# AN INVESTIGATION OF DECODING COMPLEXITY AND CODING RATE PERFORMANCE OF RAPTOR CODES

Thang Nguyen

A thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science
Concordia University
Montréal, Québec, Canada

September 2011

# Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Thang Nguyen**

Entitled: **AN INVESTIGATION OF DECODING COMPLEXITY AND CODING RATE PERFORMANCE OF RAPTOR CODES**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

| | | |
|---|---|---|
| _____ | Dr. Rabin Raut | Chair |
| _____ | Dr. Reza Soleymani | Examiner |
| _____ | Dr. Terry Fancott | Examiner |
| _____ | Dr. William E. Lynch | Supervisor |
| _____ | Dr. Tho Le-Ngoc | Co-supervisor |

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____
Dr. Robin A. L. Drew
Dean, Faculty of Engineering and Computer Science

# Abstract


AN INVESTIGATION OF DECODING COMPLEXITY AND CODING RATE PERFORMANCE OF RAPTOR CODES

Thang Nguyen

This thesis examines two aspects of wireless transmissions using Raptor codes: (i) decoding complexity and (ii) rate performance.

First, observing that the high complexity of Raptor decoding process is mainly due to the required number of decoding attempts, a strategy is proposed to reduce the decoding complexity by choosing an appropriate time to start the first decoding attempt and thus keeping a small number of decoding attempts. Simulations results show that the proposed strategy, when combined with a decoding algorithm, can achieve a significant reduction in Raptor decoding complexity. Another threshold strategy is also investigated, aiming to further reduce the decoding complexity by providing only "reliable" bits for Raptor decoding process. The effect of this considered strategy can be interpreted as simulating a better transmission channel and techniques to estimate its effective channel quality improvement are developed and evaluated.

Second, the Raptor coding rate performance over Nakagami-$m$ fading channels and in a cooperative relaying network using Binary Phase Shift Keying (BPSK) is studied. The simulation results show that the Raptor-coded BPSK scheme can provide a transmission rate closely approaching the channel capacity for different fading conditions at low SNR. For cooperative relaying network using Raptor-coded BPSK scheme, two cooperative protocols are considered: the existing Time Division (TD) and the modified Phase-2 Simultaneous Transmission (PST). Their performance is investigated in terms of average time and energy required for a successful transmission under various conditions of the Relay-Destination (RD) link. The simulation results show that the PST protocol often outperforms the TD protocol in terms of average transmission time and the TD protocol only has lower average transmission energy when the RD link's quality is better that of the Source-Destination (SD) link.

# Acknowledgments

First of all, I would like to express my deepest appreciation for my two supervisors, Prof. Le-Ngoc and Prof. Lynch. They both give me an enormous opportunity to pursue my study here, in Concordia University, which not only broadens my knowledge of the subject but also provides me an invaluable life experience. Without their broad knowledge, enthusiasm, patience and generosity my thesis would not have been possible. For all of that, I am deeply grateful.

I would also like to thank all my friends here, in Montreal, for their kindness and support. One way or another, they all make me feel less lonely and make my time here memorable. It is also true that without them I may not be able to finish my study here.

Finally, my love and thought are with my parents and my sister. Their unconditional and constant love and support are always with me and help me to overcome difficulties in study and life. I would share with them all my lifetime happiness.

# Contents

# List of Figures

# List of Tables

# Abbreviation

| | |
|---|---|
| 3GPP MBMS | 3rd Generation Partnership Project Multimedia Broadcast/Multicast Services. |
| AWGN | Additive White Gaussian Noise. |
| BIMSC | Binary Input Memoryless Symmetric Channel. |
| BPSK | Binary Phase Shift Keying. |
| BSC | Binary Symmetric Channel. |
| CDMA | Code Division Multiple Access. |
| CRC | Cyclic Redundancy Check. |
| DSTC | Distributed Space-Time Code. |
| DVB-H | Digital Video Broadcasting - Handheld. |
| IPTV | Internet Protocol Television. |
| LDPC | Low-Density Parity-Check. |
| LLR | Log-likelihood Ratio. |
| LSPIMPA | Late-Start Previously Inherited Message Passing Algorithm. |
| MPA | Message Passing Algorithm. |
| PIMPA | Previously Inherited Message Passing Algorithm. |
| RD | Relay-Destination. |
| SD | Source-Destination. |
| SNR | Signal-to-Noise Ratio. |
| SR | Source-Relay. |
| TD | Time Division. |
| TH | Two Hop. |

| | |
|---|---|
| TLSPIMPA | Thresholded Late-Start Previously Inherited Message Passing Algorithm. |
| WRN | Wireless Relay Network. |

# Chapter 1

# Introduction

The advantage of Raptor codes over other traditional channel codes is that they are rateless, i.e., the coding rate of a Raptor code for a transmission of a message is not fixed and hence it can be adapted to the channel quality. On the other hand, the downside of Raptor codes is their high decoding complexity caused by the need for multiple decoding attempts. This work investigates some aspects of transmission using Raptor codes over wireless channels, including methods of reducing the Raptor decoding complexity, Raptor coding rate performances over Nakagami-$m$ fading channels and in a cooperative Wireless Relay Network.

This chapter is organized as follows. Section 1.1 introduces the idea of fountain codes which are the general class in which Raptor codes fall. Section 1.2 presents three problems addressed in this thesis. Finally, Section 1.3 provides the thesis outline.

## 1.1 Transmission with Fountain Codes

Messages transmitted over communication channels are susceptible to errors due to transmission effects, i.e., added noise, interference etc., which cause unreliable transmissions. Communication systems have mechanisms to provide reliable transmissions. One of the traditional mechanisms is channel coding. A typical transmission scheme with channel coding is illustrated in Figure 1.1.

At the transmitter, as the message is passed to the channel encoder, the channel encoder encodes it into a codeword using an encoding algorithm. The codeword contains not only the message information but also some redundancy which can later

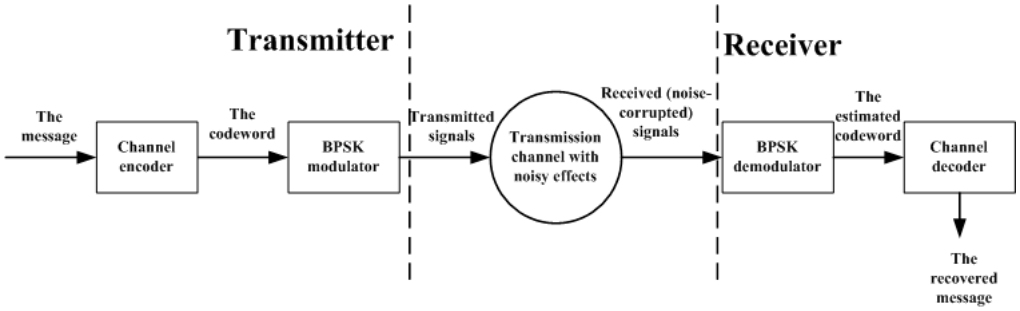



Figure 1.1: A transmission scheme with channel coding.

be used at the receiver to overcome the harmful effects encountered in the transmission of the codeword through the channel. The codeword data (or encoded bits) is then passed to the modulator and mapped into a form suitable for transmission before being sent, through the channel, to the receiver. The channel generally distorts or corrupts the transmitted signal. At the receiver, the demodulator first processes the noise-corrupted received signal and produces an estimated codeword (a sequence of received bits) to the channel decoder. The channel decoder then, based on the coding structure, uses an appropriate decoding algorithm to decode the corrupted codeword. Depending on how the codeword was damaged and the channel code's capability, the channel decoder can detect or correct errors in the corrupted codeword and recover the message.

An important parameter for a channel code is its coding rate, defined as $k/N$ where $k$ is the size of the message segment and $N$ is the size of the codeword. Obviously, $0 < k/N \leq 1$ as the codeword always contains redundancy. $k/N = 1$ stands for the trivial case of no channel code and the message segment is transmitted without any protection; on the other hand, the lower coding rate can provide stronger protection but coming at the cost of more resources for transmitting the redundancy.

Traditional fixed-rate channel codes face a problem of selecting an appropriate coding rate before the transmission. The basic requirement is that the chosen coding rate is high enough to protect the message against the impact of transmission errors. However, as transmission channels are often varying (as in wireless communications), it is hard to have a good estimation for an appropriate coding rate at the transmitter: if the chosen coding rate is too low, the message is overly protected and transmission resources are wasted; in contrast, if the chosen coding rate is too high, the message does not have enough protection leading to unreliable transmission. It is also worth

noting that for many channel codes, once the coding rate is chosen, it is fixed, i.e., it cannot be changed on the fly if that rate was too high or too low.

That problem of fixed-rate channel codes can be addressed with another kind of channel codes, called fountain codes, that can automatically adapt their coding rate to the channel state. Fountain codes [1, 2] work in a way similar to a real fountain. A real fountain continuously pours out water (drops) for a drinker until he/she has drunk enough water and quenched his/her thirst. Analogously, a fountain code encodes a message into a very long stream of encoded symbols and provides them to a receiver until the receiver has collected enough encoded symbols to successfully recover the message. In this way, the fountain coding rate is not fixed before the transmission and can be adapted to the channel state. The unknown coding rate before the transmission of fountain codes makes them rateless codes.

The rateless nature of fountain codes is especially useful for some communication applications. One is the broadcast transmission where the single transmitter has several receivers, each with a channel of varying quality. Using fountain codes, the transmitter does not need to assign coding rates for each receivers. The transmitter simply broadcasts encoded symbols of a message to all the receivers until all of them send back the information to indicate that the message was recovered (e.g., ACK signals). Another transmission scenario where fountain codes are also useful: a source needs to transmit a message to a destination and there are some other devices/nodes, called relays to facilitate that transmission. Such a cooperative transmission system is called a Wireless Relay Network (WRN) and it can make transmissions more reliable and more energy-efficient [3–5]. Applying fountain codes in WRN is one of the methods to closely achieve an information theoretic capacity of the multiple-relay channels [3] and also eliminate outage probability [6], i.e., the probability that the message is not successfully recovered at the destination. Another suitable application for fountain codes is data storage which highlights the ability of overcoming the random losses. If a large file must be backed up, it can be encoded by a fountain code and the encoded symbols can be stored in several places. To retrieve the file, the reader just needs to find an appropriate number of encoded symbols and run the decoding; corrupted encoded symbols due to faults in the device are simply skipped.

On the other hand, one of the downsides of fountain codes is their high decoding complexity [7, 8]. This mainly comes from the potential multiple decoding attempts

at the receiver. As the coding rate of fountain codes is unknown, the receiver tries to decode the first time after a certain number of encoded symbols required to recover the message successfully are collected. If the first decoding attempt fails, more encoded symbols are collected before the receiver tries to decode again. This fail-then-wait-and-try-again decoding process may repeat itself many times until the message is successfully recovered. Moreover, a fountain code usually uses the Message Passing Algorithm (MPA) or the Sum-Product Algorithm [9] in each decoding attempt. This decoding algorithm has many iterations and high complexity [10, 11]. As the decoding process accumulates received symbols, the later decoding attempt will have higher complexity. At the end, such a decoding process consisting of multiple attempts has high complexity.

Raptor codes, which were invented by A. Shokrollahi [12], are one of the classes of fountain codes known for their good performances in a wide variety of transmission channels [13–18]. Due to these good performances, Raptor codes are now used as channel codes for several multimedia broadcast applications, such as Digital Video Broadcasting - Handheld (DVB-H), 3rd Generation Partnership Project Multimedia Broadcast/Multicast Services (3GPP MBMS) and Internet Protocol Television (IPTV) [19, 20]. They are also considered for cooperative transmission systems: with multiple access [21], for broadcast in wireless cellular networks [22] and for cooperative video transmission in ad hoc networks [23]. On the other hand, as Raptor codes are fountain codes, they are also known for their high decoding complexity [7].

In this thesis, several aspects of the Raptor codes' transmission over wireless channels are considered: (i) methods of reducing the Raptor decoding complexity, (ii) Raptor coding rate performance, i.e., the number of encoded symbols required to recover the message successfully, over Nakagami-$m$ fading channels and (iii) over a single-relay WRN. Note that as Raptor codes are now investigated as a channel code at the application layer, it is convenient to use "bit" instead of "symbol" hereafter.

## 1.2 Problem Statement

In this section, three issues with Raptor codes addressed in this thesis are presented.

### 1.2.1 Raptor Decoding Complexity

Raptor codes often have high decoding complexity as they generally require multiple decoding attempts to recover the messages. More specifically, each Raptor decoding attempt involves a high number of encoded bits, many decoding iterations and high computational complexity functions (see Sections 2.1.2.2 and 2.2.2 for details). The high decoding complexity of Raptor codes is undesirable in some applications where the receivers are small and power-limited (e.g., sensor nodes in Ad hoc networks). Chapter 3 of this thesis investigates several methods to reduce Raptor decoding complexity. These methods are compared based on two figures of merits: the total Raptor decoding complexity and the number of encoded bits required for the successful recovery of the message.

The total Raptor decoding complexity is calculated as the overall computational cost of a Raptor decoding process. It consists of two separate components: (i) $C_{R1}$ is the computational cost related to the basic operations, i.e., multiplication, division, addition and subtraction; (ii) $C_{R2}$ is the computational cost related to the hyperbolic tangent (tanh) and inverse hyperbolic tangent (atanh) functions. The detailed calculations of $C_{R1}$ and $C_{R2}$ are presented in Section 3.2.

The number of encoded bits required for the successful recovery of the message, denoted as $N_F$, is a random variable which depends on the quality of the underlying transmission channel, i.e., Signal-to-Noise Ratio (SNR) values. In this thesis, the performances of Raptor codes in terms of $N_F$ are showed by the average values of $N_F$, denoted as $E(N_F)$'s, over a chosen range of channel SNRs.

### 1.2.2 Raptor Coding Rate Performance over Nakagami-$m$ Fading Channels

Raptor codes have been intensely investigated for transmissions over a wide range of channel models, including Additive White Gaussian Noise (AWGN) channels, Rayleigh fading channels and Rician fading channels [13–18]. The results showed consistently good coding rate performance of Raptor codes in all these channels. While Nakagami-$m$ models were considered to be more suitable for presenting a wide range of fading conditions in transmission channels than Rayleigh or Rician models [24–26], they have not yet been considered for transmissions with Raptor codes. In this

thesis, the Raptor coding rate is investigated for Nakagami-$m$ fading channels. The average Raptor coding rate, defined as $k/E(N_F)$ ($k$ is the size in bit of the message), is the figure of merit considered for this study. In Section 4.1, values of $k/E(N_F)$ are compared with the values of the average capacity of the underlying channel.

### 1.2.3 Raptor Coding Rate Performance over a Single-relay Wireless Relay Network

Various aspects of Raptor codes' performance in a WRN were investigated in [3, 27–30]. When employing Raptor codes, the source and relays usually continue the transmissions to the destination until the message is successfully recovered; hence, two attributes that are directly related to the Raptor coding rate performance are often chosen to be the figures of merit: (i) the overall transmission energy and (ii) the transmission time. X. Liu *et al.* in [28] studied and compared several cooperative protocols using Raptor codes in a core block of a WRN, i.e., a WRN with one source, one relay and one destination or a single-relay WRN. Their results showed that the Time-Division (TD) protocol has relatively good performances in terms of the average overall transmission energy. In this work, another protocol, named Phase-2 Simultaneous Transmission (PST) protocol, is proposed and aims to have a better performance than the TD protocol in terms of the transmission time. The PST protocol is a simplified version for a single-relay WRN based on the Asynchronous protocol proposed in [3] for multiple-relay WRN.

The performances of the TD protocol and the PST protocol are compared based on two figures of merit: the average energy expenditure $E(E_S)$ and average transmission time $E(T)$. These two attributes are directly related to the Raptor coding rate and will be described in Section 4.2.2.2.

## 1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2 provides the background material for the whole thesis. It includes the introduction of LT codes, which is the main component of Raptor codes, and then Raptor codes and how they both encode and decode a message. A particular Raptor

version and transmission channel models used for the investigation in Chapters 3 and 4 are also presented.

