decomposed into two disjoints sets U and V such that every edge e in the set of edges E connects a vertex in U and a vertex in V and no two nodes in the same set are connected.

The two types of vertices or nodes in a Tanner graph are the variable nodes (VN) which correspond to the coded bits and the check nodes (CN) which correspond to the constraint bits. Before going further into the details of Tanner graphs we present some information about the construction of the Parity-check matrix H which a Tanner graph provides a graphical representation of.

The parity-check matrix H is a $(N - K) \times N$ low density matrix on which the actual decoding is performed. When designing LDPC codes usually the parity-check matrix is constructed first with the desirable characteristics in the code and then the generator matrix G is generated based on the fact that the H matrix is a vector subspace on F2 and G is its dual subspace

$$GH^T = 0 \, mod \, 2$$

There are different methods to generate parity-check matrices and below we present a number of easy to follow methods as given in [20]:

1. Matrix H is generated by starting from an all-zero matrix and randomly flipping g not necessarily distinct bits in each column.

2. Matrix H is generated by randomly creating weight g columns.

3. Matrix H is generated with weight g per column and (as near as possible) uniform weight per row.

4. Matrix H is generated with weight g per column and uniform weight per row, and no two columns having overlap greater than 1.

5. Matrix H is further constrained so that its bipartite graph has large girth.

6. martix H = [H1|H2] is further constrained or slightly modified so that H2 is an invertible matrix.

More on the new terms and concepts introduced above later.

Back to our main discussion, a Tanner graph is constructed as follows: an edge connects a CN i to a VN j if the element h$_ij$ in the parity-check matrix H is equal to 1. Suppose that we have the parity-check matrix below with $R = \frac{K}{N} = \frac{1}{2}$, column weight $w_c = 2$ and row weight $w_r = 4$:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Then the Tanner graph corresponding to this H matrix would be as shown in Figure 3.1 :

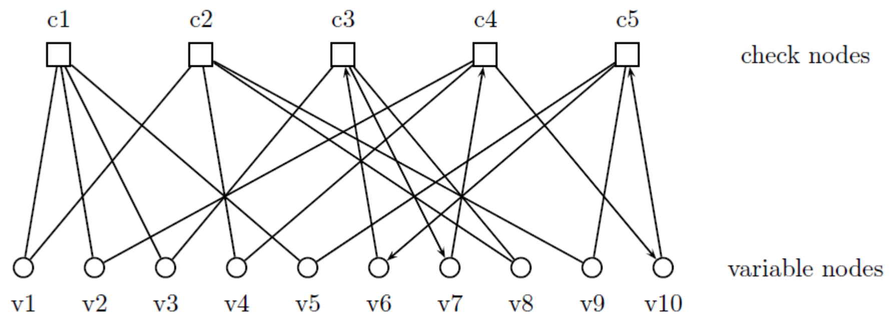

Figure 3.1. Tanner graph of the H matrix

In Figure 3.1 if we look at variable node 1 (v1) we can see that it is connected to check nodes 1 and 2, now if we examine the H matrix we can observe that elements h$_{11}$ and h$_{21}$

are equal to 1. Also if we check node 1 (c1), we can see it has four edges connected to v1, v2, v3 and v4, now if we go back to the H matrix we can observe that $\mathrm{h}_{11} = \mathrm{h}_{12} = \mathrm{h}_{13} = \mathrm{h}_{14} = 1$.

Tanner graphs of LDPC codes can help in visualizing and understanding the iterative decoding process of LDPC codes as follows:
Each of the nodes acts as a locally operating processor and each edge acts as a channel to communicate information from a given node to each of its neighbors (neighbors of a node are the nodes which share an edge with it). The exchanged information is generally of probabilistic nature describing the amount of certainty about the values of the bits assigned to variable nodes, these informations is generally represented as a likelihood ratio (LR) or the numerically more stable log-likelihood ratio (LLR). The decoding is initiated by N LLRs computed from the $(1 \times N)$ received codeword and passed to the variable nodes. The decoder then works in an iterative fashion, where at each iteration Check-Nodes receive LLRs passed from neighboring Variable-Nodes,process the information and then pass appropriate information back to each of the neighboring Variable-Nodes which utilize the new information from their different neighboring check-nodes to reach a decision about their corresponding bit values. The iterations continue until a correct codeword is found or predetermined maximum number of iterations is reached.

Now we turn our attention to another topological aspect of LDPC codes that affect the performance of iterative decoders of LDPC codes. Cycles are a structural characteristic of LDPC parity-check matrices. A cycle in Tanner graph is a sequence of edges which form a closed path that begins and ends at the same node. Cycles can degrade the performance of the iterative decoder. As shown in Figure 3.1 we can see a cycle starting and ending at v10 by following the arrows. The length of a cycle equals the number of edges forming the cycle and the minimum cycle length in a graph is called the girth of the graph. A desired property in Tanner graph of LDPC cpdes is that the girth approaches infinity. The girth of the graph in Figure 3.1 is equal to 6. The shortest cycle has length four and manifests itself in a parity-check matrix as four 1's in the four corners of a rectangular sub-matrix. Notice that steps three and four in our example method given in the begining to create an H matrix are an effort to eliminate or reduce the number of cycles in the parity-check matrix. Applying cycle-eliminating algorithms to H matrices is a common practice and the software used in our simulations uses a procedure to eliminate cycles of length four in the H matrix. Cycles force the decoder to operate locally in some portions of the graph (around the graph) so that a globally optimum solution is impossible [21].

LDPC codes can be regular, irregular or partially regular. Regular LDPC codes have parity-check matrix with constant row and column weights. The Tanner graph shown in Figure 3.1 is regular, each VN has degree 2, i.e., it is connected to two neighbors via two edges, and each CN is degree 4. Irregular LDPC codes have varying row and column weights and their row and column or CN and VN degree distributions can be expressed by degree distribution polynomials $\rho(X)$ and $\lambda(X)$ respectively.

$$\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}$$

ρ_d is equal to the fraction of edges connected to degree-d CNs and d_c is the maximum CN degree in the graph.

$$\lambda(X) = \sum_{d=1}^{d_v} \lambda_d X^{d-1}$$

λ_d is equal to the fraction of edges connected to degree-d variable-nodes and d_v is the maximum VN degree in the graph.

The regular Tanner graph in Figure 3.1 can be represented by the polynomials: $\rho(X) = X^3$ and $\lambda(X) = X$

The number of edges in a graph can be calculated as :

$$E = \frac{N}{\int_0^1 \lambda(X)dX} = \frac{M}{\int_0^1 \rho(X)dX}$$

Finally, partially regular LDPC codes have column and row weights that can be divided into sets of different weights.

Irregular LDPC codes can outperform regular LDPC codes and exhibit performance extremely close to the Shanon limit [11]. It was also shown in [11] that the performance of irregular LDPC codes further approaches channel capacity as the block length increases. The same paper also shows that irregular LDPC codes can outperform the state-of-the-art Turbo codes for different code lengths.

As shannon suggested, finding good codes which can achieve or approach the channel capacity is a problem of finding codes with random nature and large block lengths. However finding optimal decoders is also a requirement and all these requirements have complexity as an inevitable byproduct.

Given the channel characteristics and a desired error probability ϵ, degree distributions $\lambda(X)$ and $\rho(X)$ can be optimized through a process known as Density Evolution (DE) [11] such that as long as the channel parameter is less than δ and minimum number of decoding iterations $l(\delta, \epsilon)$ are performed the error probability would stay below ϵ. Density Evolution can be used to reach specific optimal $\lambda(X)$ and $\rho(X)$ distributions which produce LDPC codes that can provide certain properties such as Unequal Error Protection UEP [22].

3.2 ENCODING

The encoding process of LDPC codes is generally straightforward. Given a soure message s $(1 \times K)s$ and a $(K \times N)$ Generator matrix G, then the encoded codeword

$$t = sG \; mod \, 2$$

LDPC codes can be systematic or non-systematic. In systematic codes the source message s is a part of the encoded word and the other part is the added redundancy. In systematic LDPC codes the Generator matrix has the form:

$$G = [I_K | P_{K \times M}]$$

The above form can be reached by transforming the parity-check matrix H into the form $H = [h_{M \times K} | I_M]$ using Gaussian elimination and matrix manipulation, where $M = N - K$. Knowing the property that $GH^T = 0$ we can follow :

$$\begin{bmatrix} I_K & P \end{bmatrix} \begin{bmatrix} h^T \\ I_M^T \end{bmatrix} = 0$$

$$I_K h^T + P I_M = 0 \, mod \, 2$$

$$P I_M = -I_K h^T \; mod \, 2$$

$$P = h^T$$

The last step comes as the additive inversre of an element a on $\mathbb{F}_2$ is itself. For the procedure above, matrix rank M is assumed for H.

As the Generator matrix $G = [I_K \; P]$ then it is trivial that the first K symbols or bits of the codeword will be the $(1 \times K)$ source message s. The added $N - K$ redundancy is sP where P is the $k \times M$ parity-check matrix. In non-systematic LDPC codes G is simply the dual space of the H matrix and after decoding the original source message $s$of size $(1 \times K)$ needs to be extracted from the decoded $(1 \times N)$ block.

The complexity of the encoding process depends on the density of (i.e., the number of 1's) in G matrix and the complexity grows in $O(N^2)$ where N is the block length of the encoded message. LDPC codes further approach the capacity of the channel as the block length increases. Considering the desire for larger block lengths and the dense nature of the G matrix the encoding complexity can become an issue when designing LDPC codes, therefore there have been many attempts to design LDPC codes with lower encoding and decoding complexity [23] [24] [25].

3.3 DECODING OF LDPC CODES

The decoding problem as stated in [19] is: given the received message $r = (t + n)\,mod\,2$ where $t = sG$ is the transmitted message and n is a sparse random vector with density equal to the channel error rate and given that H is the dual space of G the decoding problem becomes finding the most probable n that satisfies :

$$nH^T = z\ mod2$$

where $z = Hr$. In other terms taking advantage of the fact that H and G are the bases of two dual subspaces then if $t = sG$ is a vector in the vector subspace whose basis is G then $tH^T = 0$. A received vector r such that $rH^T \neq 0$ implicates that the message was distorted by an added noise vector n,

$$rH^T = (t + n)H^T = tH^T + nH^T = nH^T$$

Then the decoding problem is obviously finding t by finding the error vector n.

3.3.1 THE SUM-PRODUCT ALGORITHM

The sum-product algorithm (SPA) was developed to perform iterative decoding on the parity-check matrix H. It was introduced by Gallager along with LDPC codes in his doctoral dissertation in 1963 [4] and it was rediscovered by Mackay and Neal in 1990s [5]. The sum-product algorithm is also called belief propagation algorithm BPA, the name comes from Bayesian inference literature where the algorithm was derived independently [26], we can think of the Tanner graph as a belief or Bayesian network where every bit is the parent of g check-nodes and every check-node is the child of r bits.

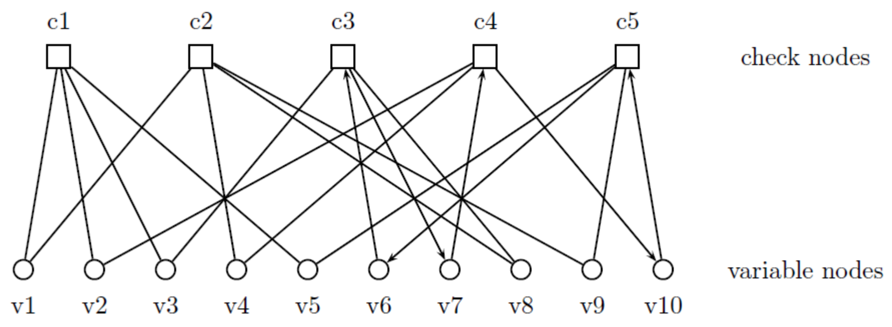

Figure 3.2. Decoding on Tanner graph

SPA is a symbol wise maximum-a posteriori (MAP) decoding algorithm. The objective of the SPA is to compute a posteriori probability (APP) that a specific bit in the transmitted codeword $t = [t_0\, t_1\, \ldots\, t_{n-1}]$ is equal to 1 given the received word $y = [y_0\, y_1\, \ldots\, y_{n-1}]$. Without loss of generality we focus on decoding a bit t_j by calculating the APP:

$$Pr(t_j = 1|y)$$

Considering that we have two competing hypotheses for the value of t_j, namely 1 and 0, we turn our attention to compute the likelihood ratio (LR):

$$l(t_j|y) = \frac{Pr(t_j = 0|y)}{Pr(t_j = 1|y)}$$

Which can be put in the more numerically stable Log- LR or (LLR) form :

$$L(t_j|y) = \log\left(\frac{Pr(t_j = 0|y)}{Pr(t_j = 1|y)}\right)$$

log is assumed to be the natural logarithm.