Chapter 3 investigates methods to reduce the Raptor decoding complexity, namely Previously Inherited Message Passing Algorithm (PIMPA), Late-Start Previously Inherited Message Passing Algorithm (LSPIMPA) and Thresholded Late-Start Previously Inherited Message Passing Algorithm (TLSPIMPA); the latter two are the contribution of this work. The idea of LSPIMPA is to begin the first decoding attempt late, i.e., after receiving a high number of bits, so that the number of decoding attempts is kept small and so is the decoding complexity. TLSPIMPA is the combination between LSPIMPA and the threshold technique in which only *reliable* received bits are input into the Raptor decoder. The first part of Chapter 3 presents these methods in detail and then provides the simulation results of their performance in terms of decoding complexity. The second part contributes to the investigation of another aspect of the threshold technique: estimating the improved quality of the channel when only *reliable* received bits are used for Raptor decoding.

Chapter 4 begins with the investigation of Raptor coding rate performance of point-to-point transmissions over Nakagami-$m$ fading channels. The simulation results are provided to show the consistently good Raptor coding rate performances over different fading conditions. Then, the second part of Chapter 4 considers Raptor coding rate performance over a single-relay WRN. Specifically, it compares the performance of two cooperative protocols using Raptor codes, the TD protocol and the PST protocol, in terms of the average transmission time $E(T)$ and the average energy expenditure $E(E_S)$.

Chapter 5 concludes the thesis and discusses some future work.

# Chapter 2

# Background

This chapter aims to introduce some preliminary knowledge for the understanding of the rest of the thesis. Specifically, the concepts of Luby-Transform (LT) codes are discussed together with its improved version Raptor codes. Also, the system model used throughout this work is presented.

The rest of this chapter is organized as follows. Section 2.1 introduces the encoding process of Luby-Transform codes, the first realization of fountain (i.e., rateless) codes. Also, two different LT decoding algorithms are presented that are applicable for both types of information output by the demodulator at the receiving end: (i) hard information (i.e., binary values of 0 or 1) and (ii) soft information (i.e., log-likelihood values). Section 2.2 summarizes some recent advances in the development of Raptor codes. Section 2.3 describes the transmission model used in this work.

## 2.1 Luby-Transform Codes

Luby-Transform (LT) codes, introduced by Luby in [31], are the first full realization of fountain codes. They are rateless codes in a sense that at the transmitter LT encoder generates as few or as many encoded bits as needed on-the-fly, making its coding rate not fixed *a priori*. For a given message, LT encoder continues to send a stream of encoded bits until enough information has been collected by the receiver for the successful recovery of the original message.

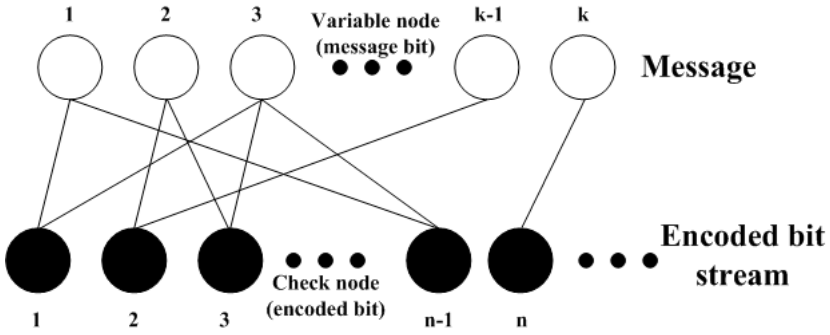



Figure 2.1: An example of factor graph representation of LT codes.

### 2.1.1 Luby-Transform Encoding Process

In the LT encoding process, a message is encoded into a potentially infinite number of bits, each of which is generated randomly and independently of one another. The LT encoder first splits the data stream into different $k$-bit messages, which are then encoded as follows. Given a distribution $D(x) = D_1x + D_2x^2 + D_3x^3 + ... + D_kx^k$ where $\sum_{i=1}^{k} D_i = 1$(called degree distribution), a number $d$ (called degree) is randomly generated. Here, $D_i$ is the probability that an arbitrary encoded bit has degree $d = i$.

Based on this, $d$ bits from the $k$ message bits are randomly chosen. Finally, these selected message bits are exclusive-OR-ed (XOR-ed), outputting one encoded bit. In this $d$-to-1 relation, the $d$ message bits are called the neighbours of the output encoded bit. Essentially, an LT code can be defined by $k$ and $D(x)$. The above-described process is repeated until enough encoded bits have been generated (i.e., the original message can be successfully recovered at the receiver side).

The encoding process can be best presented by the so-called factor graph [32] which describes the relationship between message bits and encoded bits. Specifically, a certain degree-$d$ encoded bit has $d$ edges connecting itself with its message bit neighbours. Figure 2.1 illustrates an example of such factor graph. As can be seen, there are two kinds of nodes in the factor graph (i) variable nodes to represent the message bits, and (ii) check nodes to represent the encoded bits. The index of the variable nodes, ranging from 1 to $k$, reflects the position of the message bits in the message; while the index of the check nodes gives the order in which the encoded bits are generated. Moreover, another two sets of indices are also defined:

1. $X_i$: the set of indices of variable nodes that are the neighbours of check node $i$. Assuming check node $i$ has the degree $d_i$, then $X_i = \{j_1, j_2, ..., j_{d_i}\}$ if and only

if $c_i = v_{j_1} \bigoplus v_{j_2} \bigoplus \cdots \bigoplus v_{j_{d_i}}$ where $c_i$ is the binary value of check node $i$, $v_{j_x}$ is the binary value of variable node $j_x$ and $\bigoplus$ presents the XOR operation.

For instance, in Figure 2.1, it is clear that $X_1 = \{1, 3\}$, $X_2 = \{2, k-1\}$,..., $X_n = \{k\}$,... as $c_1 = v_1 \bigoplus v_3$, $c_2 = v_2 \bigoplus v_{k-1}$ and $c_n = v_k$.

2. $Y_j$: the set of indices of check nodes that have variable node $j$ as a neighbour. Assuming variable node $j$ has, in total, $e_i$ check node neighbours, then $Y_j = \{i_1, i_2, ..., i_{e_i}\}$ if and only if $\forall k \in \{1, 2, ..., e_i\}, j \in X_{i_k}$.

   For instance, in Figure 2.1, $Y_1 = \{1, n-1\}$, $Y_2 = \{2, 3\}$,..., $Y_k = \{n\}$. As the variable node 1 is a neighbour of the check nodes 1 and $n-1$; the variable node 2 is a neighbour of the check nodes 2 and 3 and the variable node $k$ is a neighbour of the check node $n$.

### 2.1.2 Luby-Transform Decoding Process

After a message is encoded, it is sent over the transmission channel to the receiving end. The receiver collects the noise-corrupted encoded bits, called the received bits, and begins the LT decoding process to recover the original message. The factor graph is assumed to be perfectly known at the decoder side (see, e.g., [31]).

Based on the types of information received, LT decoders can perform either hard information or soft information decoding. This section presents two different LT decoding algorithms for a finite number of received bits (the noise-corrupted encode bits at the receiver). If received bits are received as either 1 or 0 or completely unknown, the LT decoding algorithm with hard information is applied; otherwise, if received bits are provided only with probabilities of their values, the LT decoding algorithm with soft information is applied.

#### 2.1.2.1 LT Decoding Algorithm with Hard Information

An LT decoding process is usually defined for a truncated LT code where the LT decoder only has access to a finite number of received bits. Based on the knowledge of the values of these received bits and the factor graph, the LT decoder uses an appropriate decoding algorithm to recover the values of the $k$ message bits. An LT decoding process is successful when all the values of $k$ bits of the message are recovered correctly.

After the LT decoder has collected $n$ received bits that are known as either 0 or 1(erasures are discarded), it begins the decoding process. First, it looks for check nodes (received bits) with degree one. For these, the only neighbouring variable node (message bit) can immediately be recovered. For example, in Figure 2.1, if the value of check node $n$ is known, the value of variable node $k$ is immediately recovered. These recovered variable nodes can be the neighbours of other degree-two check nodes. In turn, these recovered variable nodes help to recover other variable nodes that share the same neighbour check node with them. In general, if a check node with degree $d$ has $d-1$ of its neighbouring variable nodes recovered, its remaining neighbouring variable node can in turn be recovered. This process can be repeated several times and in the end, all $k$ variable nodes may be gradually recovered. However, in some cases, the decoding process with $n$ check nodes cannot recover all $k$ variable nodes, i.e., in cases that some variable nodes have no neighbour or that the decoding process cannot continue due to the fact that all the unprocessed check nodes have at least two unrecovered neighbouring variable nodes, then the LT decoder simply waits to collect more received bits and tries to decode again.

Systematically, the LT decoding algorithm with hard information proceeds in an iterative manner in the factor graph as follows. The LT decoder first creates a queue in which all the degree-one check nodes are listed. The LT decoding process begins by processing the first degree-one check nodes $j$ in the queue. The processing of check node $j$ (and other degree-one check nodes) is composed of three steps.

1. The only neighbour variable node $i$ of check node $j$ is recovered by assigning its value to the value of check node $j$.

2. XOR the recovered value of variable node $i$ to all the check nodes which have it as a neighbour.

3. Remove all the edges neighbouring to variable node $i$ in the factor graph.

After processing check node $j$, the LT decoder recovers the value of variable node $i$ and eventually reduces the degree of other neighbouring check nodes. As a result, some check nodes may become degree-one and will be put in the queue. The LT decoding process proceeds by consecutively processing other degree-one check nodes in the queue until all the variable nodes are recovered or the queue is empty. If the

queue is empty before all the variable nodes are recovered, the LT decoding process is not successful and the LT decoder collects more received bits and tries to decode again. If all the variable nodes are recovered, the LT decoding process is successful.

This LT decoding algorithm for hard information can be applied for transmissions over erasure channels, where received bits are either completely known or erasures. While the known received bits are used for decoding process, the erasures are all ignored. These ignored erasures do not affect the result of the LT decoding process as long as other received bits can form a factor graph helping the LT decoding process to proceed until all the message bits are recovered. LT codes for erasure channels are more suitable to some applications in the network layer as discussed in [2, 31].

#### 2.1.2.2 LT Decoding Algorithm with Soft Information

At the receiver side, instead of identifying each received bit as an absolute value like 1 or 0, the demodulator can assign each received bit with a reliability or a soft Log-likelihood Ratio (LLR) value, defined as:

$$L(c_i) = \ln\left(\frac{P\{c_i = 1\}}{P\{c_i = 0\}}\right) \tag{2.1}$$

where $L(c_i)$ is the reliability or soft LLR value of received bit $i$; $P\{c_i = 1\}$ and $P\{c_i = 0\}$ are the probabilities of $c_i = 1$ and $c_i = 0$, respectively. The calculations of these values are based on the knowledge of the transmission channel state at the receiver and are presented in Section 2.3.

Similar to the LT decoding algorithm with hard information, the LT decoding algorithm with soft information also uses the same factor graph for the decoding process; however the soft LLR values of check nodes complicates the recovery of variable nodes. With LLR values, the degree-one check nodes cannot recover its neighbouring variable nodes immediately; instead, the check nodes generate soft information messages and pass them to the neighbouring variable nodes, then the variable nodes also generate and pass back soft information messages. The decoding process happens in an iterative manner and at the end the values of variable nodes are recovered based on the soft information messages obtained at the final iteration. The passing back and forth of soft messages in the factor graph of this decoding algorithm lends it the name Message Passing Algorithm (MPA).

Specifically, in each decoding iteration, first all the check nodes generate and pass

the soft information messages $L(t_{i,j})$'s to their neighbouring variable nodes, where $L(t_{i,j})$ denotes the soft information message that the check node $i$ (received bit) generates and passes to its neighbouring variable node $j$ (message bit). This $L(t_{i,j})$ message represents the LLR value of the variable node $j$ calculated based on the knowledge of check node $i$. Then, all the variable nodes generate and pass the soft information messages $L(h_{j,i})$'s back to their neighbouring check nodes, where $L(h_{j,i})$ denotes the soft information message that variable node $j$ generates and passes to its neighbouring check node $i$. The $L(h_{j,i})$ message represents the LLR value of variable node $j$ and is calculated based on the knowledge of variable node $j$ itself. For each check node, the outgoing $L(t_{i,j})$ messages at the current iteration are updated based on the received $L(h_{j,i})$ messages from the previous iteration; while at the variable nodes, the process is reversed.

The details for the update rules for the $L(t_{i,j})$ and $L(h_{j,i})$ messages are as follows.

**Update rule for $L(t_{i,j})$ messages**

Before showing the update rule for $L(t_{i,j})$ messages, it is convenient first to present the following formula: If $x = x_1 \bigoplus x_2 \bigoplus \cdots \bigoplus x_n$ , where $x_1$,...,$x_n$ are independent binary random variables, then the LLR value of $x$ , denoted as $L(x)$, can be calculated as:

$$L(x) = 2 \times (-1)^{n-1} \times \operatorname{atanh}\left(\prod_{i=1}^{n} \tanh\left(\frac{L(x_i)}{2}\right)\right) \tag{2.2}$$

*Proof*

Check Formula 2.2 with $n = 2$

Consider the binary XOR operation $x = x_1 \bigoplus x_2$, then

$$\begin{aligned} P\{x=1\} &= P\left\{\Big((x_1=1)\bigcap(x_2=0)\Big)\bigcup\Big((x_1=0)\bigcap(x_2=1)\Big)\right\} \\ &= P\left\{\Big((x_1=1)\bigcap(x_2=0)\Big)\right\} + P\left\{\Big((x_1=0)\bigcap(x_2=1)\Big)\right\}. \end{aligned}$$

As $x_1$ and $x_2$ are independent with each other, then $P\{((x_1=1)\bigcap(x_2=0))\} = P\{x_1=1\} \times P\{x_2=0\}$ and $P\{((x_1=0)\bigcap(x_2=1))\} = P\{x_1=0\} \times P\{x_2=1\}$,

thus $P\{x=1\}$ becomes

$$\begin{aligned} P\{x=1\} &= P\{x_1=1\}P\{x_2=0\} + P\{x_1=0\}P\{x_2=1\} \\ &= P\{x_1=1\}\left(1-P\{x_2=1\}\right) + \left(1-P\{x_1=1\}\right)P\{x_2=1\} \\ &= P\{x_1=1\} + P\{x_2=1\} - 2P\{x_1=1\}P\{x_2=1\}. \end{aligned}$$

From the definition of soft LLR values in Equation 2.1, it is derived that $P\{x_i = 1\} = \frac{e^{L(x_i)}}{1+e^{L(x_i)}}$ . Replacing that into the above calculation provides

$$\begin{aligned} P\{x=1\} &= \frac{e^{L(x_1)}}{1+e^{L(x_1)}} + \frac{e^{L(x_2)}}{1+e^{L(x_2)}} - 2\frac{e^{L(x_1)+L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)} \\ &= \frac{e^{L(x_1)}+e^{L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)}, \end{aligned}$$

then

$$\begin{aligned} P\{x=0\} &= 1 - P\{x=1\} \\ &= 1 - \frac{e^{L(x_1)}+e^{L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)} \\ &= \frac{1+e^{L(x_1)+L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)}. \end{aligned}$$

Next, $L(x)$ is calculated as

$$\begin{aligned} L(x) &= \ln\left(\frac{P\{x=1\}}{P\{x=0\}}\right) \\ &= \ln\left(P\{x=1\}\right) - \ln\left(P\{x=0\}\right) \\ &= \ln\left(\frac{2e^{L(x_1)}+2e^{L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)}\right) \\ &\quad - \ln\left(\frac{2+2e^{L(x_1)+L(x_2)}}{\left(1+e^{L(x_1)}\right)\left(1+e^{L(x_2)}\right)}\right) \\ &= \ln\left(1 - \frac{e^{L(x_1)}-1}{e^{L(x_1)}+1}\times\frac{e^{L(x_2)}-1}{e^{L(x_2)}+1}\right) \\ &\quad - \ln\left(1 + \frac{e^{L(x_1)}-1}{e^{L(x_1)}+1}\times\frac{e^{L(x_2)}-1}{e^{L(x_2)}+1}\right). \end{aligned}$$

As $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$, $L(x)$ can be presented as follows:

$$\begin{aligned} L(x) &= \ln\left(1 - \tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right) \\ &\quad - \ln\left(1 + \tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right) \\ &= -2 \times \frac{1}{2}\left[\ln\left(1 + \tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right)\right. \\ &\quad \left. - \ln\left(1 - \tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right)\right]. \end{aligned}$$

As $\operatorname{atanh}(x) = \frac{1}{2}\left[\ln(1+x) - \ln(1-x)\right]$, $L(x)$ then finally becomes

$$\begin{aligned} L(x) &= -2 \times \operatorname{atanh}\left(\tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right) \\ &= -2 \times (-1)^{2-1} \times \operatorname{atanh}\left(\tanh\left(\frac{L(x_1)}{2}\right)\tanh\left(\frac{L(x_2)}{2}\right)\right). \end{aligned}$$

Since it agrees with Formula 2.2 for $n = 2$, Formula 2.2 for $n = 2$ is checked.

By induction, assuming Formula 2.2 is true for $n-1$, i.e., if $Y = x_1 \bigoplus x_2 \bigoplus \cdots \bigoplus x_{n-1}$ then

$$L(Y) = 2 \times (-1)^{n-2} \times \operatorname{atanh}\left(\prod_{i=1}^{n-1} \tanh\left(\frac{L(x_i)}{2}\right)\right) \tag{2.3}$$

Consider the XOR operation $x = x_1 \bigoplus x_2 \bigoplus \cdots \bigoplus x_{n-1} \bigoplus x_n = Y \bigoplus x_n$, using the Formula 2.2 for $n = 2$ provides

$$L(x) = 2 \times (-1)^{2-1} \times \operatorname{atanh}\left(\tanh\left(\frac{L(Y)}{2}\right)\tanh\left(\frac{L(x_n)}{2}\right)\right) \tag{2.4}$$

Substitute Equation 2.3 into Equation 2.4, the $L(x)$ has the form

$$\begin{aligned} L(x) &= 2 \times (-1) \times \operatorname{atanh}\left(\tanh\left((-1)^{n-2} \times \operatorname{atanh}\left(\prod_{i=1}^{n-1} \tanh\left(\frac{L(x_i)}{2}\right)\right)\right)\tanh\left(\frac{L(x_n)}{2}\right)\right) \\ &= 2 \times (-1)^{n-1} \times \operatorname{atanh}\left(\prod_{i=1}^{n} \tanh\left(\frac{L(x_i)}{2}\right)\right). \end{aligned}$$

This completes the proof.

Now consider the soft $L(t_{i,j})$ message that check node $i$ passes to its neighbouring variable nodes $j$. Assuming check node $i$ has degree $d_i$ and all of the indices of its

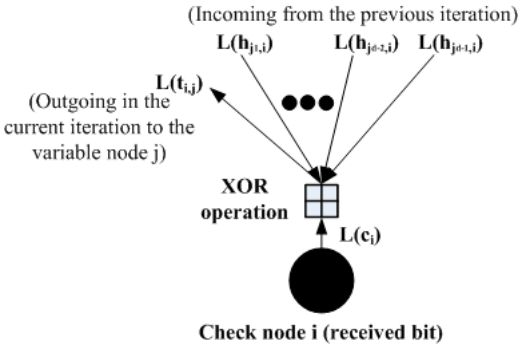



Figure 2.2: Update rule for the $L(t_{i,j})$ message.

neighbours are given by $X_i = \{j, j_1, j_2, ..., j_{d_i-1}\}$, then

$$\begin{aligned} c_i &= v_j \bigoplus v_{j_1} \bigoplus v_{j_2} \cdots \bigoplus v_{j_{d_i-1}} \quad \text{(as defined in Section 2.1.1)} \\ \Longrightarrow v_j &= c_i \bigoplus v_{j_1} \bigoplus v_{j_2} \cdots \bigoplus v_{j_{d_i-1}}. \end{aligned}$$

Since the $L(t_{i,j})$ message is the LLR value of variable node $j$, applying Formula 2.2, the $L(t_{i,j})$ message is calculated as

$$\begin{aligned} L(t_{i,j}) &= L(v_j) \\ &= 2 \times (-1)^{d_i-1} \times \operatorname{atanh}\left(\tanh\left(\frac{L(c_i)}{2}\right) \prod_{k \in X_i \text{ and } k \neq j} \tanh\left(\frac{L(h_{k,i})}{2}\right)\right) \end{aligned} \quad (2.5)$$

where $L(v_j)$ and $L(c_i)$ are the LLR values of variable node $j$ and check node $i$, respectively ; $L(h_{k,i})$ is the LLR value of variable node $k$ which is passed to check node $i$ from the previous iteration (since variable node $k$ is one of the neighbours of check node $i$). For the first iteration, all the $L(h_{k,i})$ messages are initialized as 0s.

Figure 2.2 illustrates the update rule for the $L(t_{i,j})$ message in which the value $L(t_{i,j})$ is obtained by "XOR-ing" all the LLR values of $L(h_{j,i})$ messages from the previous iteration and the $L(c_i)$ value provided by the demodulator. After the $L(t_{i,j})$ message is updated at check node $i$, it is then passed to variable node $j$ through the corresponding edge in the factor graph.

### Update rule for $L(h_{j,i})$ messages

Consider the $L(h_{j,i})$ message that variable node $j$ passes to its neighbouring check node $i$. Similar to the update rule for $L(t_{i,j})$ messages, the outgoing $L(h_{j,i})$ message

at the current iteration is updated based on the received $L(t_{k,j})$ messages from the previous iteration. However, the difference is that there is no XOR operation engaging all the received $L(t_{k,j})$ messages at variable node $j$. Recall that the $L(h_{j,i})$ message presents the LLR value for variable node $j$ itself, then one of the ways to decide the LLR value of variable node $j$ from all that $L(t_{k,j})$ messages is as follows.

Denote $P_k\{v_j = 1\}$ and $P_k\{v_j = 0\}$ as the probabilities of variable node $j$ equal to 1 and 0, respectively, according to check node $k$. These values are contained in the $L(t_{k,j})$ message passed from check node $k$ from the previous iteration. Recall that check node $k$ is just one of the neighbours of variable node $j$ and $Y_j$ lists all the neighbours of variable node $j$ (defined in 2.1.1). The probabilities of variable node $j$ equal to 1, i.e., $P\{v_j = 1\}$, and equal to 0 , i.e., $P\{v_j = 0\}$, can be calculated as: $P\{v_j = 1\} = \prod_{k \in Y_j \text{ and } k \neq i} P_k\{v_j = 1\}$ and $P\{v_j = 0\} = \prod_{k \in Y_j \text{ and } k \neq i} P_k\{v_j = 0\}$. These products exclude the probabilities with $k = i$ as the $L(h_{j,i})$ message is later sent to check node $i$ and thus it should not contain the information from the $L(t_{i,j})$ message. The LLR value of variable node $j$, i.e., $L(v_j)$, based on these probabilities is then calculated as

$$\begin{aligned} L(v_j) &= \ln\left(\frac{\prod_{k \in Y_j \text{ and } k \neq i} P_k\{v_j = 1\}}{\prod_{k \in Y_j \text{ and } k \neq i} P_k\{v_j = 0\}}\right) \\ &= \sum_{k \in Y_j \text{ and } k \neq i} \ln\left(\frac{P_k\{v_j = 1\}}{P_k\{v_j = 0\}}\right) \\ &= \sum_{k \in Y_j \text{ and } k \neq i} L(t_{k,j}). \end{aligned}$$

Hence, the update rule for the $L(h_{j,i})$ message is:

$$L(h_{j,i}) = L(v_j) = \sum_{k \in Y_j \text{ and } k \neq i} L(t_{k,j}) \tag{2.6}$$

Figure 2.3 illustrates the update rule for the $L(h_{j,i})$ message. As there is no XOR operation at variable node $j$, the summation of incoming $L(t_{i,j})$ messages from the previous iteration is assigned to the value of the outgoing $L(h_{j,i})$ message.

After the $L(h_{j,i})$ message is updated at variable node $j$, it is then passed to check node $i$ through the corresponding edge in the factor graph. Finally, at the final

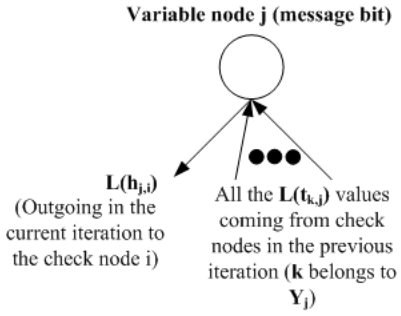



Figure 2.3: Update rule for the $L(h_{j,i})$ message.

iteration, the LLR value of each variable node $j$ is calculated as

$$L(v_j) = \sum_{k \in Y_j} L(t_{k,j}) \tag{2.7}$$

The hard value of each variable node (message bit) is then recovered based on the sign of $L(v_j)$: if $L(v_j) \geq 0$ then variable node $j$ is bit 1, otherwise it is bit 0.

Finally, the recovered $k$ message bits then are checked to see whether they are correctly recovered. This checking is employed by including some Cyclic Redundancy Check (CRC) bits in the $k$ message bits. If the CRC checking fails, the LT decoder waits to collect more received bits and then tries to decode again; otherwise, the LT decoding is considered to be successful. The probability that erroneous recovered messages pass the CRC checking is small and ignored in this work.

### 2.1.3 Soliton Distributions and The Coding Rate Performance of LT Codes

There is one question that is usually asked for any LT code: What is, on the average, the minimum number of received bits required for a successful LT decoding? This question is first addressed for the LT decoding algorithm with hard information. That result then can be served as a lower bound for the number of required received bits in the case of the LT decoder with soft information.

Concerned with this question, in [31], Luby designed the degree distribution $D(x)$ with the objective of keeping the average number of received bits required for a successful decoding as small as possible.

The result is the Ideal Soliton distribution

$$D(x) = \rho(1)x + \rho(2)x^2 + ... + \rho(k)x^k$$

where

$$\begin{cases} \rho(i) = \frac{1}{k} & i = 1 \\ \rho(i) = \frac{1}{i(i-1)} & i = 2, 3, ..., k. \end{cases}$$

This distribution is ideal in the sense of keeping the expected number of received bits needed to recover the message small; however, in practice, the decoding process often halts in the middle and fails to recover the message due to lack of degree-one check nodes. Addressing that problem, Luby proposed the Robust Soliton distribution, a modified version of the Ideal Soliton distribution, that works better in practice. The Robust Soliton distribution $D(x) = \mu(1)x + \mu(2)x^2 + ... + \mu(k)x^k$ is defined as follows. Let $R = c\ln(k/d)\sqrt{k}$ for some suitable constant $c > 0$. Define:

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

Add the Ideal Soliton distribution $\rho(.)$ to $\tau(.)$ and normalize to obtain $\mu(.)$:

$$\beta = \sum_{i=1}^{k} (\rho(i) + \tau(i))$$

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\beta} \quad i = 1, 2, ..., k.$$

Finally, Luby showed that, with the Robust Soliton distribution, the expected number of received bits required for a successful LT decoding was $K = k + \mathcal{O}\left(\sqrt{k}\ln^2(k/\sigma)\right)$ where $\sigma$ is the probability that the LT decoding process of $K$ received bits fails.

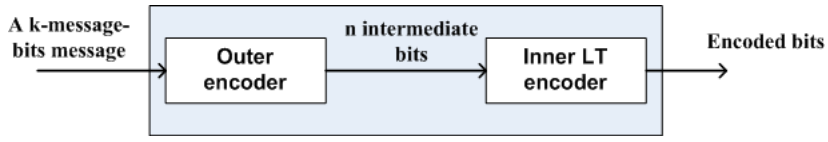
## 2.2 Raptor Codes

As discussed in the previous section, LT codes with the Robust Soliton distribution need $K = k + \mathcal{O}\left(\sqrt{k}\ln^2(k/\sigma)\right)$ received bits in average so that the message could be recovered successfully. However, $K$ is high with high value of $k$, e.g., with $k = 10^4$ and a probability of decoding failure $\sigma = 0.1$ then $K \sim 23255$, and that causes high encoding and decoding complexity. In [12], Shokrollahi proposed an extended form of LT code, called Raptor code, which has a lower required number of received bits on average for a successful decoding.

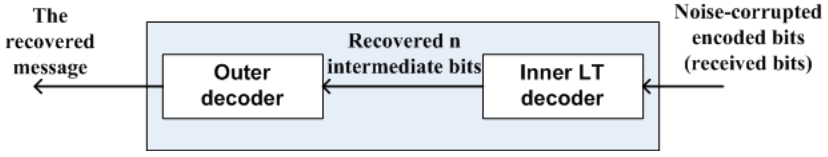
The Raptor codes' idea is to relax the condition of LT codes to recover all message bits. Instead, Shokrollahi uses an LT code version that recovers just a fraction of the

message bits, so that the required number of received bits is lower and thus the code has lower complexity. To maintain the reliability, the weakened inner LT code is then complemented with an outer code that can compensate for this loss. That is, in essence, the way a Raptor code is formed: an inner LT code combined with an outer code. Therefore, a Raptor code can be described by $(k, \mathbb{C}, D(x))$, where $k$ is the size of each message block at the input, $\mathbb{C}$ is the outer code and $D(x)$ is the degree distribution of the inner LT code. Figure 2.4a-b presents the concatenated structure of a Raptor encoder and a Raptor decoder.

As shown in Figure 2.4a-b, the concatenated structure of Raptor codes at both the receiver and the decoder implies two stages of the Raptor encoding/decoding process. At the Raptor encoder side, $k$ message bits first come through the outer code $\mathbb{C}$ encoder to produce $n$ intermediate bits. The encoded bits output by the Raptor encoder is then generated by the inner LT encoder from those $n$ intermediate bits. On the Raptor decoder side, inversely, the inner LT decoder first tries to decode $n$ intermediate bits. It needs to recover only just a fraction of these $n$ intermediate bits, leaving the outer decoder to correct the rest of the intermediate bits to get back the $k$ message bits.




(a) Raptor encoder




(b) Raptor decoder

Figure 2.4: Concatenated structure of Raptor encoder and decoder.

### 2.2.1 Raptor Encoding Process

Similar to LT codes, at the beginning of a Raptor encoding process, the data stream is split into messages, each with $k$ message bits. A $k$-bit message is encoded into $n$ intermediate bits by the outer encoder $\mathbb{C}$. These $n$ intermediate bits are

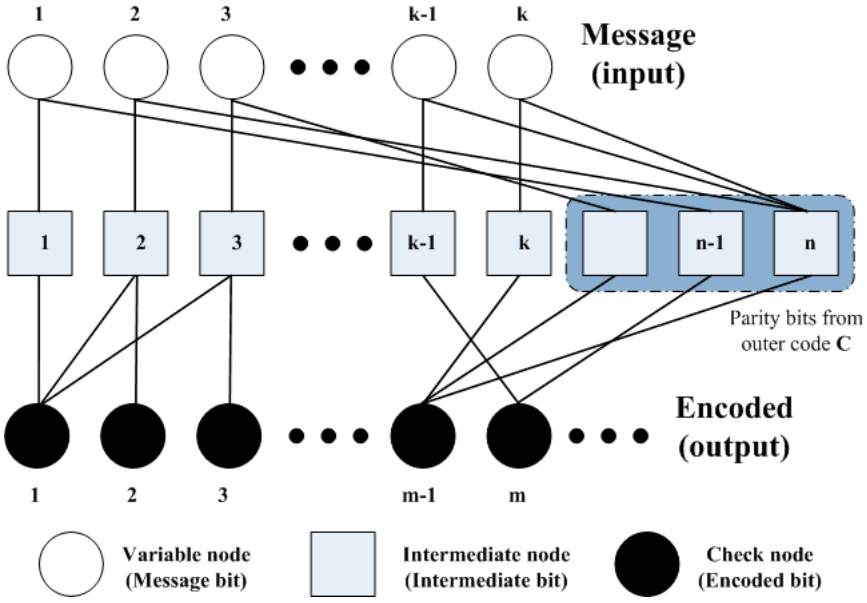



Figure 2.5: An example of factor graph representation of Raptor codes.

further encoded by the inner LT encoder into a stream of encoded bits through the same encoding process presented in Section 2.1.1.