SPA is a message passing algorithm, it operates by conveying information between Check-Nodes (CN) and Variable-Nodes (VN) of a Tanner graph. In order to understand the SPA we need to be able to answer the following questions:

- What is the role of Check-Nodes and Variable-Nodes in the decoding process?
- What is the nature of the information exchanged between Check-Nodes and Variable-Nodes?

- What effects do the structural properties of the Tanner graph impose on the BPA?

In a Tanner graph there are N variable nodes and each one of these check nodes corresponds to a bit in the $(1 \times N)$ received codeword to be decoded. VNs act as local processors that adds the LLRs values passed to it from the channel and the neighboring CNs to reach the most plausible decision about the value of the bit it represents.

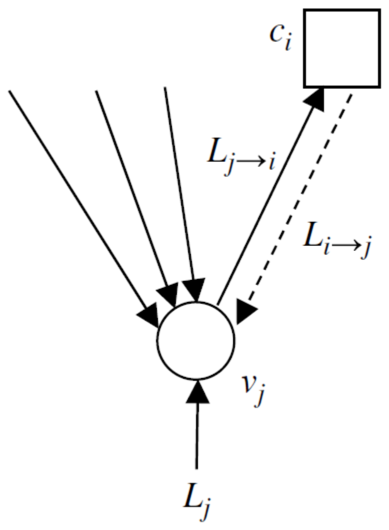

Figure 3.3. A Variable Node

In a Tanner graph there are also M CNs that correspond to the M parity bits added by the encoder. Each CN acts as a local processor that receives LLRs form its neighboring VNs, processes these information and then passes the appropriate information to each of its neighbors.

SPA algorithm is a message passing decoding algorithm. the term "message passing" refers to the process of a collection of low complexity decoders working in a distributed fashion. In SPA VNs and CNs work cooperatively in iterative fashion to reach estimates of the likelihood ratio $L(t_j|y)$ for $j = 0$ to $N-1$. In the first half of every iteration every VN processes the LLRs it received from neighboring CNs and passes extrinsic information to each of its neighboring CNs. In the second half of the iteration every CN processes the LLRs it received from neighboring VNs and passes extrinsic information to every neighboring VN.

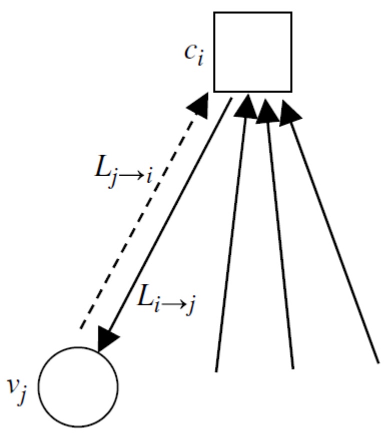

Figure 3.4. A Check Node

Message passing algorithm introduces the concepts of extrinsic and intrinsic information. Message passing imposes the constraint of exchanging only extrinsic information between nodes, the concept of extrinsic information is that a node does not pass any information to another if the other node posseses the information. In the case of SPA, a node N_i does not pass LLR_q to N_j if LLR_q was originally passed from N_j. LLR_q is considered intrinsic information to N_j. A message passing decoding algorithm cannot claimed to be optimal if cycles exist in the Tanner graph. Cycles provide a path through which intrinsic information can travel and reach their corresponding nodes, i.e., the nodes that sent them. Although there exist many procedures to remove cycles from graphs in general or from Tanner graphs in the case of parity-check matrices but most practical codes are suboptimal in the sense that they contain cycles. However, message passing decoding algorithms perform very well for properly designed codes and reach error rates deemed as acceptable for the majority of applications. From the discussion above we can understand the reason to desire Tanner graphs with the largest girth possible.

Before discussing the SPA decoding used in LDPC codes we study the operations of VNs and CNs with more detail so that we can follow the steps of the algorithm more thoroughly.

VNs as Repetition Decoders :

A repetition decoder bases its decision on a majority rule. Given a repetition code that repeats every bit d times then every bit b_i in the source word is transmitted n times and the decoder decision will be 1 or 0 based on the majority in the received d bits. Similarly, the decoder can reach the same decision with soft information i.e., probabilistic information. For a binary repetition code that transmits every bit t n times over a memory-less channel, and r is the received n-vector. A MAP on t can be reached by computing:

$$L(t|r) = \log(\frac{Pr(c=0|r)}{Pr(c=1|r)})$$

Assuming $Pr(t=0) = Pr(t=1)$

$$L(t|r) = \log(\frac{Pr(t=0,r)Pr(r)}{Pr(t=1,r)Pr(r)})$$

$$L(t|r) = \log(\frac{Pr(r|t=0)Pr(t=0)}{Pr(r|t=1)Pr(t=1)})$$

$$L(t|r) = \log(\frac{Pr(r|t=0)}{Pr(r|t=1)})$$

$$L(t|r) = \log(\frac{\prod_{i=0}^{n-1} Pr(r_i|t=0)}{\prod_{i=0}^{n-1} Pr(r_i|t=1)})$$

$$L(t|r) = \sum_{i=0}^{n-1} \log(\frac{Pr(r_i|t=0)}{Pr(r_i|t=1)})$$

$$= \sum_{i=0}^{n-1} L(r_i|x)$$

Where $L(r_i|x)$ is the LLR of the received bit or r_i. The MAP decoder of the repetition code adds the individual LLRs and a majority decision is reached such that $t_l = 1$ if $L(t_l|r) < 0$ and 0 otherwise.

A VN node in a Tanner graph is a repetition decoder in the sense that at the last iteration of decoding it adds the LLRs it receives from all its neighboring CNs, and the channel according to the equation:

$$L_{j \to i} = L_j + \sum_{i' \in N(j)} L_{i' \to j} \tag{3.3.1}$$

and depends on the value of $L_{j \to i}$ to reach a decision on the value of t_j

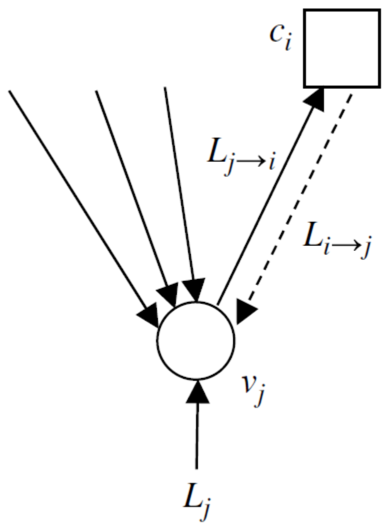

Figure 3.5. VN as a repetition decoder.

During decoding a variable-node VN$_j$ passes extrinsic information $L_{j \to i}$ to a CN$_i$ according to the equation:

$$L_{j\to i} = L_j + \sum_{i' \in N(j)-(i)} L_{i'\to j}$$

In the equation above we can notice that VN$_j$ excludes $L_{i\to j}$ when it is calculating $L_{j\to i}$ so that no intrinsic information is passed. L_j is the LLR computed from y_j the received symbol from the channel using the equation:

$$L_j = L(t_j|y) = \log\left(\frac{Pr(t_j = 0|y_j)}{Pr(t_j = 1|y_j)}\right) \tag{3.3.2}$$

Below we give the probabilities and channel LLRs used for the BEC, BSC and binary-input additive white Gaussian noise (BI-AWGN) channels.

BEC Channel:

t is the transmitted vector and $t_j \in 0, 1$, y is the received codeword and $y_j \in 1, 0, e$ where e stands for an *erasure*. $b \in 0, 1$

$$Pr(t_j = b|y_j) = \begin{cases} 1 & \text{when } \mathrm{y}_j = b, \\ 0 & \text{when } \mathrm{y}_j = b', \\ 1/2 & \text{when } \mathrm{y}_j = \mathrm{e} \end{cases}$$

Where b' is the complement of b, applying the above probabilities to equation 3.3.2 gives:

$$L(t_j|y_j) = \begin{cases} \infty & \mathrm{y}_j = 0, \\ -\infty & \mathrm{y}_j = 1, \\ 0 & \mathrm{y}_j = \mathrm{e} \end{cases}$$

BSC Channel:

$y_j \in 1, 0$, the channel error rate or transition probability $\epsilon = Pr(y_j = b' | t_j = b)$

$$Pr(t_j = b | y_j) = \begin{cases} \text{1-}\epsilon & \text{y}_j = b, \\ \epsilon & \text{y}_j = b' \end{cases}$$

And

$$L(t_j | y_j) = (-1)^{y_j} \log(\frac{1-\epsilon}{\epsilon})$$

BI-AWGN Channel:

$y_j = x_j + n_j$ where $x_j = (-1)^{v_j}$ such that $x_j = +1$ or (-1) for $v_j = 0$ or (1). n_j is Gaussian channel noise $\mathcal{N}(0, \sigma^2)$, the value of σ or an estimate must be known to the decoder in order to calculate the LLR as :

$$L(t_j | y_j) = 2y_j / \sigma^2$$

CNs as Single-Parity-Codes:

A single parity-check (SPC) of length n on $\mathbb{F}_2$ is a linear code which consists of $n-1$ information bits, and a single parity-check bit. The value of the parity-check bit is equal to the $mod\,2$ sum of the $n-1$ information bits thus the generated codeword always sums to $0\,mod\,2$ and has an even number of 1s.

Parity-check bits added by the LDPC code follow the same logic. Keeping this structure in mind when we design SPC decoder. Suppose a codeword $c(1 \times n)$ is encoded using SPC and we receive the codeword r which was transmitted over some channel. Without loss of generality we choose c_b to decode by conditioning on the received vector r and the fact that for the received vector r to be correct it must have an even weight:

$$\hat{c_b} = arg \max_{x \in 0,1} Pr(c_b = x | r, SPC)$$

Then,

$$Pr(c_b = 0 | r, SPC) = Pr(c_0, c_1, \ldots, c_{b-1}, c_{b+1}, \ldots, c_{n-1} \text{ has an even number of 1s} | r)$$

$$= \frac{1}{2} + \frac{1}{2} \prod_{l=1, \neq b}^{n-1} (1 - 2Pr(c_l = 1 | r_l))$$

This can be rewritten as:

$$2Pr(c_b = 0 | r, SPC) = 1 + \prod_{l=1, \neq b}^{n-1} (1 - 2Pr(c_l = 1 | r_l)) \tag{3.3.3}$$

Where $Pr(c_l = 0|r, SPC) + Pr(c_l = 1|r, SPC) = 1$ Then we can rewrite 3.3.3 as:

$$1 - 2Pr(c_b = 1|r, SPC) = \prod_{l=1,\neq b}^{n-1} (1 - 2Pr(c_l = 1|r_l)) \qquad (3.3.4)$$

Using the relation between binary random variables $1-2p_1 = \tanh(\frac{1}{2}\log(\frac{p_0}{p_1}))$ we rewrite 3.3.4 as :

$$\tanh(\frac{1}{2}\frac{Pr(c_b = 0|r, SPC)}{Pr(c_b = 1|r, SPC)}) = \prod_{l=1,\neq b}^{n-1} \tanh(\frac{1}{2}\frac{Pr(c_l = 0|r_l)}{Pr(c_l = 1|r_l)}) \qquad (3.3.5)$$

We already have defined $\log(\frac{p_0}{p_1})$ as the Log-Likelihood Ratio (LLR) and applying this to 3.3.5 gives:

$$\tanh(\frac{1}{2}L(c_b|r, SPC)) = \prod_{l=0,\neq b}^{n-1} \tanh(\frac{1}{2}L(c_l|r_l)$$

$$L(c_b|r, SPC) = 2\tanh^{-1}(\prod_{l=1,\neq b}^{n-1} \tanh(\frac{1}{2}L(c_l|r_l)) \qquad (3.3.6)$$

The MAP decoder of the SPC code makes decisions based on the outcome of 3.3.6 such that $\hat{c} = 1$ if $L(c_b|r, SCP) < 0$ and $\hat{c} = 0$ otherwise.

$$L_{i\to j} = 2\tanh^{-1}(\prod_{j\in N(i)-(j)}^{n-1} \tanh(\frac{1}{2}L_{j\to i})) \qquad (3.3.7)$$

The MAP decoder above was developed for SPC codes. However the CNs in a tanner graph can apply 3.3.6 to process the LLR values it receives from the VNs it is connected to since every CN and the VNs connected to it adhere to the constraint of SPC code, i.e., their sum is equal to $0\,mod2$. $L_{i\to j}$ in 3.3.7 is the value of the *extrinsic* LLR a CN_i sends to its neighboring VNs. The notation $j \in N(i) - (j)$ refers to the fact that when computing $\text{LLR}_i \to j$ a CN_i processes the LLRs sent to form all neighboring VNs except j so that only extrinxic information is sent.