After the encoding process, a Raptor code also has the factor graph representing the relationship between the $k$ message bits, $n$ intermediate bits and encoded bits. Figure 2.5 illustrates an example factor graph of a Raptor code. In this figure, the Raptor factor graph has variable nodes that represent message bits, intermediate nodes represent intermediate bits and check nodes represent encoded bits. Note that, the $k$ message bits are encoded to $n$ intermediate bits by a systematic outer code $\mathbb{C}$ in which $n$ intermediate bits are composed of $k$ unchanged message bits following by $n-k$ parity bits. This is not always the case as the outer code $\mathbb{C}$ can be employed with an non-systematic code, e.g., a non-systematic Low-Density Parity-Check (LDPC) code.

### 2.2.2 Raptor Decoding Process

After a message is encoded, the encoded bits are sent over the transmission channel to the receiver. The Raptor decoder collects a finite number $N_S$ of received bits (noise-corrupted encoded bits) and begins the decoding process. The Raptor factor graph is assumed to be perfectly known at the decoder. In the following, the decoding is described only with the LLR values of received bits output from the demodulator.

A Raptor decoding process usually proceeds with several decoding attempts. Basically, a Raptor decoding attempt consists of an inner LT soft decoding run and an outer soft decoding run. Each of these soft decoding runs has several iterations in which soft messages are passed back and forth in the factor graph as described in Section 2.1.2.2. Specifically, the Raptor decoder begins the first decoding attempt by applying the soft LT decoding algorithm to decode $N_S$ received bits and then output $n$ LLR values of intermediate bits to the outer decoder. The outer decoder then runs its soft decoding algorithm to recover the $k$ message bits. Similar to LT codes, the recovered messages of Raptor codes are also checked for errors with the embedded CRCs. If no error is found, the decoding attempt is successful and the whole Raptor decoding process for this particular message finishes; otherwise, the Raptor decoder waits until more encoded bits are received and begins a new decoding attempt.

### 2.2.3 Inner LT Distribution $D(x)$ and Coding Rate Performance of Raptor Codes

As described in the previous section, the number of received bits the Raptor decoder requires to recover the message is the same as that the inner LT code requires to recover $n$ intermediate bits. Hence, the Raptor coding rate performance is controlled by its inner LT code.

In [12], Shokrollahi showed the way of designing degree distributions for the weakened inner LT code, aiming to reduce the average number of received bits required for a successful Raptor decoding process. Applying his method for a fixed value of $k$ and a fraction $f$ of uncovered part of the intermediate bits, the degree distribution can be found. Table 2.1 reproduces Table I in [12] showing the Shokrollahi's designed degree distributions of the inner LT code for various values of $k$. In the table, $D_i$ is the probability that an encoded bit has degree $i$; $\varepsilon$ is the overhead, i.e., the average number of received bits required for a successful Raptor decoding is $K = (1+\varepsilon)k$; $a$ is the average degree of an encoded bit; the applied value of $f$ is 0.01 for all cases.

For example, from Table 2.1 with $k = 65536$, the distribution is found as:

$$\begin{aligned} D(x) = {} & 0.007969x + 0.49357x^2 + 0.16622x^3 + 0.072646x^4 + 0.082558x^5 \\ & + 0.056085x^8 + 0.037229x^9 + 0.05559x^{19} + 0.025023x^{65} \\ & + 0.003135x^{66}. \end{aligned} \tag{2.8}$$

| | k | | | |
|---|---|---|---|---|
| | 65536 | 80000 | 100000 | 120000 |
| $\mathbf{D_1}$ | 0.007969 | 0.007544 | 0.006495 | 0.004807 |
| $\mathbf{D_2}$ | 0.49357 | 0.49361 | 0.495044 | 0.496472 |
| $\mathbf{D_3}$ | 0.16622 | 0.166458 | 0.16801 | 0.166912 |
| $\mathbf{D_4}$ | 0.072646 | 0.071243 | 0.0679 | 0.073374 |
| $\mathbf{D_5}$ | 0.082558 | 0.084913 | 0.089209 | 0.082206 |
| $\mathbf{D_8}$ | 0.056058 | | 0.041731 | 0.057471 |
| $\mathbf{D_9}$ | 0.037229 | 0.043365 | 0.050162 | 0.035951 |
| $\mathbf{D_{18}}$ | | | | 0.001167 |
| $\mathbf{D_{19}}$ | 0.05559 | 0.045231 | 0.038837 | 0.054305 |
| $\mathbf{D_{20}}$ | | 0.010157 | 0.015537 | |
| $\mathbf{D_{65}}$ | 0.025023 | | | 0.018235 |
| $\mathbf{D_{66}}$ | 0.003135 | 0.010479 | 0.016298 | 0.009100 |
| $\mathbf{D_{67}}$ | | 0.017365 | 0.010777 | |
| $\varepsilon$ | 0.038 | 0.035 | 0.028 | 0.02 |
| **a** | 5.87 | 5.91 | 5.85 | 5.83 |

Table 2.1: Degree distributions of the inner LT code for several values of $k$.

The average number of received bits required for a successful decoding is then $K = (1+\varepsilon)k = (1+0.038) \times 65536 = 68026$. The corresponding Raptor coding rate is thus $k/K = 65536/68026 = 0.9634$, better than the usual coding rate of the LT code (mentioned at the beginning of Section 2.2) with $k = 10000$, $K = 23255$ and $k/K = 0.43$. With the degree distribution in Equation 2.8 and $K$ received bits, the inner LT decoder can recover a fraction $1 - f = 1 - 0.01 = 0.99$ of all intermediate bits.

Finally, in [12], the overall probability that the Raptor decoding fails with $K$ hard-decision received bits (and an appropriate outer code that can recover the message from $n$ intermediate bits with a fraction $f$ of erasures) is proved to be very small ($\sim 10^{-14}$).

### 2.2.4 Investigated Raptor Code Version

Raptor coding rate performances over transmission channels with soft information were considered in [13–18]. In [13], R. Planki and J. S. Yedidia investigated the performance of Raptor code and compared it with that of the LT code. They selected

the LT code with $k = 10000$, and for a fair comparison, they also selected a Raptor code with $k = 9500$ and with an outer LDPC code which can encode $k = 9500$ message bits into $n = 10000$ intermediate bits. The paper found that the Raptor code outperforms the LT code and has good coding rate performances on a wide variety of noisy channels. The results have been widely used by other papers in the selection and investigation of Raptor codes.

In [13], the Raptor code $(k, \mathbb{C}, D(x))$ has $k = 9500$, $D(x)$ is given in Equation 2.8 and the outer code $\mathbb{C}$ is an LDPC code which also can be presented using a factor graph. This LDPC code uses a left regular distribution (all variable nodes have degree 4) and right Poisson (check nodes chosen randomly with a uniform distribution) for its encoding process. As the LDPC encoding and decoding processes are similar to those of LT codes', readers are referred to [33] for details.

The Raptor code version mentioned above was proved to have good coding rate performances over different kinds of transmission channels [13–18]. In this thesis, this Raptor code version is also used for all the investigations in Chapters 3 and 4. Hereafter, whenever the "Raptor code" term is mentioned, it refers to this particular version.

## 2.3 Transmission Model

The transmission model considered this thesis is illustrated in Figure 2.6. Here, the transmission model takes the input from the Raptor encoder. The Raptor encoded bit stream is then Binary Phase Shift Keying (BPSK) modulated before it is transmitted through the transmission channel to the receiver. In the transmission channel, there is noise and other effects that corrupt or distort transmitted samples before they reach the receiver. At the receiver, the BPSK demodulator collects these distorted samples and, based on the channel state information, outputs their LLR values. These soft values are then output into the Raptor decoder. Section 2.3.1 describes the transmission channel effects on transmitted samples in a mathematical model. Section 2.3.2 presents how the BPSK demodulator calculates LLR values of received bits.

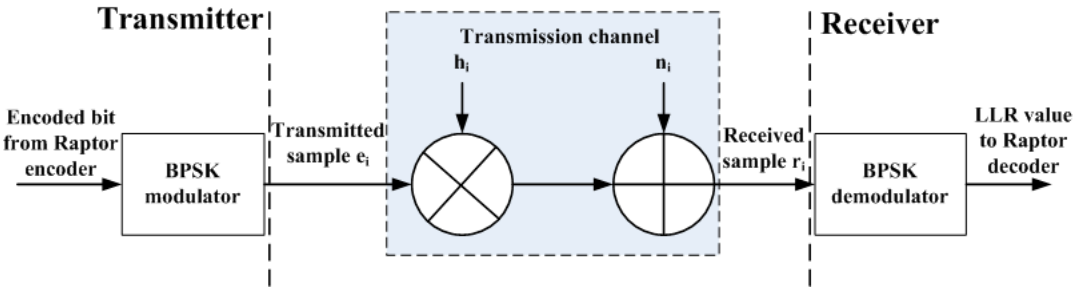



Figure 2.6: Transmission link.

### 2.3.1 The Transmission Channel Model

The transmission channel effects on transmitted samples are modeled mathematically as follows.

$$r_i = h_i e_i + n_i \tag{2.9}$$

where $e_i$ is the encoded bit $i$'s sample transmitted at time slot $i$ while $r_i$ is the corresponding noise-corrupted received sample of received bit $i$; $h_i$ is called the channel gain and presents the fading effect that usually happens in wireless transmissions; $n_i$ is an AWGN random variable that presents the random noise effecting the encoded bits signal.

While $n_i$ is generated independently for each time slot $i$, $h_i$ can be either kept the same or changed for different $i$. The number of time slot $i$ in which $h_i$ is kept unchanged is called the block fading size $B$. If $h_i = 1$ and is kept as a constant in a whole transmission, i.e., $B = \infty$, the transmission channel model becomes $c_i = e_i + n_i$ and it is called the AWGN channel (AWGNC). If $B = 1$, i.e., $h_i$ is generated independently for each time slot $i$, the transmission channel is called the fast fading channel.

In this thesis, $h_i$'s are randomly generated based on a distribution. That distribution lends its name to the corresponding transmission channel. For example, the Rayleigh fading channel has $h_i$ generated based on the Rayleigh distribution in Equation 2.10.

$$f(h|\sigma) = \frac{h}{\sigma^2} e^{-\frac{h^2}{2\sigma^2}} \tag{2.10}$$

with $h > 0$ and a parameter $\sigma > 0$.

Other distributions that will be mentioned in Section 4.1 are the Rician and Nakagami-$m$ distributions. The Rician distribution is given in Equation 2.11 and the Nakagami-$m$ distribution is given in Equation 2.12.

$$f\left(h|\sigma,\nu\right)=\frac{h}{\sigma^2}e^{-\frac{h^2+\nu^2}{2\sigma^2}}I_o\left(\frac{h\nu}{\sigma^2}\right) \tag{2.11}$$

with $I_o(z)$is the modified Bessel function of the first kind with order zero.

$$f\left(h|m\right)=\frac{2m^m h^{2m-1}}{\Gamma(m)}e^{-mh^2},\forall h\geq 0 \tag{2.12}$$

with $\Gamma(m)=\int_0^\infty x^{m-1}e^{-x}\,\mathrm{d}x$.

### 2.3.2 The Calculation of LLR Values

Similar to Equation 2.1, the LLR value of a received bit $i$ is defined as follows.

$$L(c_i)=\ln\left(\frac{P\{e_i=+1|r_i\}}{P\{e_i=-1|r_i\}}\right) \tag{2.13}$$

where $P\{e_i=+1|r_i\}$ the conditional probability that the transmitted sample $e_i=+1$, i.e., encoded bit $i$ is 1, for a given received sample $r_i$ at the receiver; similarly, $P\{e_i=-1|r_i\}$ is the conditional probability that the transmitted sample $e_i=-1$, i.e., encoded bit $i$ is 0, for a given received sample $r_i$ at the receiver. Recall that $c_i$ is the binary value of received bit (check node) $i$.

These two conditional probabilities are then may be expressed as

$$\begin{aligned}P\{e_i=+1|r_i\}&=\frac{f(r_i|e_i=+1)P\{e_i=+1\}}{f(r_i)}\\P\{e_i=-1|r_i\}&=\frac{f(r_i|e_i=-1)P\{e_i=-1\}}{f(r_i)}\end{aligned}$$

where $f(r_i)$ is the probability density function (pdf) of $r_i$; $f(r_i|e_i=+1)$ and $f(r_i|e_i=-1)$ are the conditional pdf's of $r_i$ given $e_i=+1$ and $e_i=-1$, respectively.

Assuming the message has an equal number of bit 1 and bit 0, then $P\{e_i=+1\}=P\{e_i=-1\}=1/2$. Therefore,

$$\begin{aligned}L(c_i)&=\ln\left(\frac{P\{e_i=+1|r_i\}}{P\{e_i=-1|r_i\}}\right)\\&=\ln\left(\frac{f(r_i|e_i=+1)}{f(r_i|e_i=-1)}\right).\end{aligned}$$

If $n_i$ is generated based on a normal distribution with variance $\sigma^2$, $f(r_i|e_i = +1)$ and $f(r_i|e_i = -1)$ are:

$$f(r_i|e_i = +1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i - h_i)^2}{2\sigma^2}}$$
$$f(r_i|e_i = -1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i + h_i)^2}{2\sigma^2}}.$$

Finally,

$$\begin{aligned} L(c_i) &= \ln\left(\frac{f(r_i|e_i = +1)}{f(r_i|e_i = -1)}\right) \\ &= \ln\left(e^{-\frac{(r_i - h_i)^2}{2\sigma^2}}\right) - \ln\left(e^{-\frac{(r_i + h_i)^2}{2\sigma^2}}\right) \\ &= \frac{-(r_i - h_i)^2}{2\sigma^2} - \frac{-(r_i + h_i)^2}{2\sigma^2} \\ &= \frac{2h_i r_i}{\sigma^2}. \end{aligned} \tag{2.14}$$

In all the investigations of Chapters 3 and 4, Equation 2.14 is used to calculate the LLR values of all received bits for the Raptor decoder. Note that, in this thesis, the receiver is assumed to always know the SNR of the channel. Techniques to estimate channel SNR at the receiver can be found in [34, 35].

# Chapter 3

# Methods of Reducing Raptor Decoding Complexity

Section 2.2.2 described the overall Raptor decoding process at the receiver. It is observed that the Raptor decoding process is costly in terms of computation since it requires multiple decoding attempts to recover the message, and that each decoding attempt has multiple iterations. Also, the complexity of each iteration is dependent on the number of received bits in use which can be quite high.

There are several strategies to reduce the Raptor decoding complexity: (i) By choosing appropriately the number of received bits to be used in the initial decoding attempt, the number of decoding attempts can be reduced. This is called the late-start strategy. (ii) Reusing of the decoding results from the previous attempt can reduce the number of iterations required in each attempt. This is called the reuse-of-previous-results strategy. (iii) Discarding of uncertain received bits so that only "more certain" or "reliable" received bits are used can leads to fewer received bits being used in each iteration. This is called the threshold technique. The reuse-of-previous-results strategy and the threshold technique were already investigated in [7] and [28], respectively, while the late-start strategy is the contribution of this work.

Note that, discarded uncertain received bits in the threshold technique are similar to erasures in the erasure channel. However, for erasure channels, the received bit is either known for its hard value (1 or 0) or unknown (erasure). On the other hand, with the threshold technique, the output is the soft LLR value of the received bit.

These three strategies can be combined. Table 3.1 presents combinations that will

| Algorithm | Late-start | Reuse-of-previous-results | Threshold |
|---|---|---|---|
| PIMPA | | X | |
| LSPIMPA | X | X | |
| TLSPIMPA | X | X | X |

Table 3.1: Three algorithms for reducing Raptor decoding complexity.

be investigated in this chapter. In the table, PIMPA stands for Previously Inherited Message Passing Algorithm and is employed with only the reuse-of-previous-results strategy; LSPIMPA stands for Late Start PIMPA and is employed with both the late-start and the reuse-of-previous-results strategies; TLSPIMPA stands for Threshold LSPIMPA and is employed with all three strategies: the late-start, the reuse-of-previous-results and the threshold strategies. As the late-start strategy is the contribution of this work, the main focus of this chapter is on the complexities of LSPIMPA and TLSPIMPA. Note that the reuse-of-previous-results strategy or PIMPA is chosen to be the basics for LSPIMPA and TLSPIMPA here as it was shown in [7] to be able to reduce the decoding complexity without any sacrifice in the number of received bits for a successful decoding.

In Section 3.3, the decoding complexities of PIMPA, LSPIMPA and TLSPIMPA are compared with that of the standard Raptor decoding process which uses the Message Passing Algorithm (MPA) (presented in Section 2.1.2.2) for both the inner LT decoding and the outer LDPC decoding.

The threshold technique of providing the Raptor decoder with more reliable bits is similar to simulating a better underlying channel. Section 3.4 investigates this observation and developes methods to estimate the improved underlying channel.

This chapter is organized as follows. Section 3.1 presents the detailed operation of the three algorithms. Section 3.2 presents the detailed calculation of the Raptor decoding complexity in terms of the number of basic operations (i.e., multiplication, division, subtraction and addition) $C_{R1}$ and the number of hyperbolic functions (i.e., tanh or atanh) $C_{R2}$. Simulation results are shown in Section 3.3 to demonstrate how PIMPA, LSPIMPA, and TLSPIMPA reduce the decoding complexity in the comparison with MPA. Section 3.4 investigates the effect of applying the threshold technique, i.e., the virtual improved channel quality. Finally, a summary of the

chapter is given in Section 3.5.

## 3.1 Methods of Reducing Raptor Decoding Complexity

In this section, the detailed operations of PIMPA, LSPIMPA and TLSPIMPA are presented. However, first, some of the Raptor decoding parameters that affect the decoding complexity are reviewed.

### 3.1.1 Raptor Decoding Parameters

Figure 3.1 presents the overall Raptor decoding process and highlights some important parameters that will be mentioned later. The Raptor decoding process starts the first decoding attempt when $N_S$ received bits are collected at the receiver. The inner LT decoder first uses the MPA (for soft information) with $T_1$ iterations to recover the intermediate bits and output the LLR values of intermediate bits into the outer LDPC decoder. The outer decoder then also uses the MPA with $T_2$ iterations to recover the message. If the recovered message fails the CRC check, the decoder will wait to receive more $I$ received bits before having another decoding attempt. Finally, recall that $N_F$ is the number of received bits for a successful recovery of the message, then the number of the decoding attempts $f_{da}$ is $f_{da} = \frac{N_F - N_S}{I}$.

### 3.1.2 Raptor Decoding Algorithms for Reducing Complexity

PIMPA tries to reduce the complexity by reusing the results from the previous decoding attempt. Specifically, in the first iteration of the first decoding attempt of PIMPA, all the soft messages that each intermediate node (intermediate bit) passes to its neighbouring check nodes (received bits) are initialized as 0's. This is the same with the MPA presented in Section 2.1.2.2. In the first iterations of subsequent decoding attempts, MPA also initializes these soft messages as 0's. However, unlike MPA, in the first iteration of the $i^{th}$ decoding attempt ($i > 1$) of PIMPA, the soft messages that each intermediate node $j$ ($1 \leq j \leq n$) passes to its neighbouring check nodes are set to its LLR value obtained from the last iteration of the previous decoding attempt. K. Hu *et al.* in [7] first proposed PIMPA (called Algorithm B),

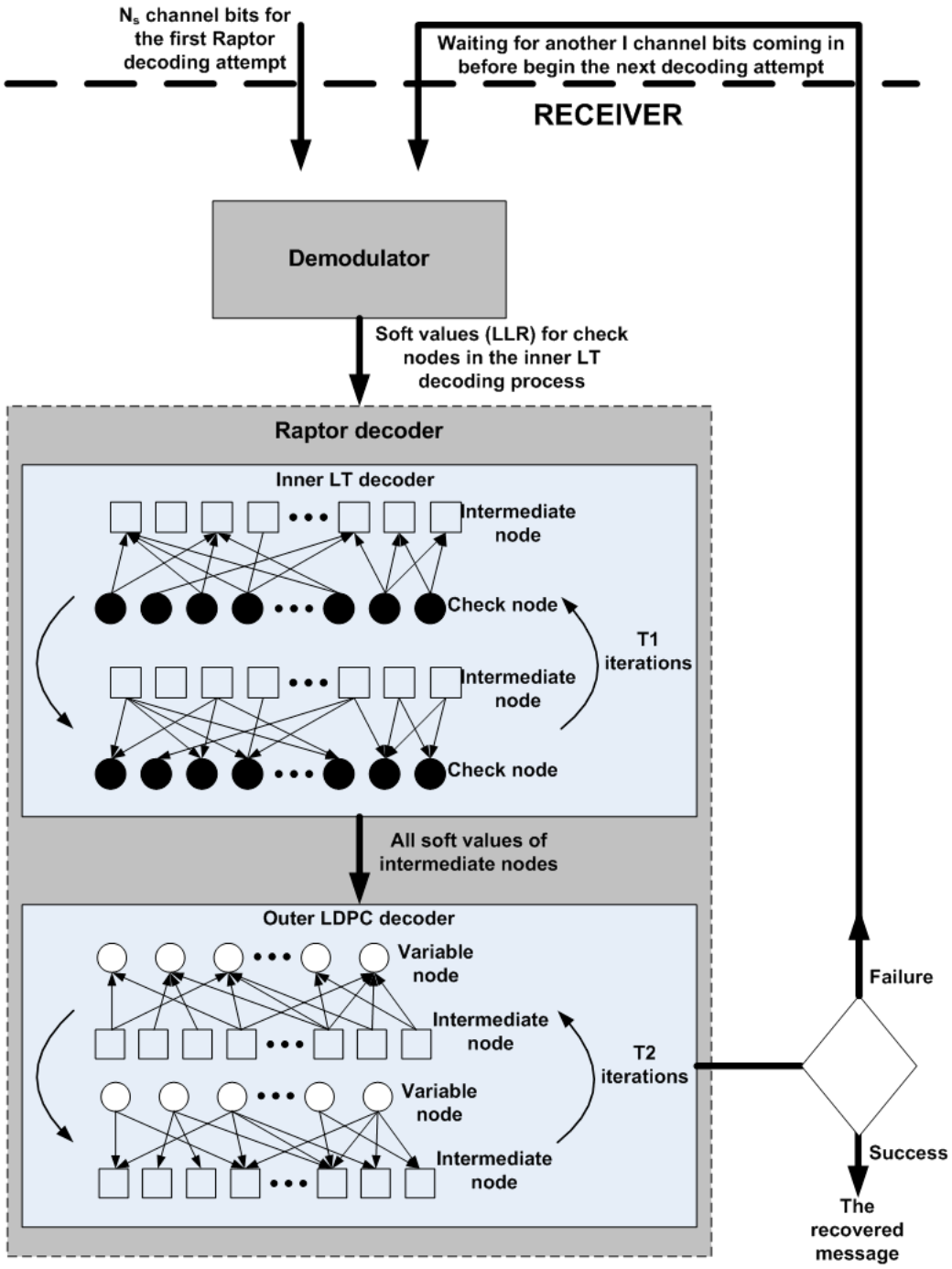



Figure 3.1: Overall Raptor decoding process.

and then showed that PIMPA not only outperforms MPA (called Algorithm A) in terms of decoding complexity, but also requires fewer received bits for a successful Raptor decoding process.

LSPIMPA is proposed to further reduce the decoding complexity by combining PIMPA with the late-start strategy. Specifically, LSPIMPA uses a larger $N_S$ so that it is more likely that the decoding process will be successful in the first decoding attempt. In this way, the number of decoding attempts $f_{da}$ is kept small and thus the average Raptor decoding complexity is reduced. However, this method clearly costs the average Raptor coding rate performance as it begins the decoding with a higher number of received bits in most of the transmissions. Investigation of the values of $N_S$ for LSPIMPA under various channel and SNR conditions will be presented in Section 3.3.

TLSPIMPA is an extension of LSPIMPA. It aims to further reduce the complexity by combining LSPIMPA with the threshold technique. Specifically, assuming that the threshold is $S \geq 0$, all the received bits that have their *absolute* LLR values greater than $S$ are considered as reliable and will be used for the decoding process. Note that, for TLSPIMPA, the mechanism of choosing the number of received bits for the first attempt inherited from LSPIMPA is now based on the number of *reliable* received bits.

## 3.2 Overall Complexity of a Raptor Decoding Process

As described in Section 2.2.2, a Raptor decoding process usually has many attempts and each attempt consists of an inner LT decoding and an outer decoding. Section 3.2.1 presents the calculation for the complexity of an inner LT decoding while Section 3.2.2 presents that for an outer LDPC decoding. Finally, Section 3.2.3 presents the overall Raptor decoding complexity composed of multiple decoding attempts.

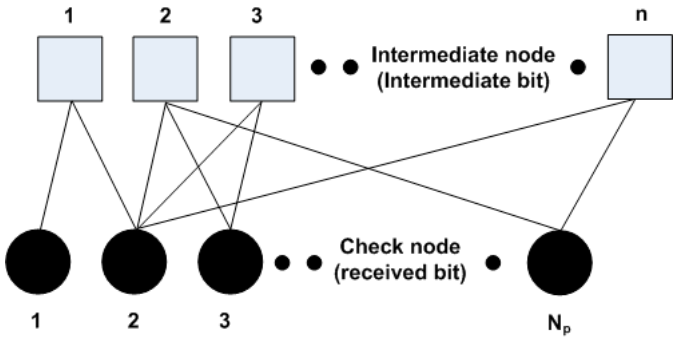



Figure 3.2: Factor graph of the inner LT code with $N_p$ check nodes and $n$ intermediate nodes.

### 3.2.1 Complexity of an Inner LT Decoding Attempt

This section presents the calculation of the complexity of the $p^{th}$ ($p \geq 1$) inner LT decoding attempt $C_{LT}^p$. Before proceeding to the details, it is convenient to present here, in Figure 3.2, the factor graph of the inner LT code where $N_p$ is the number of received bits (check nodes) used in the $p^{th}$ Raptor decoding attempt.

As discussed in Chapter 2, an inner LT decoding with soft information requires many iterations. In each iteration, the check nodes calculate and send soft messages to the intermediate nodes, then the intermediate nodes calculate and send back soft messages to the check nodes. The complexity of the calculation done at the check nodes in each iteration is denoted as $C_c^p$ while the complexity of the calculation at the intermediate nodes in each iteration is denoted $C_i^p$. Note that these complexities are the same for each iteration in the same attempt. Hence, if there are $T_1^p$ iterations in the $p^{th}$ attempt, then the total complexity of the $p^{th}$ attempt is

$$C_{LT}^p = (C_c^p + C_i^p) \times T_1^p.$$

(Note that this is the complexity of the inner LT decoding attempt only. The outer LDPC attempt complexity will be given in Section 3.2.2)

Detailed calculations of $C_c^p$ and $C_i^p$ are as follows.

#### 3.2.1.1 Derivation of $C_c^p$

As discussed in Section 2.1.2.2, in each iteration of the LT soft decoding process, each check node calculates and sends soft information to each of its neighbouring

intermediate nodes. Specifically, it sends an estimate of the LLR value of that intermediate node, denoted as $L(t_{i,j})$ where $i$ is the index of the check node and $j$ is the index of the check node's neighbouring intermediate nodes. Assuming check node $i$ has $d_i$ neighbouring intermediate nodes, the formula for $L(t_{i,j})$ is then given by Equation 2.5 which is reproduced below.

$$L(t_{i,j}) = 2 \times (-1)^{d_i - 1} \times \operatorname{atanh}\left(\tanh\left(\frac{L(c_i)}{2}\right) \prod_{k \in X_i \text{ and } k \neq j} \tanh\left(\frac{L(h_{k,i})}{2}\right)\right) \quad (2.5)$$

where $X_i$ is the set of all the indices of the neighbouring intermediate nodes of the check node $i$ as mentioned in Section 2.1.1.

Some observations for the calculation of Equation 2.5 are as follows.

1. $\tanh(L(c_i)/2)$ depends only on the received bit $i$ and is the same for each iteration and indeed for every decoding attempt. Thus, it should be calculated at the time received bit $i$ is received. Hence, the total complexity for these values is $N_F \times c_{tanh}$, where $c_{tanh}$ is the computational cost of the hyperbolic tangent (tanh) function ($N_F$ is the number of received bits required for a successful decoding as mentioned in Section 3.1.1). As this complexity is only a small part of the overall Raptor decoding complexity (approximately 0.1%), it will be neglected in the complexity calculation here.

2. Each $L(h_{k,i})$ is associated with an edge joining a check node to an intermediate node and its value changes in each iteration. Thus, the $\tanh(L(h_{k,i})/2)$ must be calculated in each iteration, but for all calculations in that iteration the $\tanh(L(h_{k,i})/2)$ is the same. Thus, these $\tanh(L(h_{k,i})/2)$'s are calculated once in each iteration. Now there is one $L(h_{k,i})$ for each edge joining a check node to an intermediate node; the total number of edge is $\sum_{i=1}^{N_p} d_i = N_e$, and so there are $N_e \times \tanh(L(h_{k,i})/2)$'s to be calculated in each iteration.

3. For each calculation of Equation 2.5, the argument of the atanh function must be determined. For each check node $i$, these arguments are similar. Specifically, all have $d_i - 1$ of the $d_i$ $\tanh(L(h_{k,i})/2)$'s associated with the check node $i$ plus the $\tanh(L(c_i)/2)$. All these arguments of the check node $i$ can be done with $d_i$ multiplications and $d_i - 1$ divisions. Then, for all the check nodes in each iteration, this is $N_e$ multiplications and $N_e - N_p$ divisions.

For example, in Figure 3.2, check node 2 has four neighbouring intermediate nodes 1, 2, 3 and $n$, i.e., $d_i = 4$. It must calculate Equation 2.5 $d_i = 4$ times. It first calculates the four values of $\tanh(L(t_{2,1})/2)$, $\tanh(L(t_{2,2})/2)$, $\tanh(L(t_{2,3})/2)$ and $\tanh(L(t_{2,n})/2)$. Defining $A_{j,i}$ as the argument of the atanh function in Equation 2.5, then the four arguments calculated by check node 2 are $A_{2,1}$, $A_{2,2}$, $A_{2,3}$ and $A_{2,n}$. They are calculated first by four multiplications of

$$\begin{aligned} M_1 &= \tanh\left(\frac{L(c_i)}{2}\right) \times \tanh\left(\frac{L(t_{2,1})}{2}\right), \\ M_2 &= M_1 \times \tanh\left(\frac{L(t_{2,2})}{2}\right), \\ M_3 &= M_2 \times \tanh\left(\frac{L(t_{2,3})}{2}\right), \\ M_4 &= M_3 \times \tanh\left(\frac{L(t_{2,n})}{2}\right). \end{aligned}$$

Note that $A_{2,n} = M_3$. The other $A_{j,i}$'s are calculated with three divisions, namely

$$\begin{aligned} A_{2,1} &= M_4/\tanh\left(\frac{L(t_{2,1})}{2}\right), \\ A_{2,2} &= M_4/\tanh\left(\frac{L(t_{2,2})}{2}\right), \\ A_{2,3} &= M_4/\tanh\left(\frac{L(t_{2,3})}{2}\right). \end{aligned}$$

4. To finish the calculation of Equation 2.5, an atanh function must be done (note that the multiplications by $2 \times (-1)^{d_i - 1}$ are taken as trivial).