Sum-Product Algorith:

Having discussed the sum-product algorithm and investigated the operations performed at each iteration in terms of: 1) type of message communicated by the BPA and the rules than control this message exchanging process between Check-Nodes and Variable-Nodes 2) the local operations performed by the VNs and CNs to process these message, below we summarize the SPA as given in [21] p. 220:

Algorithm 1 The Gallager SumProduct Algorithm:

1. Initialization: For all j, initialize L_j according to 3.3.2 for the appropriate channel model. Then, for all i, j for which $h_{ij} = 1$, set $L_{j\to i} =$ Lj.
2. CN update: Compute outgoing CN messages $L_{i\to j}$ for each CN using 3.3.7

$$L_{i\to j} = 2\tanh^{-1}(\prod_{j\in N(i)-(j)}^{n-1} \tanh(\frac{1}{2}L_{j\to i}))$$

and then transmit to the VNs.
3. VN update: Compute outgoing VN messages Lji for each VN using Equation 3.3.1

$$L_{j\to i} = L_j + \sum_{i'\in N(j)-(i)} L_{i'\to j}$$

and then transmit to the CNs.
LLR total: For $j = 0, 1, \dots, n1$ compute:

$$L_{j\to i} = L_j + \sum_{i'\in N(j)} L_{i'\to j}$$

Stopping criteria: For $j = 0, 1, \dots, n1$, set

$$\hat{v}_j = \begin{cases} 1 & \text{if } L_j^{total} < 0, \\ 0 & \text{else,} \end{cases}$$

to obtain $\hat{v}$. If $\hat{v}H^T = 0$ or the number of iterations equals the maximum limit,stop; else, go to Step 2.

Chapter 4

4. RAPTOR CODES

Raptor codes are an extension of LT codes [3], which makes them a class of Fountain codes. LT codes like other Fountain codes have a non-constant average encoding cost, and a nonlinear decoding cost. The motivation behind Raptor codes is to provide linear and fast encoding and decoding algorithms for an LT based code. Raptor codes are Forward Error Correction (FEC) codes that provide application-layer protection against packet losses. They were proposed by Amin Shokrallahi in late 2000, and filed for patent in 2001. All variations of Raptor codes outperform LT codes in terms of computational cost and decoding performance. Raptor codes also have better over-head failure performance in comparison to LT codes [27]

4.1 DESIGN OF RAPTOR CODES

In order to better understand the design of Raptor codes we need to understand the factors which led to proposing Raptor codes as an extended version of LT codes.

Let $\Omega_0, \Omega_1, , \Omega_k$ denote the probability distribution on $0, 1..., k$ such that Ω_i is equal to the probability of the value i being chosen. This distribution is typically represented by a generator polynomial $\Omega(x)$.

$$\Omega(x) = \sum_{i=0}^{k} \Omega_i x^i = 1$$

where $\Omega_0 = 0$ but it is kept for convenience. The notation above can be used to represent the output degree distribution of LT codes in the following manner:
the distribution $\Omega(x)$ can be used to denote the probability distribution on a vector space $\mathbb{F}_2^k$.

Given a Fountain code such as LT codes, where $\Omega(x)$ denotes the Robust Soliton Distribution , this Fountain code can be defined by its two parameters $(k, \Omega(x))$. Encoding is the process of linear mapping $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^N$ where N is potentially an infinite number but in practice it is some finite number. Considering the above notation the encoding process proceeds as follows; for a given source block of k symbols $(x_1, \ldots, x_k)$ every output symbol (which we previously used to call encoding symbol) is generated independently by sampling the degree distribution $\Omega(x)$ for a degree d and then uniformly at random a vector v of weight d is chosen from the vector space $\mathbb{F}_2^k$ and the output symbol is computed as $\sum_{i=1}^{k} v_i x_i$

An LT code $(k, \Omega(x))$ possesses a reliable decoding algorithm if the algorithm can recover all k input symbols from any set of size $n > k$ of output symbols, with error probability at most $1/k^c$, where c is a positive constant. The decoding graph of LT codes is a bipartite graph with k input nodes on one side, and n output nodes on the other sides representing the collected encoding symbols by the receiver side. The error probability of the decoder is lower-bounded by the probability that there exists uncovered input symbols since it is not possible to identify input symbols that did not participate in the encoding process. This leads to a lower bound on the number of edges in the graph of $ck\log(k)$. Considering that the encoding cost of LT codes is the expected number of operations sufficient to generate an encoding symbol then the encoding cost of LT codes is at least of the order $\log(k)$, or as stated in [3] $O(\ln(k/\delta))$. For the desirable case of n that approaches k, output symbols need to have an average weight of $\Omega(\log(k))$.

To analyze the decoding cost we assume a BEC channel, and Maximum Likelihood (ML) decoding, which would be equivalent to Gaussian elimination. With the assumption above the decoding process becomes equivalent to solving a consistent system (a system that has a solution) of n linearly independent equations corresponding to the collected output bits for the values of input bits $(x_1, \cdots, x_k)$. For the system above to be solvable its corresponding $(n \times k)$ matrix must have a full rank. The cost of solving such a matrix is $O(nk^2)$ operations which means that the average decoding cost per symbol is $O(nk)$. [3] states that for LT codes the average decoding cost is $O(k \cdot \ln(k/\delta))$.

From the discussion above we find that LT codes have a non-constant average encoding cost of $O(\log(k))$ symbol operations per encoding symbol, and a nonlinear decoding cost of $O(k \log(k))$ symbol operations per source block. Raptor codes designed in [1] has the following properties; for a given source block of k symbols, and overhead $\varepsilon > 0$ Raptor codes have an average number of $O(\log(1/\varepsilon))$ symbol operations per generated encoding symbol, and an average decoding cost of $O(k \cdot \log(1/\varepsilon))$ per source block with failure probability of $1/k^a$ for a positive constant $a > 1$ that is independent of ε. The favorability of Raptor codes in terms of computational complexity is clear.

The reason it is difficult to construct LT codes with constant average degree is because the decoding graph needs at least $k \log(k)$ edges in order to ensure all input symbols are covered otherwise there is a high probability that a fraction of the input symbols would not be covered, these input symbols therefore would never be recovered. The solution proposed by Raptor codes to solve this issue is to precode the input symbols by a traditional erasure correcting code C, and then apply an LT code with a constant average output degree to the symbols generated by the precode C.

Raptor codes can be parameterized by $(k, C, \Omega(x))$, k denotes the number of input symbols. The precode C is linear block code of dimensions (k, n), and typically the precode used for Raptor codes is a high rate LDPC code, so hence forth we use the terms precode, and LDPC code interchangeably. The precode encodes the k input and adds a relatively small redundancy

$(n-k)$ to produce $n > k$ symbols known as the *intermediate symbols.* The intermediate symbols are used to generate the *output symbols* by an LT code with degree distribution $\Omega(x)$.

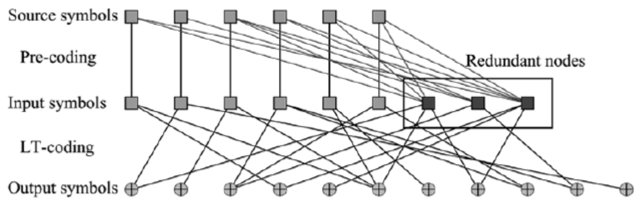

Figure 4.1. Graphical representation of Raptor codes

Below we give a toy example to describe the mechanism in which Raptor codes operates, and how the encoding porcess porceeds:

In Figure 4.2 input symbols x1, . . . ,x6 are encoded using an LT code. We can observe that x2 was not covered by the LT code and hence it will not be recovered by the decoder.

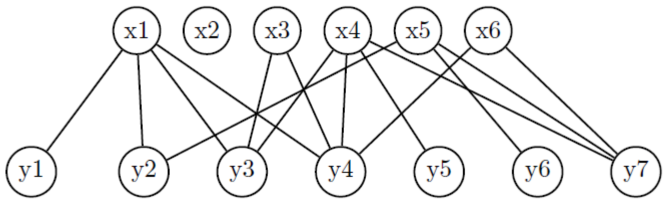

Figure 4.2. LT code

Now we apply Raptor code to the same input set, first we encode the input symbols by a precode such that two redundancy symbols z1, and z2 are added as shown in Figure 4.3 below:

Next we apply an LT code to the input symbols x1, . . . , x6 in addition

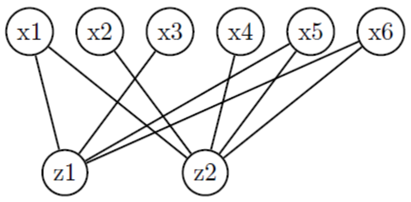

Figure 4.3. Precode of Raptor codes

to z1, and z2 which are collectively called intermediate symbols. The code graph will be as illustrated by Figure 4.2. Despite the fact that x2 was not covered by the LT code but it was covered by the precode, and it can be recovered in the LDPC decoding part of Raptor codes.

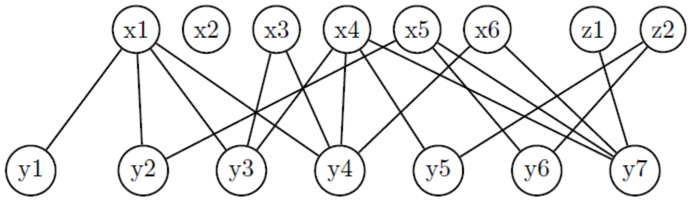

Figure 4.4. Raptor code

4.2 DECODING OF RAPTOR CODES

The decoding graph of length m Raptor code is a bipartite graph that consists of n nodes on one side called the input nodes, they represent the n output symbols of the LDPC code, where the objective of the decoding process is to identify their correct values. On the other side there are $m + n - k$ nodes, that are called the output nodes, and they can be divided into to sets.

One set consists of the m output bits or symbols collected from the channel

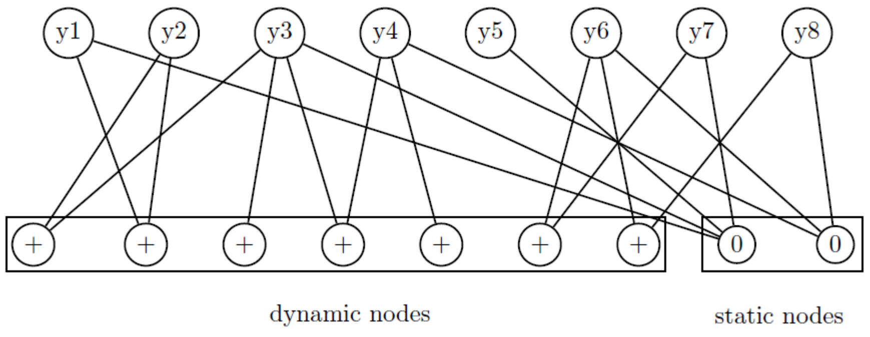

Figure 4.5. Decoding graph of Raptor code

and each node in this set is connected to the input nodes of which it is a neighbor, and its value is equal to the sum of their values. This set is named as the dynamic set. The name dynamic comes as this part of the graph depends on the particular set of m encoding symbols collected by the decoder. The dynamic set corresponds to the LT code part of Raptor codes.

The other set of $n - k$ nodes are called the static set and it corresponds to the $n - k$ parity-check bits added by the LDPC code. Remember that the decoding graph is reconstructed at the decoder side based on information passed by the encoder.

Typically decoding algorithms start the decoding process on the dynamic part and after a pre-fixed number of iterations it shifts to decoding on the static part. Decoding on the dynamic part is actually an LT decoding process, and decoding on the static part is an LDPC decoding process on the parity-check matrix H of the precode of the Raptor code. The inner workings of each phase is governed by the rules, and procedures related to each code as given in sections 2.1.2 and 2.2.3.