Now, in each iteration, Equation 2.5 must be calculated for each neighbouring intermediate node of each check node, i.e., for each edge in Figure 3.2 or $N_e$ times. Hence, the complexity to calculate all messages from the check nodes to the intermediate nodes per iteration at the $p^{th}$ decoding attempt $C_c^p$ is

$$\begin{aligned} C_c^p &= N_e c_{tanh} + N_e c_{mul} + (N_e - N_p) c_{div} + N_e c_{atanh} \\ &= N_e(c_{tanh} + c_{mul} + c_{div} + c_{atanh}) - N_p c_{div} \end{aligned} \tag{3.1}$$

where $c_{div}$, $c_{mul}$, $c_{tanh}$ and $c_{atanh}$ are the computational cost of the division, the multiplication, the hyperbolic tangent (tanh) and the inverse hyperbolic tangent (atanh) functions, respectively.

#### 3.2.1.2 Derivation of $C_i^p$

The soft information that an intermediate node calculates and sends to its neighbouring check nodes is the LLR value of the intermediate node itself denoted as $L(h_{j,i})$. Here, $j$ is the index of the intermediate node and $i$ is the index of the neighbouring check nodes of intermediate node $j$. $L(h_{j,i})$ is given by Equation 2.6 which is reproduced below.

$$L(h_{j,i}) = \sum_{k \in Y_j \text{ and } k \neq i} L(t_{k,j}) \tag{2.6}$$

where $Y_j$ is the set of all the indices of the neighbouring check nodes of the intermediate node $j$ as mentioned in Section 2.1.1.

Define $e_j$ as the number of the neighbouring check nodes of intermediate node $j$. Equation 2.6 for intermediate node $j$ is then calculated $e_j$ times in each iteration and the calculation can be done with $e_j - 1$ additions and $e_j - 1$ subtractions. Then, for all the intermediate nodes in each iteration, this is $\sum_{j=1}^{n} (e_j - 1) = N_e - n$ additions and $N_e - n$ subtractions (note that $\sum_{j=1}^{n} e_j = \sum_{i=1}^{N_p} d_i = N_e$ and they indicate the number of edges of the inner LT factor graph).

For example, in Figure 3.2, intermediate node 2 has three neighbouring check nodes 2, 3 and $N_p$, i.e., $e_2 = 3$. The intermediate node 2 calculates $e_2 = 3$ values of $L(h_{2,2})$, $L(h_{2,3})$ and $L(h_{2,N_p})$. These three values can be calculated first by $e_2 - 1 = 2$ additions of

$$B_1 = L(t_{2,2}) + L(t_{3,2}),$$
$$B_2 = B_1 + L(t_{N_p,2})$$

then $e_2 - 1 = 2$ subtractions of

$$L(h_{2,2}) = B_2 - L(t_{2,2}),$$
$$L(h_{2,3}) = B_2 - L(t_{3,2})$$

while $L(h_{2,N_p}) = B_2$.

Hence, the complexity to calculate all the messages from the intermediate nodes to the check nodes per iteration at the $p^{th}$ decoding attempt $C_i^p$ is

$$C_i^p = (N_e - n)(c_{add} + c_{sub}) \tag{3.2}$$

where $c_{add}$ and $c_{sub}$ are the computational costs of the addition and subtraction, respectively.

#### 3.2.1.3 Derivation of $C_{LT}^p$

Combining Equations 3.1 and 3.2, the total complexity of the $p^{th}$ inner LT decoding attempt $C_{LT}^p$ with $T_1^p$ iterations is

$$\begin{aligned} C_{LT}^p &= T_1^p \left(C_c^p + C_i^p\right) \\ &= T_1^p (N_e(c_{tanh} + c_{atanh} + c_{mul} + c_{div} + c_{add} + c_{sub}) \\ &\qquad - n \times c_{add} - n \times c_{sub} - N_p \times c_{div}) \end{aligned} \tag{3.3}$$

Equation 3.3 presents the complexity of the $p^{th}$ inner LT decoding attempt in terms of the computational complexity of basic operations, i.e., multiplication, division, addition and subtraction, and two hyperbolic functions, i.e., tanh and atanh. From the implementation point of view, the tanh and atanh functions are rather complex as they involves operations such as divisions, additions, exponentials and logarithms (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and $\text{atanh}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$). The piecewise linear function approximation and the quantisation table approximation were proposed to reduce the computational complexity of these two functions [10, 36]. Even though these approximation methods reduced the tanh and atanh complexity by using off-line computations, it is still hard to compare the complexity of these two functions and that of the basic operations. Therefore, in this work, the decoding complexity is considered in two separate terms: (i) the complexity in terms of the number of tanh or atanh functions and (ii) the complexity in terms of the number of addition, subtraction, multiplication and division. Also, there are two other approximations for the complexity calculation: (i) the number of edges in the inner LT factor graph $N_e$ is approximated by the product between the number of check nodes (received bits) $N_p$ and the average degree of each check node $a = 5.87$ (Section 2.2.3); (ii) the computational complexities of multiplication and division are assumed to be equal

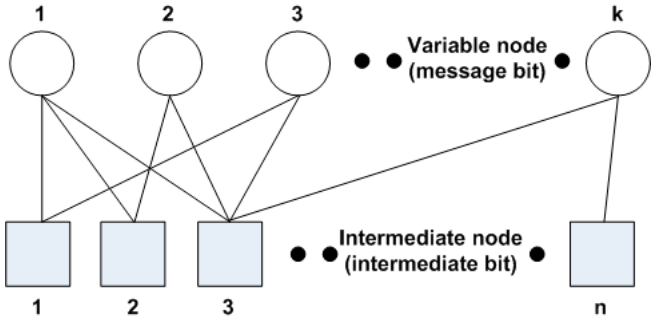



Figure 3.3: Factor graph for the outer LDPC code with $n$ intermediate nodes and $k$ variable nodes.

to those of addition and subtraction (as in case where a digital signal processor is used [10]). Denote $c$ as the computational complexity for the basic operations, the complexity of the $p^{th}$ inner LT decoding attempt $C_{LT}^{p}$ in Equation 3.3 is represented by two separate terms as follows.

$$C_{LT1}^{p} = T_1^p N_p a(c_{tanh} + c_{atanh}) \tag{3.4}$$

$$C_{LT2}^{p} = T_1^p N_p \left(4a - 2\frac{n}{N_p} - 1\right) c \tag{3.5}$$

where $C_{LT1}^{p}$ presents the complexity related to the tanh and atanh functions while $C_{LT2}^{p}$ presents the complexity related to basic operations.

### 3.2.2 Complexity of an Outer LDPC Decoding Attempt

Similar to the inner LT code, the outer LDPC code also has the factor graph which is shown in Figure 3.3.

As the outer LDPC code also uses MPA for the decoding process, the complexity of its $p^{th}$ attempt is derived in a similar way to Equation 3.3 and also has two separated items $C_{LDPC1}^{p}$ and $C_{LDPC2}^{p}$ as follows. Note that, here, $N_e'$ is the number of edges in the outer LDPC factor graph and $T_2^p$ is the number of iterations of the $p^{th}$ outer LDPC decoding attempt.

$$C_{LDPC1}^{p} = T_2^p N_e'(c_{tanh} + c_{atanh}) \tag{3.6}$$

$$C_{LDPC2}^{p} = T_2^p (4N_e' - 2k - n)c \tag{3.7}$$

### 3.2.3 Complexity of a Raptor Decoding Process

Recall that $f_{da}$ is the number of decoding attempts that the Raptor decoder has tried before the message is successfully recovered. Using Equations 3.4, 3.5, 3.6 and 3.7, the total Raptor decoding complexity can be represented by $C_{R1}$ (in terms of tanh and atanh functions) and $C_{R2}$ (in terms of basic operations) as follows.

$$\begin{aligned}
C_{R1} &= \sum_{p=1}^{f_{da}} (C_{LT1}^{p} + C_{LDPC1}^{p}) \\
&= \sum_{p=1}^{f_{da}} T_1^p N_p a (c_{tanh} + c_{atanh}) + \sum_{p=1}^{f_{da}} T_2^p N_e' (c_{tanh} + c_{atanh}) \\
&= (c_{tanh} + c_{atanh}) \left( \sum_{p=1}^{f_{da}} T_1^p N_p a + \sum_{p=1}^{f_{da}} T_2^p N_e' \right) \qquad (3.8) \\
C_{R2} &= \sum_{p=1}^{f_{da}} (C_{LT2}^{p} + C_{LDPC2}^{p}) \\
&= \sum_{p=1}^{f_{da}} T_1^p N_p \left( 4a - 2\frac{n}{N_p} - 1 \right) c + \sum_{p=1}^{f_{da}} T_2^p (4N_e' - 2k - n) c \\
&= c \left( \sum_{p=1}^{f_{da}} T_1^p N_p \left( 4a - 2\frac{n}{N_p} - 1 \right) + \sum_{p=1}^{f_{da}} T_2^p (4N_e' - 2k - n) \right) \qquad (3.9)
\end{aligned}$$

Table 3.2 summarizes the values of the parameters used in the Raptor decoding process in this work.

| **Parameter** | **Value** |
|---|---|
| Number of iterations for an outer LDPC decoding attempt | $T_2^p = 75$ |
| Number of edges of the outer LDPC factor graph | $N_e' = 38000$ |
| Number of intermediate nodes of the outer LDPC factor graph | $n = 10000$ |
| Number of variable nodes of the outer LDPC factor graph | $k = 9500$ |

Table 3.2: Values of parameters used in the Raptor decoding process in this work.

## 3.3 The Complexities of Raptor Decoding Algorithms

In this section, the complexity of MPA, PIMPA, LSPIMPA and TLSPIMPA over an AWGN channel is examined using Equations 3.8 and 3.9 and the parameter values in Table 3.2.

Section 3.3.1 presents the parameter settings for each algorithm, including $I$ - the interval between two consecutive decoding attempts in received bits, $T_1^p$ - the number of inner LT iterations for the $p^{th}$ attempt and $N_S$ - the number of received bits for the first decoding attempt. Then, Section 3.3.2 shows the numerical results for the complexity of each algorithm.

### 3.3.1 Parameter Settings

#### 3.3.1.1 Choices of $I$ and $T_1$

| **Parameter** | **MPA** | **PIMPA** | **LSPIMPA or TLSPIMPA** |
|---|---|---|---|
| $I$ - The interval between two consecutive decoding attempts | If the $p^{th}$ ($p \geq 1$) decoding attempt fails to recover the original message, Raptor decoder waits to receive more $I = 50$ bits before beginning the $(p+1)^{th}$ decoding attempt. | | |
| $T_1^p$ - Number of iterations of the inner LT decoding in the $p^{th}$ Raptor decoding attempt. | $T_1^p = 75, \forall p \geq 1$ | $T_1^p = 25, \forall p \geq 1$ | $T_1^p = \begin{cases} 75 & p = 1 \\ 25 & p > 1 \end{cases}$ |

Table 3.3: Values of $I$ and $T_1^p$ in each Raptor decoding algorithm.

Table 3.3 presents the settings for $I$ and $T_1^p$ in each Raptor decoding algorithm. They are chosen based on the existing literature. In particular, $I$ is typically chosen to be 50 [28] or 100 [27]. $I = 50$ is chosen here as it allows more precise measurement of the number of received bits required for a successful decoding $N_F$.

With the inner LT decoding, MPA begins each attempt with no results from

previous attempts. This needs a certain number of iterations to converge. Typically $75 \leq T_1^p \leq 100$ is sufficient to generate the saturated results for MPA [7, 17, 18]. In this section, the lower bound of $T_1^p = 75$ is chosen for MPA. On the other hand, PIMPA uses the decoding results from the previous attempt. It only requires the information from the new received bits for the current attempt able to propagate through the factor graph, thus fewer iterations are sufficient for PIMPA. Moreover, in [7], PIMPA with the ratio $T_1^p/I = 1/2$ showed low complexity. Here $I = 50$ thus $T_1^p = 25$. With LSPIMPA (or TLSPIMPA), the decoding starts late with a high value of $N_S$ and applies 75 (inner LT) iterations for the first decoding attempt; then if it fails, the following attempts will have 25 iterations. The reason of a higher number of iterations for the first attempt is to recover the message immediately; the following attempts, if required, need fewer iterations (as the previous decoding result is reused).

#### 3.3.1.2 Choice of $N_S$

For MPA, $N_S$ is typically chosen to be slightly higher than $k/C$ where $C$ is the channel capacity, which is a function of the SNR. $k/C$ shows the minimum number of received bits required for a successful decoding.

For PIMPA, $N_S$ is chosen to be 150 received bits less than the corresponding $N_S$ of MPA so that: by the time the Raptor decoder with PIMPA has received the same number of received bits as MPA, it already has $150/I = 3$ decoding attempts and thus the total number of inner LT iterations is $3 \times 25 = 75$, equal to $T_1^p$ of MPA.

For LSPIMPA, $N_S$ is selected to be close to $N_F$ so that the number of decoding attempts $f_{da} = \frac{N_F - N_S}{I}$ is kept small. The selection can be done as follows. It first estimates $N_F$ and then sets $N_S$ close to that estimation. As $N_F$ is a random variable, the estimation is based on the empirical data of the number of received bits required for a successful decoding. In this chapter, 30000 transmissions using the Raptor code over a range of channel SNR of [-4dB,7dB] are simulated, and at the end, all values of $N_F$ are recorded. With this statistical data, $N_S$ of LSPIMPA is chosen to be equal to the 95 percentile value of $N_F$ for each channel SNR.

Table 3.4 presents the numerical values selected for $N_S$ of MPA, PIMPA and LSPIMPA for various channel SNRs used for the simulations. Note that, as mentioned in Section 2.3.2, the receiver is assumed to know the SNR of the channel.

| Algorithm | $N_S$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SNR = -2.83dB** | **SNR = -2dB** | **SNR = -1dB** | **SNR = 0dB** | **SNR = 1dB** | **SNR = 2dB** | **SNR = 3dB** |
| MPA | 20450 | 18000 | 15650 | 13900 | 12600 | 11650 | 11000 |
| PIMPA | 20300 | 17850 | 15500 | 13750 | 12450 | 11500 | 10850 |
| LSPIMPA | 21500 | 19100 | 16700 | 14750 | 13450 | 12450 | 11850 |

Table 3.4: Selected values of $N_S$ of MPA, PIMPA and LSPIMPA for different SNR over an AWGN channel.

### 3.3.2 Decoding Complexity

Figure 3.4 compares the average decoding complexity of MPA, PIMPA and LSPIMPA over an AWGN channel with the settings mentioned in the previous section. These results are obtained through averaging $C_{R1}$ and $C_{R2}$ values calculated by Equation 3.8 and Equation 3.9 over 100 runs.

As expected, MPA and LSPIMPA exhibit the highest and the lowest complexity, respectively. Specifically, PIMPA achieves a reduction of 32% to 45% in $C_{R1}$ and 33% to 46% of $C_{R2}$ as compared to MPA. These results are consistent with [7] in which MPA was compared to PIMPA. LSPIMPA has a reduction of 84% to 86% in $C_{R1}$ and 95% in $C_{R2}$ as compared to PIMPA (or a reduction of 90% to 92% in $C_{R1}$ and 97% in $C_{R2}$ as compared to MPA).

PIMPA outperforms MPA in terms of complexity because (i) PIMPA has a lower number of iterations per attempt for the inner LT decoding process ($25 < 75$) as compared to MPA; (ii) simulation results indicate that PIMPA requires fewer received bits for a successful decoding as compared to MPA. This is likely because it reuses the results of previous decoding attempts.

On the other hand, LSPIMPA has the lowest complexity as it keeps the number of decoding attempts as small as possible. These arguments are supported by Table 3.5 showing the average number of decoding attempts of the three algorithms. As shown, the average numbers of decoding attempts of MPA and PIMPA are similar and usually above 10 (PIMPA usually has lower values); while most of the time, the decoder with LSPIMPA succeeds, as expected, in the first attempt.