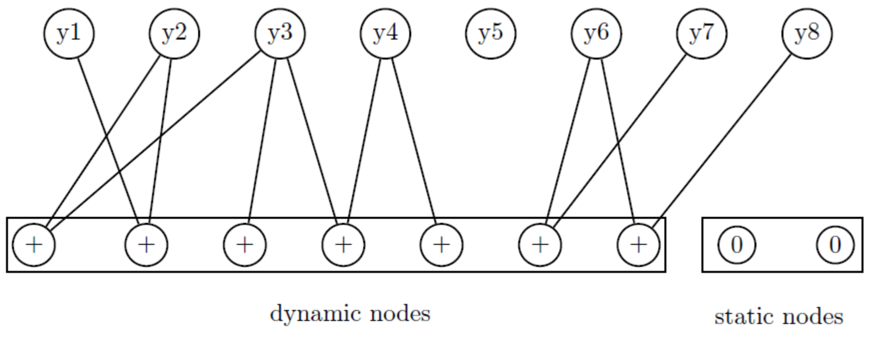

Figure 4.6. Decoding on the dynamic part

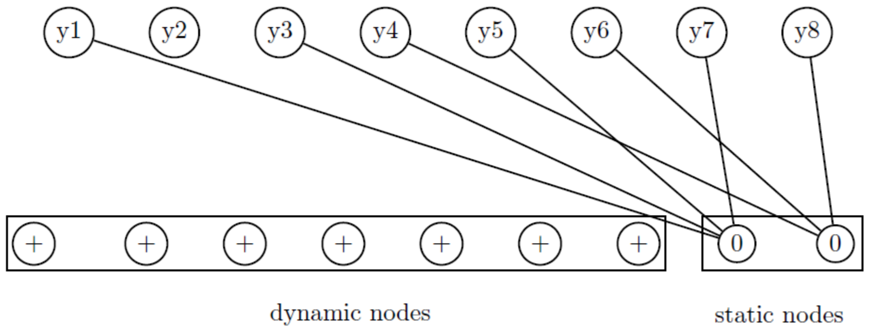

Figure 4.7. Decoding on the static part

Raptor codes are a concatenation of two codes; LDPC codes, and LT codes. It is necessary for the two codes to be interfaced properly, and in a careful manner. The m symbols we collect from the channel are output symbols of Raptor codes which were encoded by an LT code before transmission. As we mentioned earlier, decoding starts on the dynamic part of the decoding graph which corresponds to the LT-decoding part of the Raptor decoder. As the belief-propagation algorithm (BPA) proceeds, messages from output nodes to input nodes $m^l_{o,i}$ and from input nodes to output node $m^l_{i,o}$ are conveyed back and forth according to rules and constraints of BPA algorithm. For more on BPA check section 3.3.1. During the decoding process, the output nodes act as local single parity-check (SPC) decoders, they process $m^l_{i,o}$s, which are LLRs. They receive and send $m^l_{o,i}$ to their neighboring input nodes by applying the equation:

$$\tanh\left(\frac{m^l_{o,i}}{2}\right) = \tanh\left(\frac{Z_o}{2}\right) \cdot \prod_{i \neq i} \tanh\left(\frac{m^l_{i,o}}{2}\right)$$

Z_o is the channel LLR of every output bit o. It is computed from the value of the corresponding symbol received from the channel using equation 3.3.2.

Input nodes act as local Repetition processors. They receive LLRs from neighboring output nodes ($m^l_{o,i}$), and apply equation 4.2.1 to compute the intrinsic information they pass to the output nodes they are connected to.

$$m^{l+1}_{i,o} = \sum_{o' \neq o} m^l_{o',i} \tag{4.2.1}$$

After a pre-set maximum number of iterations decoding stops on the dynamic part of the graph, and every input node computes its total LLR as $\sum_o m_{o,i}$. Next decoding on the static, or LDPC part of the graph starts. Decoding is initiated by input nodes passing their LLRs computed in the dynamic phase of decoding as channel LLR to the LDPC decoder, and again input nodes act as Repetition decoders while the nodes of the static set act as SPC decodes. After a maximum number of iterations has been reached, or a stopping criterion has been met decoding stops, and the decoder declares a value for each bit of the input bits. The decoded n bits are the output of the

LDPC code that was used as precode and they contain the k original source bits which can be seen readily if the code is systematic.

4.3 THE OUTPUT DEGREE DISTRIBUTION

In Raptor codes the Robust Soliton distribution is not used as the output degree distribution for the LT code part, instead as shown in [1] a new degree distribution is developed based on heuristic analysis which leads to optimized degree distributions for different values of source data size k.

The analysis is performed on the dynamic (LT) part of the decoding graph. Before going further into the analysis we need to define few parameters of the graph that are used in the analysis.

w_i is the probability that a randomly chosen edge is connected to a degree i output node, $w(x) = \sum_i w_i x^{i-1}$. Ω_i is the probability that a randomly chosen output node is of degree i, $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$. ι_i is the probability that a randomly chosen edge is connected to a degree i input node, $\iota(x) = \sum_i \iota_i x^i$. The following relations exist between the parameters:

$$w(x) = \frac{\Omega'(x)}{\Omega'(1)}$$

$$\iota(x) \triangleq e^{\alpha(x-1)} \tag{4.3.1}$$

Assuming belief propagation for decoding, we view the decoding process from a binary prospective that is the messages exchanged are 0 or 1 for the sake of simplifying the analysis. An input node sends a message 1 to a neighboring output node if its value has been recovered and an output node sends a message 1 to an input node if it has identified its value and vice versa. Let p_i denote the probability that a randomly chosen edge carries a message 1 from a output node at iteration i, then we have the recursion:

$$p_{i+1} = w(1 - \iota(1 - p_i)) \tag{4.3.2}$$

This comes from the and-or tree analysis argument [28] that an input node (Variable Node (VN)) needs to receive at least one message of value 1 to be decoded which makes the probability to be decoded $1 - (1 - p_i)^{d_{in}}$ where d_{in} is the degree of the input node, while for an output node (Check Node (CN)) to send a 1 to a neighbor it needs to receive message 1 from all the other neighbors. Then the probability of a CN sending a message 1 becomes $(1 - (1 - p_i)^{d_{in}})^{d_{out}-1}$.

Now we define u_i as the probability that an input symbol is recovered at iteration i. u_i is equal to $1 - (1 - p_i)^d$ since an input symbol is recovered if it receives a message 1 from any of its neighboring d output nodes. In terms of the decoding graph this probability is written as $1 - \iota(1 - p_i)$. Applying 4.3.1 we get:

$$1 - \iota(1 - p_i) = 1 - e^{-\alpha p_i} = u_i \tag{4.3.3}$$

and

$$p_i = -\ln(1-u_i)/\alpha \tag{4.3.4}$$

$u_{i+1} = e^{-\alpha p_{i+1}}$. Taking a look at equations 4.3.2 and 4.3.3 we can express u_{i+1} as:

$$u_{i+1} = 1 - e^{-\alpha w(u_i)} \tag{4.3.5}$$

Equation 4.3.5 implies that having a fraction-x of recovered symbols at some point i, then in the next iteration $(i+1)$ the fraction increase to $1 - e^{-\alpha w(x)}$, i.e., an increase of $1 - x - e^{-\alpha w(x)}$.

If decoding is performed on $k(1+\epsilon)$ output symbols, then we have $w(x) = \frac{(1+\epsilon)\Omega'(x)}{\alpha}$, and the expected fraction of input symbols in the input ripple is:

$$1 - x - e^{-\Omega'(x)(1+\epsilon)} \tag{4.3.6}$$

hence the expected input ripple size is equal to:

$$k(1 - x - e^{-\Omega'(x)(1+\epsilon)}) \tag{4.3.7}$$

The derivation above is based on heuristic assumptions but it does not affect the findings because after reaching candidate degree distributions the error probability of the LT-decoder is computed for each distribution.

From Luby's analysis in [29] the concept of keeping the expected ripple size equal to, or larger than $c\sqrt{(1-x)k}$ for a positive constant c is adapted. Using this boundary condition on equation 4.3.7 for some ϵ, δ, and k we get:

$$1 - x - e^{-\Omega'(x)(1+\epsilon)} \geq c\sqrt{(1-x)/k}$$

$x \in [0, 1-\delta]$ and :

$$\Omega'(x) \geq \frac{-\ln(1 - x - c\sqrt{(1-x)/k})}{1+\epsilon} \qquad (4.3.8)$$

where ϵ is the overhead, and δ is the fraction of intermediate symbols that are not recovered.

By discretizing the interval $[0, 1-\delta]$ and requiring the above inequality to hold on the discretization points, we obtain linear inequalities in the unknown coefficients of $\Omega(x)$. The optimized degree distributions are for the values of k shown in the table, $\delta = 0.01$, the overhead ϵ used in the optimaztion process, and a is the average degree of output symbols.

Table 4.1. Degree distributions for given values of k

k	65536	80000	100000	120000
Ω_1	0.007969	0.007544	0.006495	0.004807
Ω_2	0.493570	0.493610	0.495044	0.496472
Ω_3	0.166220	0.166458	0.168010	0.166912
Ω_4	0.072646	0.071243	0.067900	0.073374
Ω_5	0.082558	0.084913	0.089209	0.082206
Ω_8	0.056058		0.041731	0.057471
Ω_9	0.037229	0.043365	0.050162	0.035951
Ω_{18}				0.001167
Ω_{19}	0.055590	0.045231	0.038837	0.054305
Ω_{20}		0.010157	0.015537	
Ω_{65}	0.025023			0.018235
Ω_{66}	0.003135	0.010479	0.016298	0.009100
Ω_{67}		0.017365	0.010777	
ϵ	0.038	0.035	0.028	0.02
a	5.87	5.91	5.85	5.83

Chapter 5

5. UNEQUAL ERROR PROTECTION RAPTOR CODES

Codes with the unequal error protection property provide different parts of encoded data with different, and unequal error correction capability according to the relative importance of the different parts of the source data. Considering the vast development in information technology, and the expanding architecture of data communication networks the need for UEP arises in many cases such as transmitting data frames with header, MPEG streaming where I-frames demand more protection than P-frames, and storage on holographic memories.

Codes with UEP property were first studied by Masnick and Wolf in [30] and since then research has continued to develop, and apply the UEP property to existing, and new codes. In our research we focused on studying UEP-LT codes, and UEP-LDPC codes since Raptor codes are a concatenation of these two codes, and our objective is to apply UEP property to Raptor codes.

5.1 UNEQUAL ERROR PROTECTION CODES

Applying the UEP property to different codes requires different approaches based on the structure of the code of interest and different approaches produce different performances and results for the same code. Below we give a number of UEP LT-codes :

- Fekri et al. were the first to develop rateless codes with UEP property in [6] and [7]. For source data of k bits with two levels of importance the bits are divided into two sets. The first set consists of the more important bits (MIB) with $k_1 = \alpha k$ bits $(0 < \alpha < 1)$, and the other set contains the less important bits (LIB) with $k_2 = k - k_1$. They proposed to construct a traditional LT code with one modification. That is, the neighbors of the encoding symbols will be chosen non-uniformly at random, i.e., every time the output distribution is sampled for a degree d, $d_1 = \min([\alpha d k_M], k_1)$ for $(k_M > 1)$ of the selected bits will be from MIB, and $d_2 = d - d_1$ will be selected from the LIB. k_M is the factor that controls what fraction of the neighbors of every encoding symbol with degree $(d < k_1)$ will be from the MIB, and $d_2 = d - d_1$ will be selected from the Less Important Bits (LIB), which increses the probability of selecting bits from MIB. Therefore the probability of recovering the MIB bits by the decoder will be higher compared to LIB bits.

- in [31] the authors employ windowed rateless codes technique originally developed to provide low complexity rateless codes to devise what they call UEP expanding window Fountain codes.

- UEP schemes are also developed to target specific purposes as in [32], where the authors developed UEP rateless code for MPEG video transmission and [33] where LT code is optimized for multimedia multicasting applications.

UEP property can be applied to raptor codes by applying it to the precode part, or the LT code. However, all the attempts we encountered embedded the UEP property into Raptor codes by employing an UEP-LT code that uses one of the methods mentioned above, e.g., [7] uses the method described in [6], and [12] uses the UEP windowed coding method proposed in [31]. Later we came across the only exception [13], where UEP property was applied on both the precode and LT part.

LDPC codes are typically used as a precode for Raptor codes. UEP property has been applied to LDPC codes and many of the relative papers have reported that concrete UEP performance and higher protection on the MIB part can be achieved without much degradation on the LIB part. This led us to the question: how will applying the UEP property on the LDPC part only of Raptor codes affect the performance of UEP Raptor codes? In order to proceed towards answring our question, first we need to investigate the methods of applying UEP to LDPC codes, and choose a UEP-LDPC code compatible with general structure of Raptor codes.

Many methods have been developed to construct UEP LDPC codes with varying complexities, and performances. Below we list few interesting methods we encountered:

1. LDPC codes can be used to provide UEP by employing two Tanner graphs, i.e., two LDPC codes, one to protect all the data, and another with high rate to provide protection for the MIB bits only. This approach was proposed in [8].

2. In [9] the authors used a similar approach to what is mentioned in method (1) above. Data are encoded and decoded using multiple codes by applying Plotkin-type structure which is a code design that increases the code length by concatenating several codes additively, i.e., by adding their codewords. The codes are aligned in a specific order such that the order of decoding controls the level of protection.

3. In [10] Density Evolution (DE) is used to design UEP LDPC codes. The codes are designed with partially regular parity-check matrices.The degree of the columns belonging to the MIB is higher than those of the LIB which results in higher correction capability for the MIB.

The method presented in (3) has lower complexity in terms of design, and cost. It was the method of choice for us to use as a precode in one of the proposed models of UEP Raptor codes with the UEP property in the precode part. In the next section we discuss with detail this method, and the process of designing the code and embedding it into Raptor code.

5.2 DESIGN OF RAPTOR CODES WITH UEP PRE-CODE

5.2.1 USING PARTIALLY REGULAR LDPC CODES

Suppose we want to transmit a data block of k symbols such that the first αk symbols $(0 < \alpha < 1)$ receive higher protection than the rest of the data. When we encode and transmit data we cannot control the error pattern that the channel may impose on the transmitted data, but we can design the FEC code to encode the source data such that the error correction capability on the MIB part of the data is higher than the LIB part. This may involve degrading the error correction performance of the LIB part, however, good UEP schemes provide higher protection on the MIB part with no, or small degradation on the LIB part.

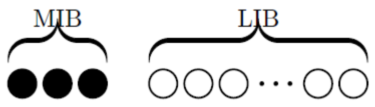

Figure 5.1. Dividing data to MIB and LIB

LDPC codes are linear block codes of dimension k, length N, with Variable-Node (VN) degree distribution $\lambda(X) = \sum_{d=1}^{d_v} \lambda_d X^{d-1}$, and Ceck-Node (CN) degree distribution $\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}$. For more on this see section 3.1.2.

One of the methods to design UEP LDPC codes is to develop parity-check matrices H such that the columns corresponding to the MIB bits has higher weight than the rest of the columns. In a Tanner graph this translates to VNs corresponding to the MIB bits having a higher degree than the rest of the VNs as shown in Figure 5.2.

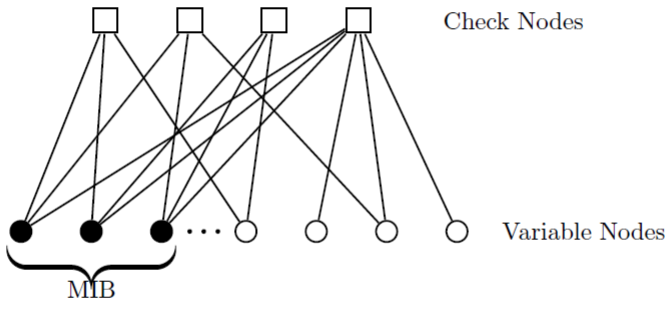

Figure 5.2. Tanner graph of a simple UEP LDPC code

The significance of this type of degree distribution is that the LLR value of higher degree VNs converge faster, and to a higher value compared to those of VNs with smaller degree. This is discussed in [25] on the notion that the slope of the SNR$_{out}$ versus SNR$_{in}$ curve for a degree d node is equal to $d-1$. Intuitively a higher degree VN is connected to more CNs and its LLR has a higher chance of converging to a correct value and here it is trivial to ask why do not we apply this higher degree distribution to the whole code and achieve such a good performance on all bits? Such a design will violate the sparse nature of the parity-check matrix H of LDPC codes. Also, a parity check matrix H with high weight on all columns may lead to significantly degraded BER performance, because as the weight of check nodes increases the probability of the check failing increases [25].

In [10] the authors employ this property of LDPC codes and develop what they call "Unequal Error Protection Using Partially Regular LDPC Codes". The proposed parity-check matrix H has the form $[H_m|H_l|H_p]$ where the sub-matrices H_m, H_l and H_p belong to the MIB, LIB and PB (Parity Bits) with dimensions $((N-K)\times\alpha K)$, $((N-K)\times(1-\alpha)K)$, and $((N-K)\times(N-K))$ respectively. The columns of MIB, LIB, and PB have degrees d_m,d_l, and d_p respectively, where $(d_m > d_l > d_p)$.

UEP Density Evolution is used to assign the values of d_m ,d_l, and d_p which results in very low error rates for the MIB part and still comparable error rates to Equal Error protection (EEP) on the LIB part. The degree optimization is based on two conditions, the number of erassure messages exchanged in iteration $i+1$ is less than that of iteration i, and the ratio of the number of erasures in the MIB part to the number of erasures in the LIB part is an increasing function of the number of iterations.

Table 5.1. Optimized Degree distributions for UEP-LDPC codes and error rates after 25 iterations

Code	d_m	d_l	d_p	d_c	m_{25}	l_{25}	p_{25}
1	23	3	2	7	$2.18e-6$	$1.48e-1$	$2.58e-1$
2	24	4	2	8	$2.31e-12$	$1.52e-2$	$1.45e-1$

The paper reports the optimized degree distributions shown in the table above, which can be adapted to achieve UEP on LDPC codes. The degree distributions are optimized for rate $= 1/2$, and $\alpha = 0.1$. m_{25}, l_{25}, and p_{25} are the observed error rates for MIB, LIB, and PB respectively after 25 iteration of BPA decoding.

As mentioned previously the H matrix has the form $[H_m|H_l|H_p]$. In order for this structure to be valid the submatrix H_p must be full rank. The reason it is necessary for H_p to be full rank arises in the process of deriving the Generator matrix G from the parity-check matrix H.
Let $[H_m|H_l|H_p] = [H_I|H_p]$

Where the submatrix H_I $(M \times K)$ corresponds to the columns that encode the information bits. Proceeding through a procedure similar to what was given in section 3.2 we get:

$$G = [I_{K\times K}\ H_I^T H_p^{-T}]$$

Hence the reason for demanding H_p to be a full rank becomes obvious.

Another constraint to consider are *cycles*. As mentioned in [25], for a parity-check matrix the maximum number of weight-2 columns before a cycle is created is $M - 1 = N - K - 1$. For codes free of these length-4 cycles created by weight-2 columns the submatrix H_p consisting of degree-2 nodes only is a permutation of the dual-diagonal matrix:

In H, H_p has column weight 2 and dimensions $M \times M$. To go around this issue we replace one of the columns of H_p with another of weight one; this

$$\begin{bmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & & \cdots & & \\ & & & & 1 & 1 \\ & & & & & 1 \end{bmatrix}$$

would affect the error rate negligibly.

If we pay attention, the constraint above removes cycles from the submatrix H_p but it does not treat the whole matrix, or prevent columns in H_p from creating length-4 cycles with columns in H_I. A procedure we employed in the software used for simulations in order to remove cycles created by combination of a column in H_p, and another from H_I was to permutate columns with consecutive 1s.

At this point we have the UEP LDPC code we intend to use as a precode in the UEP Raptor code model we have in mind. For the LT part we use the degree distribution $\Omega_1(x)$ given in tabe 4.1:

$$\begin{aligned} \Omega_1(x) &= 0.007969x + 0.49357x^2 + 0.1662x^3 + 0.072646x^4 + 0.082558x^5 + 0.056058x^8 \\ &\quad +0.037229x^9 + 0.05559x^{19} + 0.025023x^{65} + 0.003135x^{66} \end{aligned} \tag{5.2.1}$$

Our next step is to embed this UEP parity-check matrix into the software of Raptor codes we have developed, run simulations, and observe the results. We denote this design of UEP Raptor codes of EEP LT code, and partially regular LDPC code by *code1* henceforth so as to ease distinguishing it from other code designs when presenting simulations results.

5.2.2 USING EEP LDPC CODES WITH DIFFERENT RATES

The rate of a channel code is equal to $R = \frac{k}{N} = \frac{N-M}{N}$, where k is the number of input symbols, N is the codeword length or number of output symbols, and M denotes the added redundancy by the code as a mean to enable the code to resolve possible errors in the received codeword. For any practical channel code the capability to decode received data correctly or achieve an acceptable BER is an increasing function of the size of redundancy M, and as $R = \frac{N-M}{N}$, i.e., lower code rates relates to more added redundancy and hence higher protection.

Suppose we want to encode a data block of k symbols such that the first αk symbols $(0 < \alpha < 1)$ receive higher protection than the rest of the data, then we can encode the first αk MIB bits using an LDPC code of lower rate than of the code we use to encode the $(1-\alpha)k$ LIB bits.

In general UEP codes achieve lower BER on the MIB bits at the expense of degrading the BER of LIB part and hence the overall BER. However, for codes that adapt BP algorithm in the decoding process such as LDPC codes, reduction in the capability of the code to resolve errors in the LIB part can reflect negatively on the performance on the MIB part as well. Lets examine a sample situation where such a condition can be observed:
In a Tanner graph the neighbors of each VN, i.e., the check nodes (CN) that

the VN is connected to are selected randomly. In the same manner the CNs are assigned neighbors randomly which means the bits from the LIB and the MIB parts can be a neighbor of the same CN. For more on Tanner graphs see section 3.1.2.

Bits sharing the same CN depend on each other to be decoded in the following mechanism; suppose a CN sends a message 1, -1 to communicate that the value of VN i.e., a bit is 0,1 and sends a message 0 to communicate that it is unable to identify the value of the VN, and the same applies to messages sent from VNs to CNs. By looking at equation 3.3.6 we can see that a CN would send message 0 to all its neighbors as long as more than one of its neighbors send message 0 which reflect that the value of the VN have not been resolved by the CN yet. Figure 5.3 below illustrates a toy example of the effect of decoding LIB bits can have on decoding MIB bits. If either of the LIB bits $l2$, or $l4$ are not decoded their corresponding VNs will send 0 to cn4 and hence cn4 cannot decode the value of the MIB bit $m3$. Decoding $l1$ leads to decoding $m2$ which in turn leads to decoding $m1$.

Above we examined an example where decoding LIB bits led to a chain effect that resulted in information propagating through the graph and participating in resolving bits in the MIB part.

From the discussion above we can understand that for codes using belief propagation where information travel across the graph decoding bits on any part of the graph can have an effect on decoding the other parts. This takes

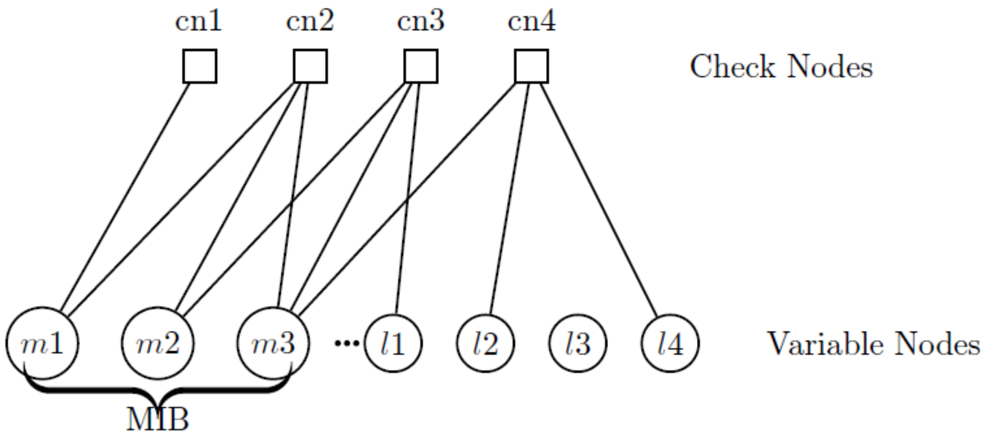

Figure 5.3. The effect of decoding LIB bits on MIB bits in BPA decoding

us to the conclusion that using two EEP LDPC codes with different rates to encode the MIB, and LIB bits, where the degree of protection is controlled by the code rate can lead to better performance on both the MIB and LIB parts.

This design of UEP Raptor codes of two EEP LDPC codes of different rates, and EEP LT code has the flexibility to adapt to different values of α. However, it is more suitable for values of $0.7 \geq \alpha \geq 0.4$ with source data of moderate length $10^3 \leq n \leq 10^4$ so that the LDPC code used for the MIB has adequate length especially taking into account the high rates used in Raptor codes. Also values of $\alpha \geq 0.7$ mean that the performance on the LIB will be unacceptably poor because of the higher rate used for the LDPC code and its short length. On the other hand using UEP LT codes for values of $\alpha \geq 0.4$ can lead to poor performance on both the MIB, and LIB bits. This stems from the mechanism in which UEP LT codes provide higher protection for the MIB bits.

In UEP LT codes the MIB bits receive higher fraction of the average degree of output symbols compared to EEP LT codes e.g., in [7] for every degree sampled from the degree d distribution the αk MIB bits are assigned $d_1 = \min([\alpha d k_M], k_1)$ for $(k_M > 1)$ and $k_1 = \alpha k$ and the LIB part receive the remaining $d_2 = d - d_1$. For the values of $\alpha = 0.1$, and $k_M = 2$ which are seen frequently in the literature $d_1 = \min([0.2d], 0.1k)$ so the $0.1k$ MIB receive a fraction of 0.2 of the average output degree for a ratio of

$$\frac{0.2}{0.1} = 2 = k_M$$

and this increases the probability of recovering the MIB bits at the decoder.

In the same manner for $\alpha = 0.4$ keeping a ratio of two leads to

$$d_1 = \min([0.8d], k_1)$$

i.e., the $0.4k$ MIB bits will posses around 0.8 of the average output degree and this leads to poor performance on the LIB part which also affects the BER of MIB negatively as we show in the simulations because LT codes use BP in the decoding process.