Figure 3.5 compares the average number of received bits for a successful decoding $E(N_F)$ for each of the three algorithms. The three algorithms have similar $E(N_F)$

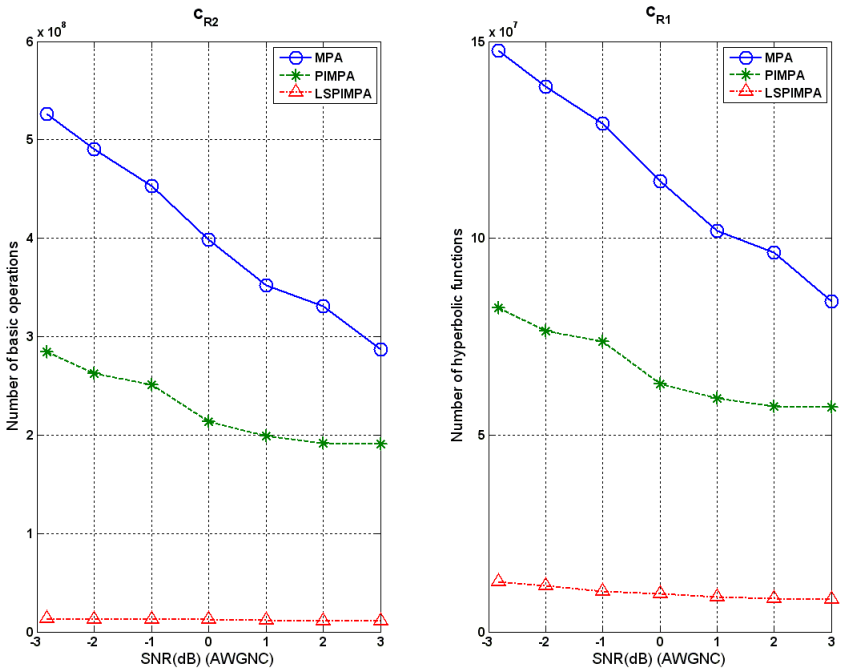



Figure 3.4: Average decoding complexity of MPA, PIMPA and LSPIMPA in terms of $C_{R1}$ and $C_{R2}$.

values: PIMPA has the lowest values, followed by MPA and LSPIMPA requires the highest values. The better $E(N_F)$ with PIMPA is quite reasonable due to the fact that it starts earlier than MPA and re-uses the decoding results. On the other hand, because LSPIMPA uses a value of $N_S$ such that most encoded bit-streams will be able to be decoded in the first attempt, and some of these bit-streams could have been decoded with fewer than $N_S$ received bits, that savings in received bits is lost. The gain is the reduction in the average decoding complexity.

However, the trade-off in $E(N_F)$ of LSPIMPA is not always a disadvantage as the energy for the transmission of these bits is spent equally at the transmitter with or without LSPIMPA. To illustrate this, Figure 3.6 shows the frequency of $N_F$ for MPA and LSPIMPA. It has two sub-figures: the left sub-figure presents the distribution of $N_F$ for MPA over an AWGN channel at SNR = 0dB and the right sub-figure presents the similar results for LSPIMPA. In the left sub-figure, as $N_F$ values for MPA are spreading from 14100 to 15000 bits, it shows that some users experiencing good transmission channels can recover the message before the others. However, with other users are still not able to decode the message successfully, these users are idle

| Algorithm | Average number of decoding attempt | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SNR = -2.83dB** | **SNR = -2dB** | **SNR = -1dB** | **SNR = 0dB** | **SNR = 1dB** | **SNR = 2dB** | **SNR = 3dB** |
| MPA | 13.3 | 13.69 | 14.06 | 12.58 | 12.94 | 12.88 | 12.75 |
| PIMPA | 13 | 13.85 | 14.24 | 12.83 | 12.55 | 12.5 | 12.71 |
| LSPIMPA | 1.08 | 1.07 | 1.02 | 1.06 | 1.01 | 1.02 | 1.1 |

Table 3.5: Average number of decoding attempts for MPA, PIMPA and LSPIMPA.

while the transmitter continues the transmission of the current message. On the other hand, in the right sub-figure, as the $N_F$ values for LSPIMPA concentrate around the value of 14750 bits, it shows that all the users try to start the decoding late at the same time and most of them recover the message in the first attempt. By this way, the $E(N_F)$ is increased. However, this increase is not really a trade-off as the transmitter spends the same amount of energy for the transmission as MPA.

Another method to reduce complexity is to not use received bits that are very uncertain or unreliable. This strategy is employed in conjunction with LSPIMPA and the combined method is called Threshold LSPIMPA (TLSPIMPA). When a threshold on the soft values of received bits is applied, the only change in the parameter setting with TLSPIMPA as compared to LSPIMPA is that the number of received bits of the first decoding attempt $N_S$ is now counted only for reliable received bits. Similar to the strategy of choosing $N_S$ described in Section 3.3.1.2, the statistical data about the number of reliable received bits required for a successful decoding with a threshold is first recorded. Then, the 95% highest value is assigned to $N_S$. All the chosen values for $N_S$ of TLSPIMPA with two thresholds 1.0 and 2.0 are then listed below in Table 3.6.

The average decoding complexity with TLSPIMPA is presented in Figure 3.7. In the right sub-figure of Figure 3.7, TLSPIMPA is observed to have a reduction of 3% to 14% and 4% to 25% in the average $C_{R1}$ as compared to LSPIMPA when the thresholds 1.0 and 2.0 are applied, respectively. On the other hand, in the left sub-figure of Figure 3.7, TLSPIMPA and PIMPA have very close results for the average $C_{R2}$. Also, it is worth noting that the reduction in average decoding complexity is more significant in the low SNR region than in the high SNR region.

Figure 3.8 shows $E(N_F)$ of TLSPIMPA versus SNR. According to the simulation

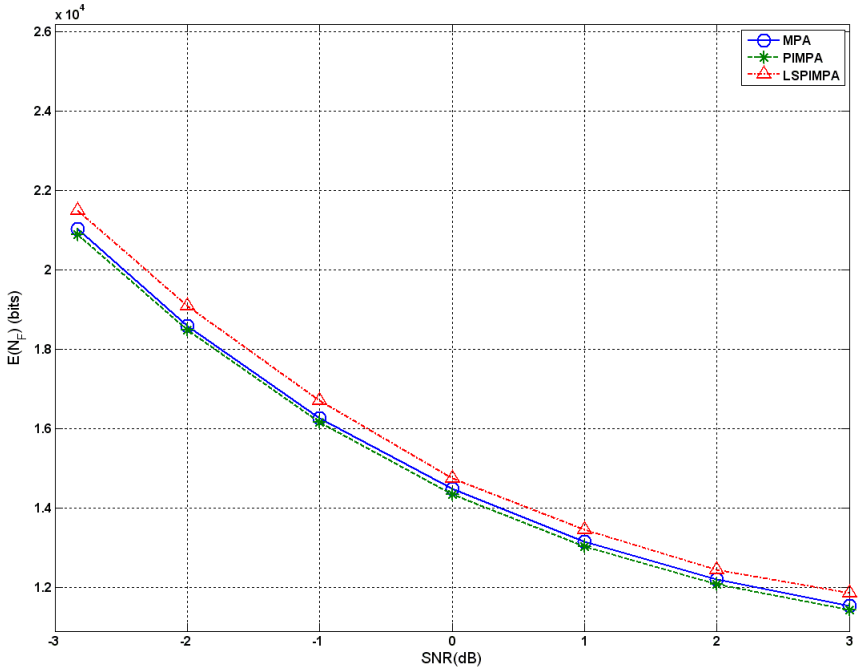



Figure 3.5: Average number of received bits for a successful decoding $E(N_F)$ of MPA, PIMPA and LSPIMPA.

results, TLSPIMPA can have an increase up to 3% and 21% in $E(N_F)$ as compared to LSPIMPA with the thresholds 1.0 and 2.0, respectively.

## 3.4 Effects of The Threshold Technique

As shown in the last section, the threshold technique can be applied to reduce the decoding complexity. In a sense, providing the Raptor decoder with more reliable bits is similar to simulating a better channel quality, and so the Raptor decoder requires fewer received bits to recover the message, thus the decoding complexity is reduced. This section tries to estimate that better channel quality caused by the threshold technique in different ways. For convenience, the effectively improved SNR is designated as the equivalent SNR. In particular, the AWGN channels are considered in the following study. For other channel models, the estimation process for the equivalent SNR can be applied in a similar way.

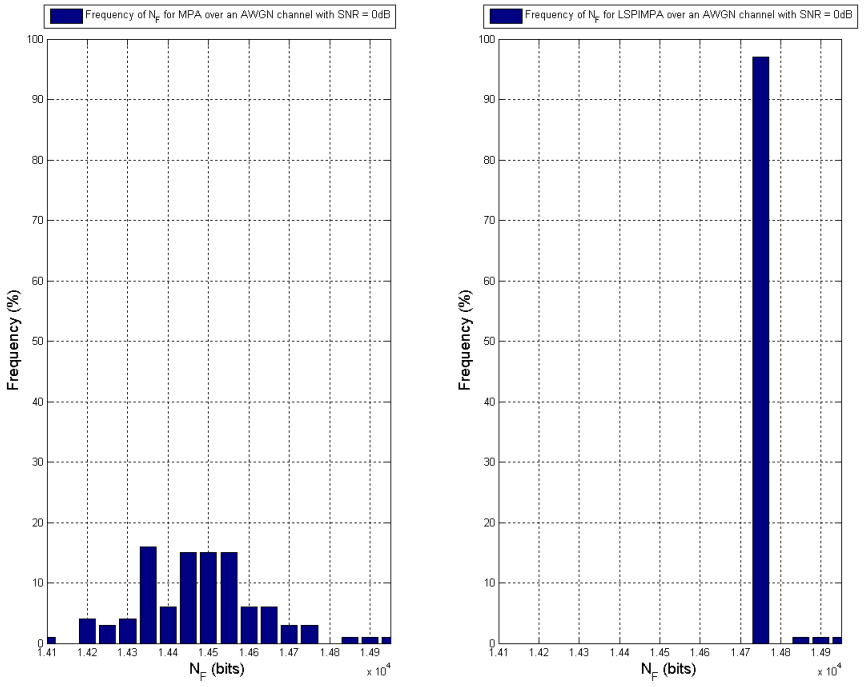



Figure 3.6: Frequency of the number of received bits for a successful decoding $N_F$ for MPA and LSPIMPA over an AWGN channel with SNR = 0dB.

| Algorithm | $N_S$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SNR = -2.83dB** | **SNR = -2dB** | **SNR = -1dB** | **SNR = 0dB** | **SNR = 1dB** | **SNR = 2dB** | **SNR = 3dB** |
| TLSPIMPA with 1.0 | 16950 | 15700 | 14400 | 13300 | 12550 | 11900 | 11500 |
| TLSPIMPA with 2.0 | 13950 | 13350 | 12700 | 12200 | 11800 | 11450 | 11200 |

Table 3.6: Empirical values for $N_S$ of TLSPIMPA.

This section is organized as follows. First, Section 3.4.1 describes three different estimation methods for equivalent SNRs. Then, Section 3.4.2 presents all the numerical results of the estimation of equivalent SNRs.

### 3.4.1 Estimation Methods for Equivalent SNR

For convenience, $K$ is denoted as the SNR value of the underlying transmission channel and $\tilde{K}$ as the equivalent SNR of $K$ when a threshold is applied (obviously, $\tilde{K} > K$). $Nr_F$ is denoted as the number of reliable received bits required for a successful decoding, to be different from the total number of received bits required for a successful decoding $N_F$. Again, $E(.)$ of a symbol means its average value, e.g., $E(Nr_F)$ is the average value of $Nr_F$ and $E(N_F)$ is the average value of $N_F$.

The first method is based on the average number of reliable received bits required for a successful decoding $E(Nr_F)$. Specifically, for each value of the underlying channel SNR $K$, with a threshold is applied, $E(Nr_F)$ is recorded. The equivalent SNR $\tilde{K}$ of $K$ is then defined as the SNR value (when the threshold is applied) at which the average total number of received bits $E(N_F)$ is equal to the recorded $E(Nr_F)$ value. Note that, this method is based on simulation results.

The second and third methods are theoretical estimation processes. For the second method, with each value of $K$ and a threshold is applied, the error probability of received bits is calculated. The equivalent SNR $\tilde{K}$ of $K$ is defined as the SNR value that has the same error probability of received bits as in case of channel SNR $K$ and a threshold is applied. Similarly, the third method is based on the calculations of the average soft value of received bits.

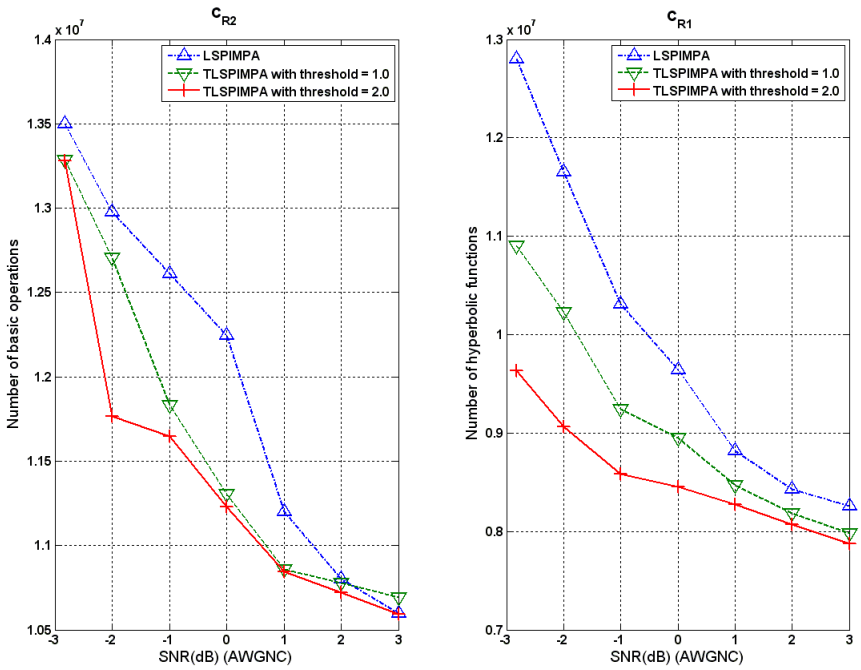



Figure 3.7: Average decoding complexity of LSPIMPA and TLSPIMPA in terms of $C_{R1}$ and $C_{R2}$.

#### 3.4.1.1 Estimating Equivalent SNR Based on The Average Number of Reliable Received Bits Required for a Successful Raptor Decoding Process

Monte Carlo simulations are performed for MPA with three thresholds 1.0, 2.0 and 3.0 over an AWGN channel with SNR range from -3dB to 6dB. For each threshold and each SNR value $K$, the average number of reliable received bits required for a successful decoding $E(Nr_F)$ is recorded. Figure 3.9 illustrates the estimation process for the estimate of equivalent SNR $\tilde{K}$.

The left sub-figure of Figure 3.9 shows $E(Nr_F)$ versus SNR when MPA and the threshold 1.0 are applied; while the right sub-figure shows $E(N_F)$ versus SNR in case when no threshold is set. From the left sub-figure of Figure 3.9, $E(Nr_F)$ can be obtained corresponding to a specific SNR value on the X-axis, e.g., $K$ = 0dB corresponding to $E(Nr_F)$ = 13062 bits. In the right sub-figure, corresponding to the obtained $E(Nr_F)$ is a specific SNR $\tilde{K}$, e.g., $E(Nr_F)$ = 13062 bits on the Y-axis corresponds to the estimated equivalent SNR $\tilde{K}$ = 1.08dB. In other words, with the

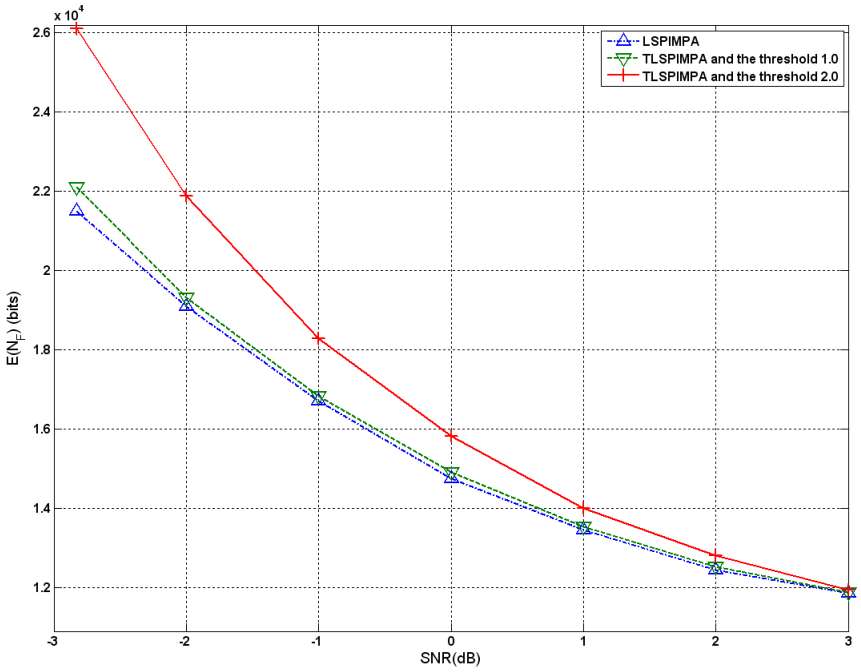



Figure 3.8: Average number of received bits for a successful decoding $E(N_F)$ for LSPIMPA and TLSPIMPA.