Selecting a smaller ratio such as 1.5 gives

$$d_1 = \min([0.6d], k_1)$$

this improves the BER of both the MIB and LIB parts but affects the desired UEP property negatively and decreasing the ratio k_M further means lowering the average fraction of the output degree assigned to the MIB bits and hence less protection while $k_M = 1$ corresponds to EEP LT code.

The second design scheme of UEP Raptor codes we propose has the structure of two EEP LDPC codes with different rates such that the MIB part of data is encoded by the code with lower rate and an EEP LT code using the degree distribution $\Omega_1(x)$. We call this design *code2* for abbreviation. This design combines the advantages of using EEP LT code, the overall lower BER of EEP LDPC codes with different rates, and the flexibility of this scheme to adapt to higher values of α in order to produce improved performance on both the MIB, and LIB parts compared to code applying the UEP on the LT part.

5.3 SIMULATIONS AND RESULTS

The LDPC part of the software used in simulations is based on the software by Radford M. Neal which can be found at [34] free for research purposes. We modified the software to provide the options of systematic LDPC code and UEP protection and then developed the LT part such that it is easy to interface with the LDPC part or for each part to operate alone.

Our simulations aim at comparing the UEP Raptor code designs we propose, code1, and code2 with UEP Raptor codes that consists of EEP LDPC code, and UEP LT code. The comparisons are based on two criteria: the achieved BER by the decoding process, and the consumed overhead by the code. By studying different codes our objective is to find a code that achieves lower BER while requiring as low overhead as possible.

Figure 5.4 below shows the BER of the MIB bits and LIB bits versus the channel erasure probability for an LDPC code with the parameters: $\alpha = 0.1, d_m = 23, d_l = 3, d_p = 2, d_c = 7$ as given in Table 5.2.1 and code rate = 0.5. The curves are almost identical to what is given in [10].

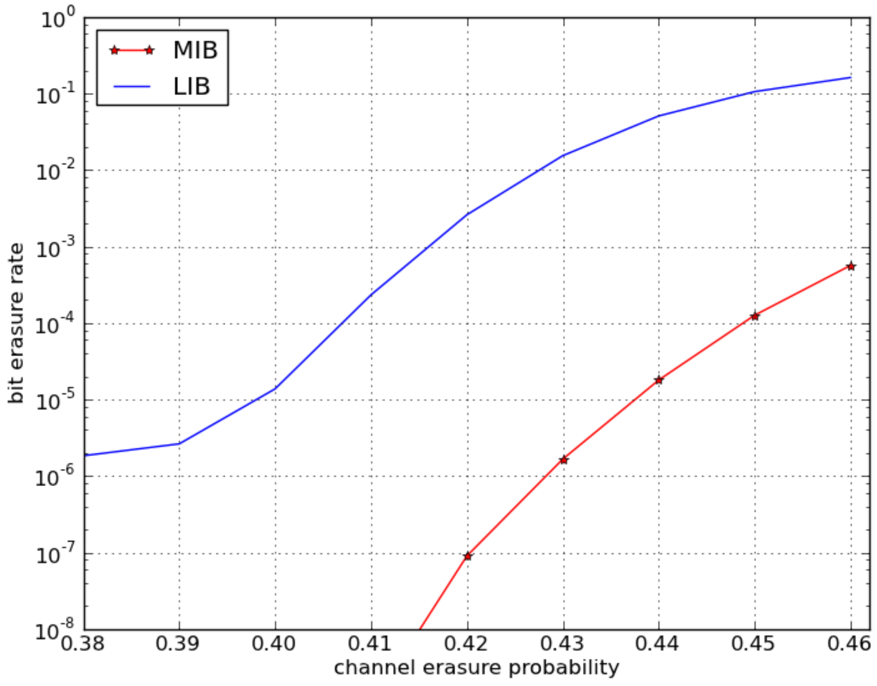

Figure 5.4. BER of MIB and LIB bits vs the channel erasure probability of UEP LDPC code

Figure 5.5 shows the performance of an LT code using degree distribution $\Omega_1(x)$ given by equation 5.2.1 with $k = 2000$ and over binary erasure channel (BEC) channel. The curves MIB and LIB are of an UEP LT code with parameters $K_M = 2$, and $\alpha = 0.1$ and are compared to the EEP curve of LT code with same parameters. The results are similar to what is reported in [7].

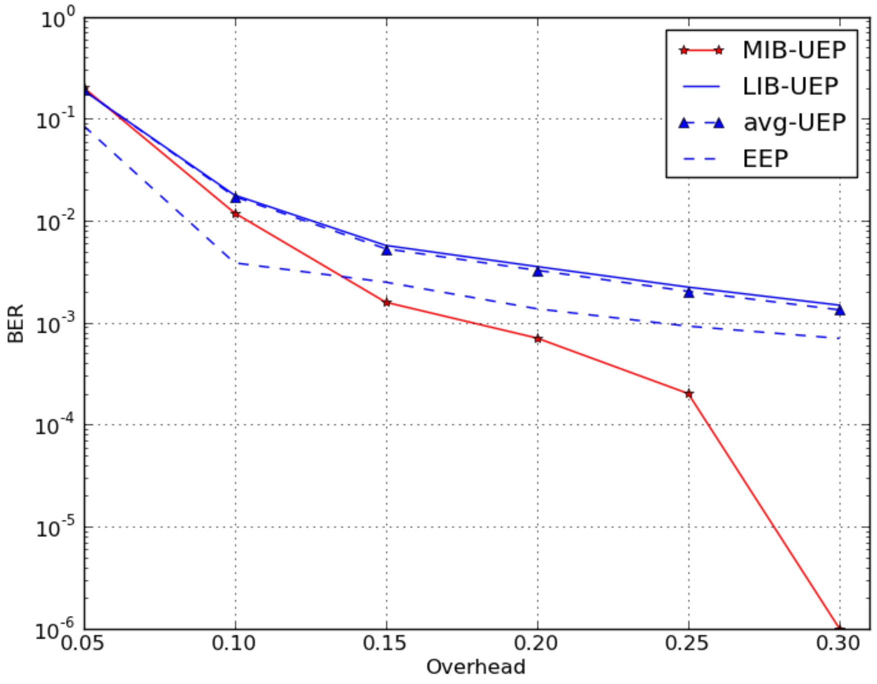

Figure 5.5. BER of MIB and LIB bits of UEP LT-code and an equivalent EEP LT-code

Figure 5.6 illustrates the bit error rate versus overhead performance of LT code using the distribution $\Omega_1(x)$ given by equation 5.2.1. The simulations are performed over binary symmetric channel (BSC) channel with error rate = 0.11. The results are similar to what is reported in [35]. Figure 5.6 depicts the effect of the code length on the performance of the LT code and in fact channel codes in general. As the length increases the code performance improves and the code operates closer to the channel capacity.

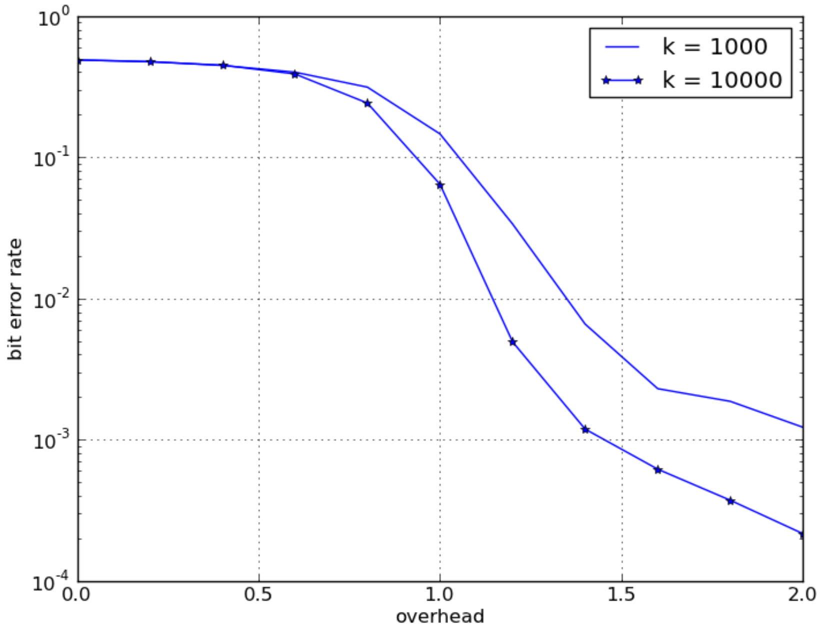

Figure 5.6. Performance of LT codes with distribution $\Omega_1(x)$

In Figure 5.7 one of the curves belong to an LT code with $k = 10000$ bits, and degree distribution $\Omega_1(x)$ and the other to a Raptor code with $k = 9500$ bits, precode is LDPC code with rate = 0.95, column weight = 4 and $\Omega_1(x)$ is used as the output degree distribution. The simulations are performed over BSC channel with error rate = 0.11. The results are similar to what is given in [35].

In Figure 5.8 one of the curves belong to an LT code with $k = 4000$ bits, and degree distribution $\Omega_1(x)$ and the other to a Raptor code with $k = 3800$ bits, precode is LDPC code with rate = 0.95, column weight = 4 and $\Omega_1(x)$ is used as the output degree distribution. The simulations are performed over BEC channel.

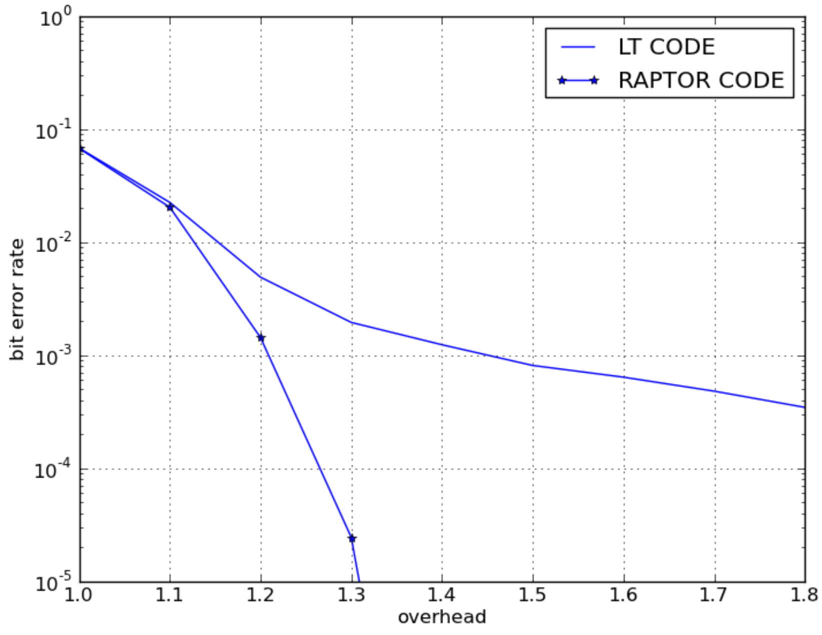

Figure 5.7. BER performance of UEP Raptor codes for LDPC code rate = 0.95

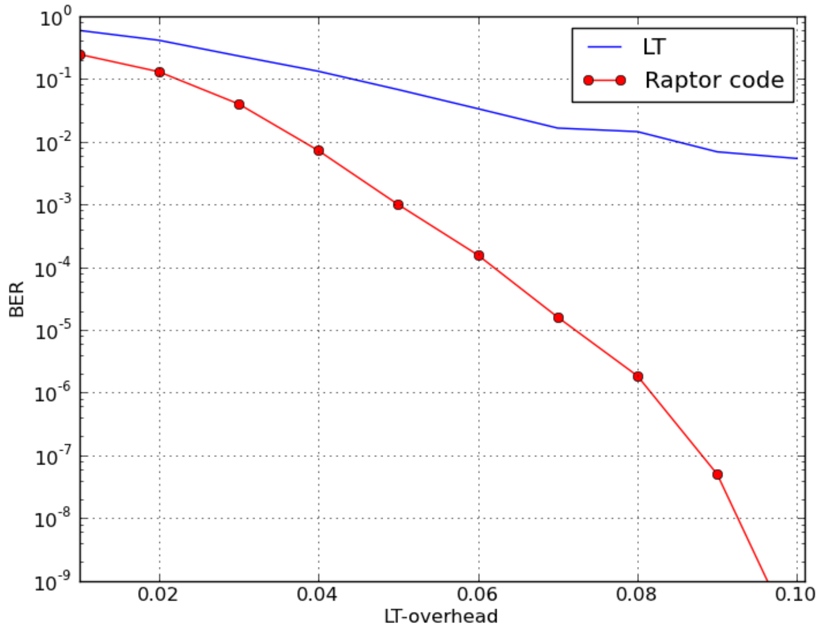

Figure 5.8. BER performance of UEP Raptor codes for LDPC code rate = 0.97

Figures 5.7, and 5.8 show the an important difference between LT codes, and Raptor codes, that is Raptor codes have substantially lower error floors while requiring considerably lower overheads compared to LT codes. This improvement is due to using an outer code (precode) in Raptor codes in addition to the LT code part and this was one of the reasons that initially led us to believe that applying the UEP in the precode part can produce improved results.

Figure 5.9 shows the BER performance of code1 (UEP Raptor codes with partially regular LDPC code as precode) for two rates of LDPC code 0.95 and 0.97 and the abscissa denotes the overhead used for the LT part. The parameters are $k = 3800, \alpha = 0.1$, $\Omega_1(x)$ is the degree distribution adapted by the LT code part, and the LDPC code has the distribution $d_m = 11$,$d_l = 3$ and $d_p = 2$.

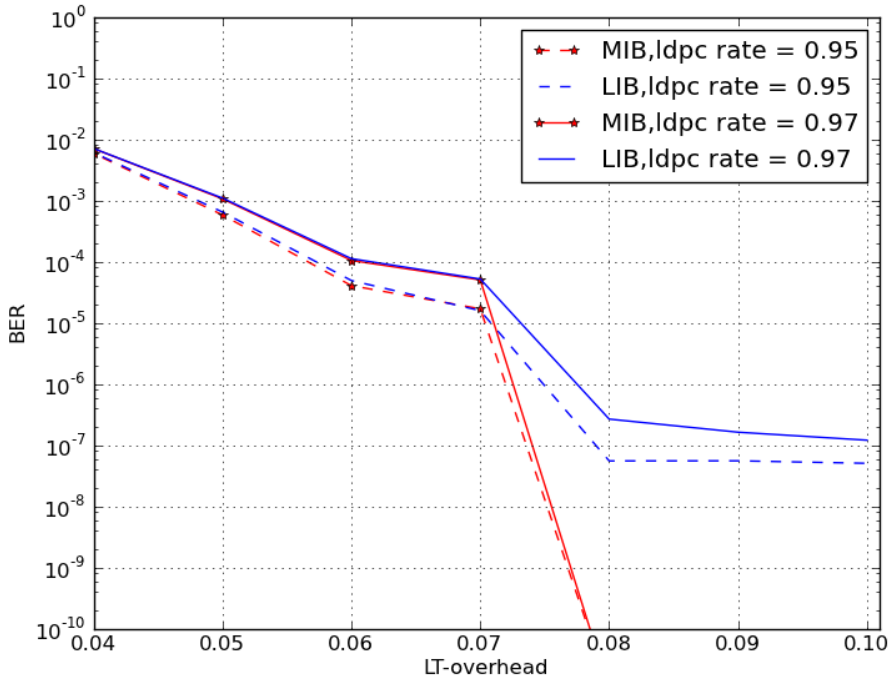

Figure 5.9. BER performance of code1 for different rates of LDPC code

Figure 5.10 shows the BER of MIB and LIB of:

- Code1 consisting of UEP LDPC code with column degree distribution $d_m = 11$,$d_l = 3$ and $d_p = 2$ and EEP LT code with degree distribution $\Omega_1(x)$.
- UEP Raptor code constructed of EEP LDPC code with column weight 4 and UEP LT code with parameters $\alpha = 0.1$, $K_M = 2$ and output degree distribution $\Omega_1(x)$.

For both codes the LDPC code rate = 0.95, k = 3800 and the x-axis denotes the overhead used by the LT part.

Figure 5.11 shows a comparison between the same codes with the same parameters except the LDPC code rate is raised to 0.97.

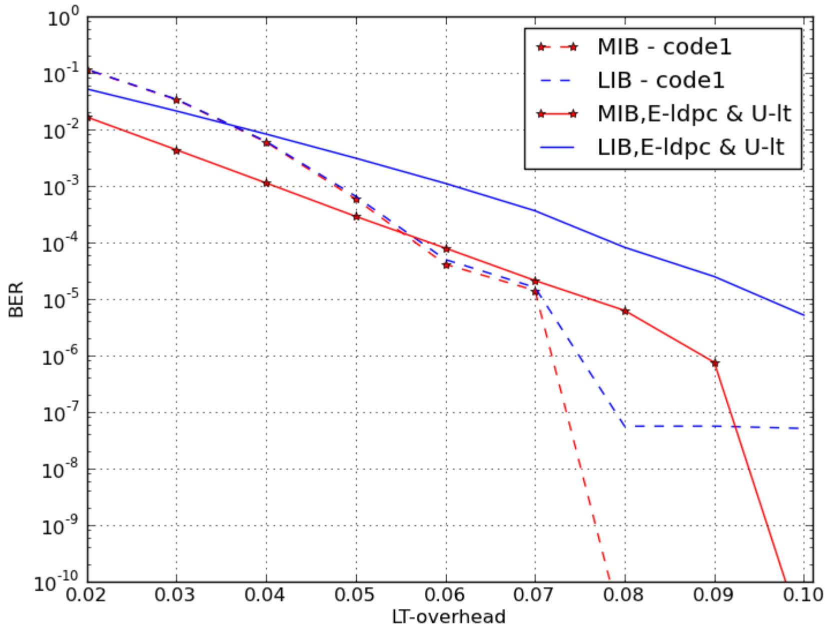

Figure 5.10. BER performance of UEP Raptor codes for LDPC code rate = 0.95

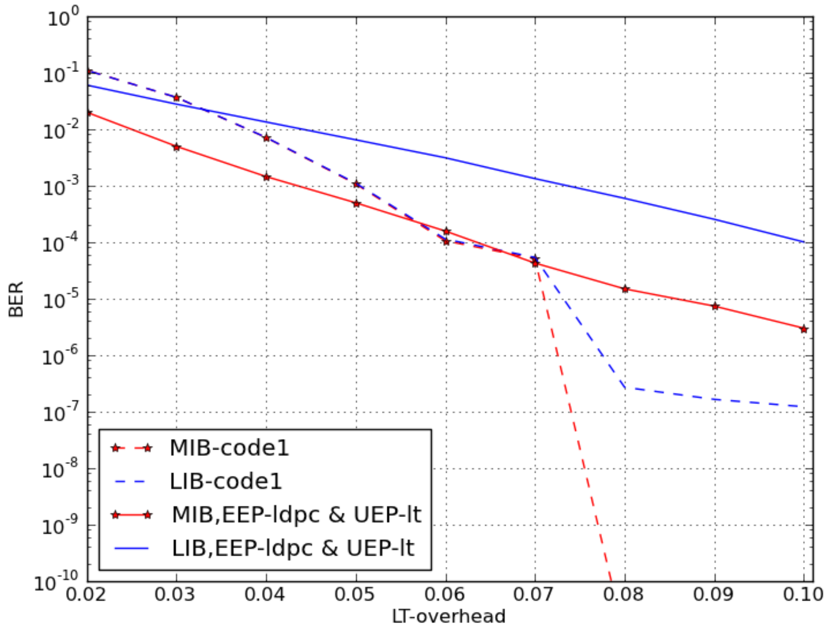

Figure 5.11. BER performance of UEP Raptor codes for LDPC code rate = 0.97

Figures 5.10 and 5.11 demonstrate the BER performance of code1 and UEP Raptor codes with UEP LT code. We observe that the UEP property of code1 does not start until after LT-overhead of 0.07 after which the MIB curve exhibits a strong waterfall effect. Interestingly for LT-overhead γ, $0.4 < \gamma \leq 0.7$, both MIB and LIB curves of code1 are close to or less than the MIB curve of UEP Raptor codes with UEP LT code.

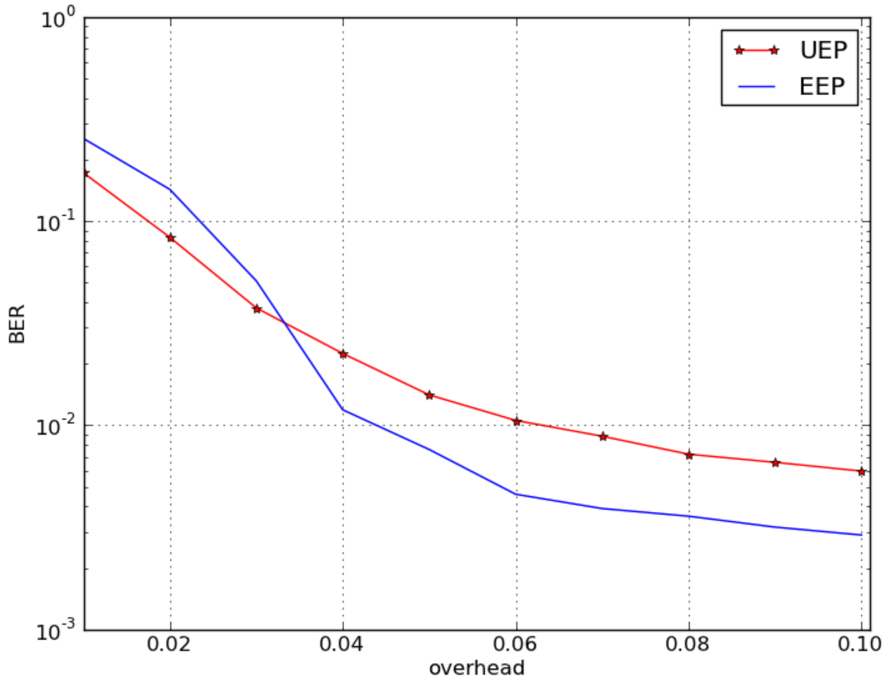

Figure 5.12. comparing overall BER of UEP and EEP LT codes

Figure 5.12 shows the average BER considering all bits MIB and LIB of EEP and UEP LT codes for overhead range of 0.01 to 0.1. The EEP curve drops below the UEP curve after 0.03 and by examining figures 5.10 and 5.11 we notice that the curves of code1 improves after 0.03.

Figure 5.13 illustrates how the performance of code1 is affected by the presence of high number of cycles in the parity check matrix of the LDPC code by using the column degree distribution $d_m = 21$,$d_l = 2$ and $d_p = 2$. For more on cycles see sections 3.1.2 and 5.2.1.

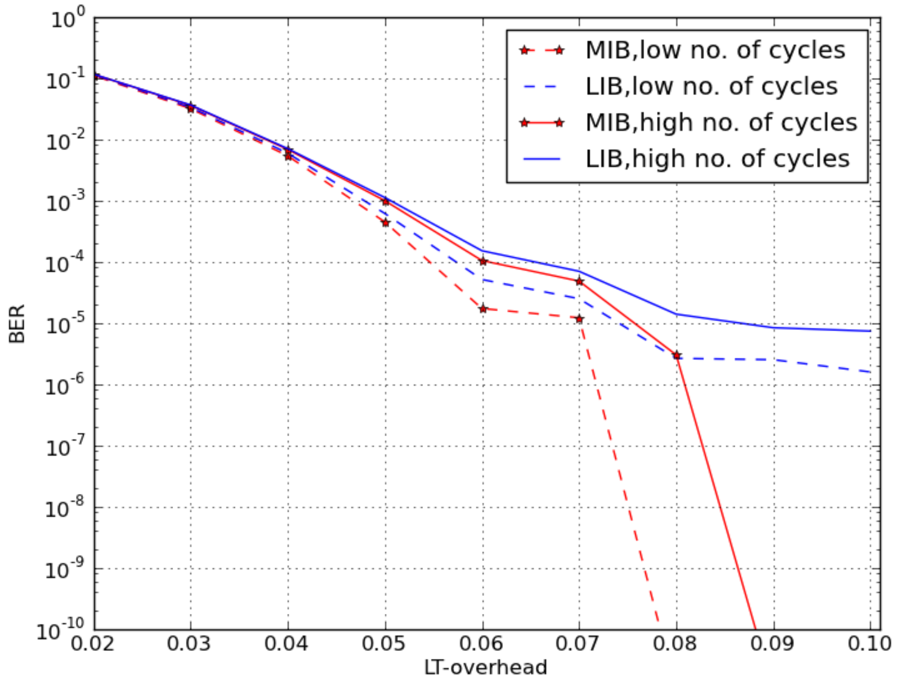

Figure 5.13. performance of UEP Raptor codes with cycles present in the praity check matrix

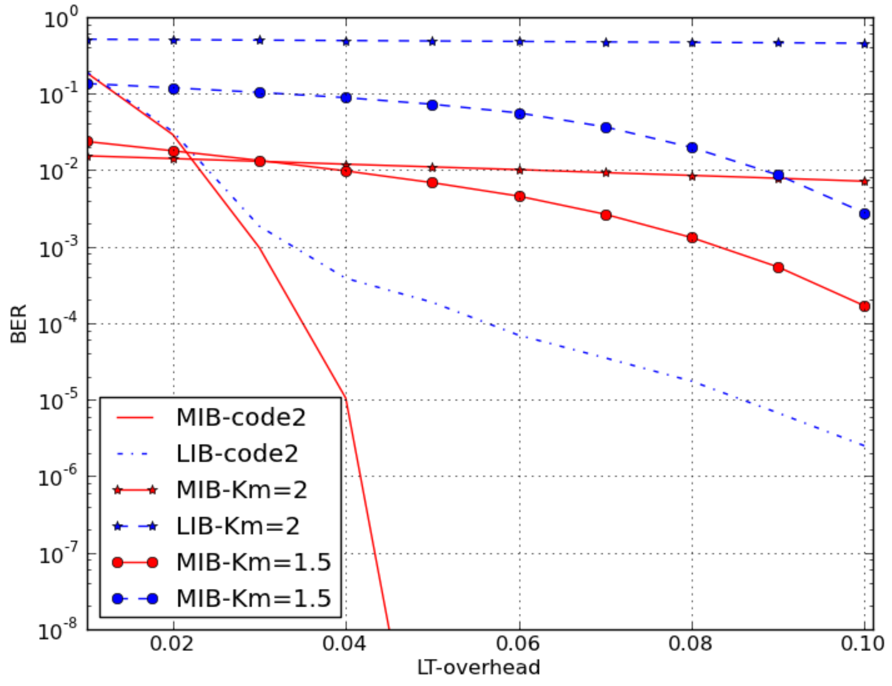

Figure 5.14. BER of code2 and UEP Raptor codes with UEP in LT part for $\alpha = 0.4$

Figure 5.14 shows the BER of:

- code2 with MIB and LIB bits encoded by EEP LDPC codes of rates = 0.95 and 0.985 respectively and both codes have column weight 4.
- UEP Raptor code using UEP LT code with $k_M = 2$ and 1.5.

$k = 10000$ and $\alpha = 0.4$ for both codes. In figure 5.14 we observe that the performance of UEP Raptor codes with UEP LT code is affected adversely when used for value of $\alpha = 0.4$ while code2 operates effectively.

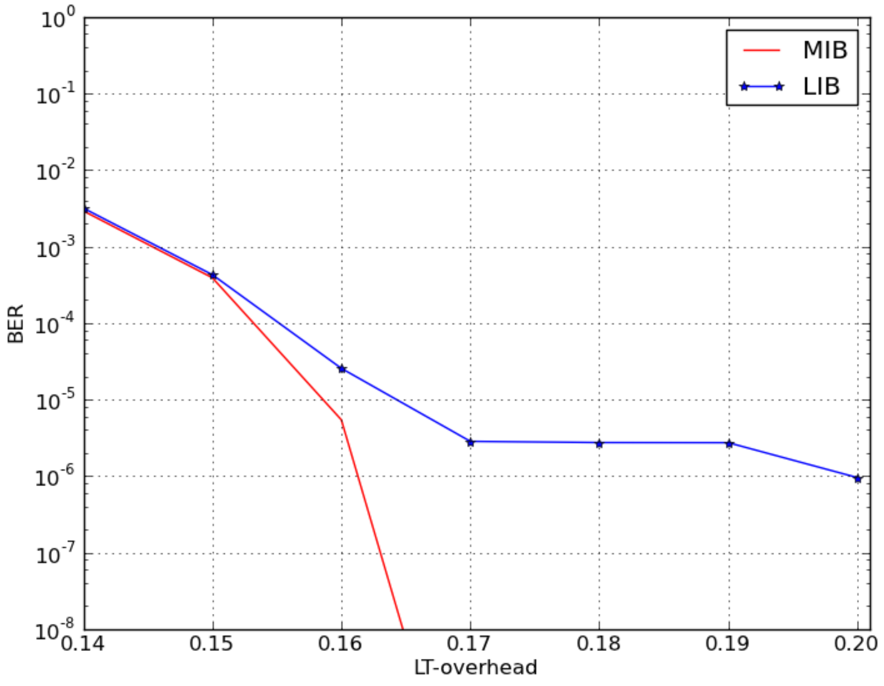

Figure 5.15. Performance of code2 on AWGN channel

Figure 5.15 shows the BER performance of code2 on AWGN channel with $\sigma = 0.5$ and code parameters $k = 10000, \alpha = 0.4$, code rate of LDPC code used for MIB =0.93 and for LIB = 0.97.

Chapter 6

6. CONCLUSIONS AND FUTURE RESEARCH

We study the performance of UEP Raptor codes when the UEP is applied on the precode part of Raptor codes. The design under study has the general structure that consists of a UEP linear block precode, which is an LDPC code, and an EEP LT code.

We propose two schemes with such design. The first consists of EEP LT code, and a partially regular LDPC code, which has UEP properties, as preocde. The second design scheme has the structure of EEP LT code, and in the precode part two EEP LDPC codes of different rates are used such that the code with lower rate is used to encode the MIB part and thus the MIB receive higher protection. In this design the average BER passed from the LT part to the LDPC part of the decoder of Raptor codes is lower compared to UEP Raptor codes with UEP LT codes. The lower average BER leads to improved BER performance on both the MIB and LIB parts of both proposed designs while consuming less overhead compared to UEP Raptor codes with UEP LT code. This lower average BER is an important advantage that the proposed designs have, and it is a major factor in the improved BER performance of the design under study. In fact UEP Raptor codes with UEP LT code can achieve BER that is lower than or close to the BER of the studied design, if it could pass lower average BER from the LT part to the LDPC part of the decoder as we observed in simulations.

Both proposed code designs show good performances for higher rates of the LDPC code compared to UEP Raptor codes with UEP LT codes. The proposed design of UEP Raptor codes with precode consisting of two LDPC codes of different rates has the flexibility to provide UEP for fractions of MIB ($\alpha \geq 0.4$) while UEP Raptor codes with UEP LT codes exhibits poor performance in such cases as was shown.

The proposed design can be applied in cases where UEP Raptor codes are needed. It also can be employed in applications where UEP in general is required such as tranmission of multimedia information based on the good performance exhibited by the design as we can observe in simulations.

The performance of the proposed design of UEP Raptor codes consisting of EEP LT code, and a partially regular LDPC code can be improved by further optimizing the column weight distribution of the parity-check matrix, such that the UEP property starts earlier and for less overheads. Also, existing methods of using partially regular LDPC codes to provide unequal error protection is based on optimizing the column weight (variable node degree) distribution of the parity-check matrix of LDPC codes. However, designing the parity-check matrix for UEP can be improved by optimizing the row weight (check node degree) distribution in addition to the VN degree distribution through analysis or optimization algorithm. This in turn can further improve the UEP property of partially regular LDPC codes in general, and the proposed designs as a consequence.

In the simulations and for research purposes we use what is known as tandem decoding i.e., decoding on the dynamic (LT code) first and then decoding on the static (LDPC code) part of Raptor codes. There are more sophisticated methods such as joint decoding i.e., iterating between the dynamic, and static parts of the decoding graph, or Inactivation decoding, which requires even less overhead for decoding to succeed by combining the optimality of Gaussian elimination with the efficiency of belief propagation. Applying such decoding methods can further improve the performance of the proposed designs of UEP Raptor codes with UEP precode.

BIBLIOGRAPHY

Bibliography

[1] A. Shokrollahi, "Raptor codes,," *Proc. IEEE Int. Symp. Information TheoryTheory, Chicago, IL*, p. 36, Jun./Jul. 2004.

[2] M. M. J. Byers, M. Luby and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *in Proc. ACM SIGCOMM, Vancouver, BC, Canada*, p. 5667, Jan. 1998.

[3] M. Luby, "LT codes,," *in Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, p. 271280, Nov. 2002.

[4] R. G. Gallager, "Low density parity-check codes," *Cambridge, MA: MIT Press*, 1963.

[5] D. MacKay and R. Neal, "Good codes based on very sparse matrices," *Proc. 5th IMA Conf., Cryptography and Coding (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag*, vol. 1025, pp. 100–111, 1995.

[6] N. Rahnavard and F. Fekri, "Finite-length unequal error protection rateless codes: Design and analysis," *Proc. IEEE GLOBECOM*, Nov.Dec. 2005.

[7] B. N. V. N. Rahnavard and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Trans. Inform. Theory*, vol. 53, pp. 1521–1532, Apr. 2007.

[8] N. Rahnavard and F. Fekri, “New results on unequal error protection using ldpc codes,” *IEEE IEEE Communications Letters*, vol. 10, Jan. 2006.

[9] V. Kumar and O. Milenkovic, “On unequal error protection ldpc codes based on plotkin-type constructions,” *IEEE Trans. Commun.*, vol. 54, June 2006.

[10] H. P.-N. N. Rahnavard and F. Fekri, “Unequal error protection using partially regular ldpc codes,” *IEEE Trans. Commun.*, vol. 55, pp. 387–391, Mar. 2007.

[11] A. S. T. Richardson and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inform.Theory*, vol. 47, pp. 619–637, Feb. 2001.

[12] T. T. E. M. P. Cataldi, M. Grangetto and G. Olmo, “Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection,” *IEEE Trans. on Image Processing*, vol. 19, June 2010.

[13] L. Yuan and J. An, “Design of uep-raptor codes over bec,” *European Trans. on Telecomm.*, vol. 21, pp. 30–34, Jan. 2010.

[14] D. Mackay, “Fountain codes,” *IEEE Proc. Commnications*, vol. 152, Dec. 2005.

[15] A. S. D. S. M. Luby, M. Mitzenmacher and V. Stemann, “Practical loss-resilient codes,” *in Proc. 29th Annu. ACM Symp. Theory of Computing*, p. 150159, May 1997.

[16] F. J. L. R. Bahl, J. Cocke and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, p. 84287, Mar. 1974.

[17] A. S. J. Richardson and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *EEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[18] R. M. Tanner, "A recursive approach to low complexity codes," *EEE Trans. Information Theory*, vol. 27, pp. 533–547, Sep. 1981.

[19] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[20] D. J. MacKay and R. M. Neal, "Good codes based on very sparse matrices," *Cryptography and coding: 5th IMA conference*, pp. 100–111, Dec. 1995.

[21] W. E. Ryan and S. Lin, *Channel Codes:Classical and Modern.* Cambridge University press, 2009.

[22] H. P.-N. N. Rahnavard and F. Fekri, "Unequal error protection using partially regular ldpc codes," *IEEE Trans. Commun.*, vol. 55, Mar. 2007.

[23] F. Z. J. Zhao and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (ldpc) codes," *IEEE Trans. Commun.*, vol. 53, p. 549554, Apr. 2005.

[24] T. J. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, p. 638656, Feb. 2001.

[25] Y. L. M. Yang and W. Ryan, "Design of efficiently encodable moderate-length high-rate irregular ldpc codes," *IEEE Trans. Commun.*, vol. 52, pp. 564–571, Apr. 2004.

[26] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA, Morgan Kaufmann, 1988.

[27] M. W. T. S. M. Luby, A. Shokrollahi and L. Minder, "Raptorq forward error correction scheme for object delivery," in *Available at http://tools.ietf.org/html/draft-ietf-rmtbb-fec-raptorq-03*, Aug. 2010.

[28] M. M. M. Luby and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," *in Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms, San Francisco,CA*, p. 364373, Jan. 1998.

[29] M. Luby, "A note on the design of degree distributions," *unpublished*, 2001.

[30] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inf. Theory*, vol. IT-3, p. 600607, Oct. 1967.

[31] A. D. V. e. D. Sejdinovic, D. Vukobratovic and R. J. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Trans. ON Commun.*, vol. 57, pp. 2510–2516, Sep. 2009.

[32] A. Talari and N. Rahnavard, "Unequal Error Prptection Rateless Coding For Efficient MPEG Video Transmission," in *In Proc. IEEE Military Communications Conference, 2009*, MILCOM 2009. IEEE, 2009.

[33] S. B. Yu Cao and W. Chan, "Unequal error protection rateless coding design for multimedia multicasting," *Information Theory Procd.*, pp. 2438 – 2442, June 2010.

[34] R. M. Neal, "avilable at : http://www.cs.toronto.edu/ radford/ftp/ldpc-2006-02-08/index.html,"

[35] R. Palanki and J. Yedidia, *Rateless codes on Noisy Channels.* MITSUBISHI ELECTRIC LABS, 2003.

VITA

Hussein Fadhel graduated top of his class with B.Sc. degree in electronics and control engineering from college of technology, Kirkuk, Iraq in 2007 where he worked as a teaching assistant at the labs of electronics and control engineering department from 2008 to 2010. In 2010 he was awarded a Fulbright scholarship to pursue M.S. degree in electrical engineering. He is currently working towards a M.S. degree in electrical engineering at The University of Mississippi, MS.