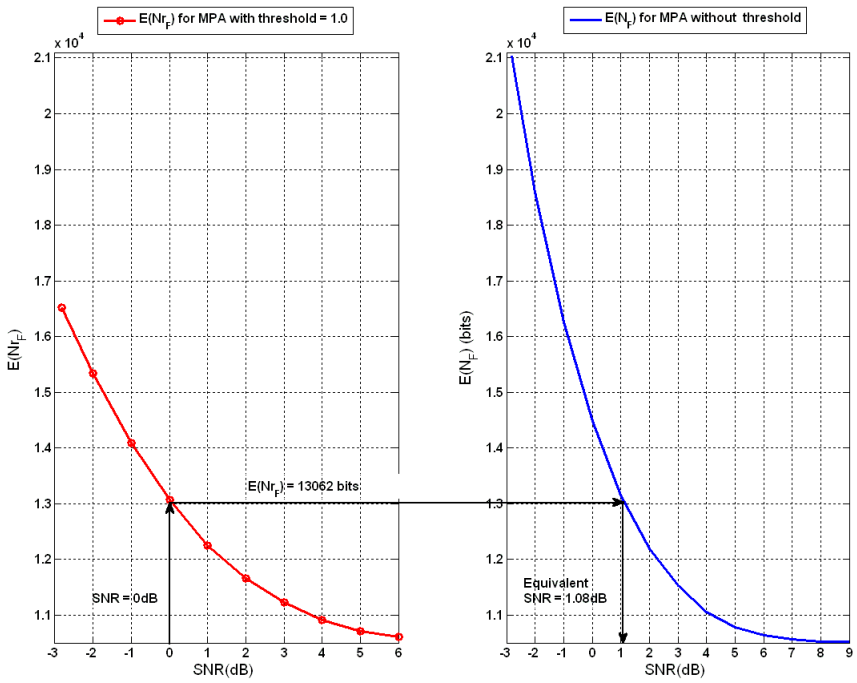



Figure 3.9: The estimation of equivalent SNR $\tilde{K}$ based on the $E(N_F)$ data (average number of received bits for a successful decoding) for MPA without threshold.

threshold of 1.0 over an AWGN channel with $K = 0$dB, the Raptor code is expected to have the required $E(Nr_F)$ equal to the required $E(N_F)$ as in case of an AWGN channel with SNR $\tilde{K} = 1.08$dB when no threshold is set.

The results of equivalent SNR $\tilde{K}$ based on this method are shown in Figure 3.14 and Figure 3.15 of Section 3.4.2 - for all three thresholds 1.0, 2.0 and 3.0.

#### 3.4.1.2 Estimating Equivalent SNR Based on The Error Probability of Received Bits

According to Equation 2.14, the LLR value of a received bit $i$ can be calculated as follows.

$$L(c_i) = \frac{2h_i r_i}{\sigma^2}$$

where $\sigma^2$ is the variance of the channel Gaussian noise; $h_i$ is the channel gain and here, $h_i = 1$ as an AWGN channel is considered. Also, $r_i = h_i e_i + n_i = e_i + n_i$, then

$$L(c_i) = \frac{2(e_i + n_i)}{\sigma^2}.$$

**Case of $e_i = +1$ (bit 1)**

$$L(c_i|e_i = +1) = \frac{2(1 + n_i)}{\sigma^2} \tag{3.10}$$

Equation 3.10 shows $L(c_i|e_i = +1)$ is a linear transformation of a Gaussian random variable $n_i$ [37]. Therefore, $L(c_i|e_i = +1)$ is also a Gaussian random variable with

mean

$$m = E\left[L(c_i|e_i = +1)\right] = E\left[\frac{2(1 + n_i)}{\sigma^2}\right] = \frac{2}{\sigma^2} + \frac{2}{\sigma^2}E[n_i] = \frac{2}{\sigma^2} \tag{3.11}$$

and variance

$$v = E\left[\left(L(c_i|e_i = +1) - \frac{2}{\sigma^2}\right)^2\right] = E\left[\left(\frac{2}{\sigma^2}n_i\right)^2\right] = \frac{4}{\sigma^4}E[n_i^2] = \frac{4}{\sigma^2} \tag{3.12}$$

Assuming a threshold $S$ ($S \geq 0$) is set on the soft outputs of the demodulator so that all the received bits having their LLR values falling into the range $[-S; S]$ are not output into the Raptor decoder. In this case, the error probability of received bits $P_1^S$ (the upper script $S$ indicates the threshold value, the lower script 1 presents the condition of $e_i = +1$) is defined as:

$$P_1^S = \frac{P\{L(c_i|e_i = +1) < -S\}}{1 - P\{-S \leq L(c_i|e_i = +1) \leq S\}}$$

where the numerator $P\{L(c_i|e_i = +1) < -S\}$ is the probability of the LLR value of received bit $c_i$ less than $-S$. If that happens, the received bit is wrongly identified as bit 0. While the denominator $1 - P\{-S \leq L(c_i|e_i = +1) \leq S\}$ is the probability that the LLR value of received bit $c_i$ is outside the unreliable region $[-S; S]$, i.e., $c_i$ is reliable. By this definition, $P_1^S$ is the probability that the LLR value of a received bit is considered "reliable" but wrongly indicates the correct bit value.

A simple example is presented here to illustrate how to calculate $P_1^S$. Assuming SNR $= -2.83$dB and $S = 1.0$. If BPSK modulation uses $\mp 1$ for bits 0 and 1, respectively, the Gaussian noise variance is related with the SNR as $\sigma^2 = 1/2\text{SNR}$. Changing SNR value to linear scale, the noise variance is obtained as $\sigma^2 = 0.9593$. Hence, $L(c_i|e_i = +1)$ is a Gaussian random variable with mean $m = 2/\sigma^2 = 2.0848$ and variance $v = 4/\sigma^2 = 4.1696$. Figure 3.10 shows the pdf of $L(c_i|e_i = +1)$. Section Y (the area under the curve of $L(c_i|e_i = +1) \in [-S; S]$) presents the value of $P\{-S \leq L(c_i|e_i = +1) \leq S\}$ and section X (the area under the curve of $L(c_i|e_i = +1) \in$

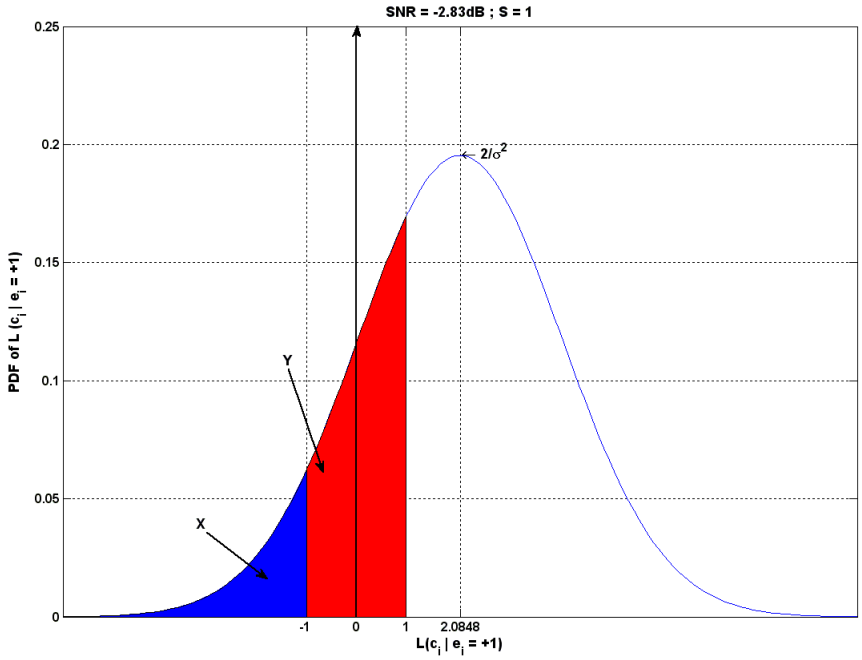



Figure 3.10: Pdf of $L(c_i|e_i = +1)$ over an AWGN channel with SNR $= -2.83$dB and $S = 1.0$.

$(-\infty; -S))$ presents the value of $P\{L(c_i|e_i = +1) < -S\}$. Then, the probability of error is calculated as:

$$P_1^S = \frac{P\{L(c_i|e_i = +1) < -S\}}{1 - P\{-S \leq L(c_i|e_i = +1) \leq S\}} = \frac{X}{1 - Y} = 0.1536$$

**Case of encoded bit $e_i = -1$ (bit 0)**

Similarly, the probability of error of recived bit $c_i$ in case of $e_i = -1$ is defined as

$$P_0^S = \frac{P\{L(c_i|e_i = -1) > S\}}{1 - P\{-S \leq L(c_i|e_i = -1) \leq S\}}.$$

As they are symmetrical: $P_1^S = P_0^S$.

**Error probability of received bits**

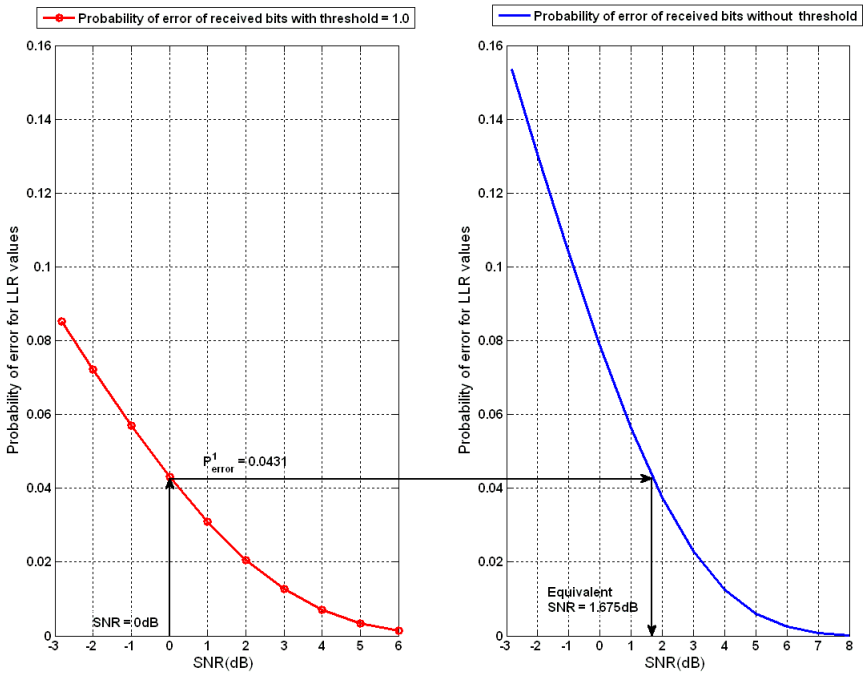



Figure 3.11: The estimation of equivalent SNR $\tilde{K}$ by using the error probability of received bits.

Finally, the error probability of received bits $P^S_{error}$ is calculated as:

$$P^S_{error} = P\{e_i = +1\}P^S_1 + P\{e_i = -1\}P^S_0 = \frac{1}{2}P^S_1 + \frac{1}{2}P^S_0$$
$$= P^S_1.$$

Figure 3.11 illustrates the estimation process for the equivalent SNR $\tilde{K}$ by using the error probability of received bits over an AWGN channel.

The left sub-figure of Figure 3.11 presents the error probability of received bits versus SNR in which the threshold is $S = 1.0$. For example at SNR $K = 0$dB, the error probability of received bit is $P^{S=1.0}_{error} = 0.0431$ . According to the curve in the right sub-figure of Figure 3.11 (error probability of received bits versus SNR without threshold), the error probability of 0.0431 in the Y-axis corresponds to the SNR value of 1.675 in the X-axis. Hence, the estimated equivalent SNR $\tilde{K}$ is 1.675dB.

All the numerical equivalent SNR $\tilde{K}$ results based on the error probabilities of received bits obtained by the estimation process above are later presented in Figure 3.14 of Section 3.4.2.

Furthermore, as the distributions for $L(c_i|e_i = +1)$and $L(c_i|e_i = -1)$ are derived

earlier in this section, it is convenient to include the pdf of $L(c_i)$ in this part.

$$\begin{aligned} P\{L(c_i) \leq y\} &= P\{e_i = +1\}P\{L(c_i|e_i = +1) \leq y\} + P\{e_i = -1\}P\{L(c_i|e_i = -1) \leq y\} \\ &= \frac{1}{2}P\{L(c_i|e_i = +1) \leq y\} + \frac{1}{2}P\{L(c_i|e_i = -1) \leq y\} \end{aligned}$$

Therefore,

$$\begin{aligned} f_{L(c_i)}(r) &= \frac{1}{2}f_{L(c_i|e_i=+1)}(r) + \frac{1}{2}f_{L(c_i|e_i=-1)}(r) \\ &= \frac{1}{2}\left[\frac{1}{\sqrt{2\pi v}}\exp\left(\frac{(r-m)^2}{2v}\right) + \frac{1}{\sqrt{2\pi v}}\exp\left(\frac{(r+m)^2}{2v}\right)\right] \end{aligned} \tag{3.13}$$

where $m$ and $v$ are calculated based on the noise variance of the underlying AWGN channel, as in Equations 3.11 and 3.12.

The pdf's of $L(c_i)$, according to Equation 3.13, over AWGN channels with three values of SNR = $-2.83$dB, 0dB and 3dB are plotted in Figure 3.12; with the shaded part in each sub-figure presents the distribution of $L(c_i)$ in [-3;3]. The figure demonstrates the logical expansion of $L(c_i)$ towards the more reliable region (high values) as the channel SNR increases, suggesting the threshold technique would have less significant effect in the high SNR region.

#### 3.4.1.3 Estimating Equivalent SNR Based on The Average Value of $L(c_i)$

As suggested by Figure 3.12, the distribution of $L(c_i)$ is one of the characters that reflects the channels quality: the higher the SNR value, the higher the *absolute* value of $L(c_i)$. However, the exact distribution of $L(c_i)$ can not be used for a numerical estimation process, instead the statistical average value of $L(c_i)$, i.e., $E[L(c_i)]$, is proposed. Unfortunately, $E[L(c_i)] = 0$ for all SNR values as the pdf of $L(c_i)$ is always symmetric over 0. To overcome this inconvenience, $E[L(c_i|e_i = +1)]$ or $E[L(c_i|e_i = -1)]$ is then used instead of $E[L(c_i)]$. As they are symmetric, in this part, the estimation process is conveniently based on $E[L(c_i|e_i = +1)]$: the more positive the value of $E[L(c_i|e_i = +1)]$, the better the channel quality. The estimation of the equivalent SNR $\tilde{K}$ then proceeds as follows.

1. $E[L(c_i|e_i = +1)]$ values over an AWGN channel without threshold are obtained for each value of SNR by averaging 10000 $L(c_i|e_i = +1)$ values in each case.

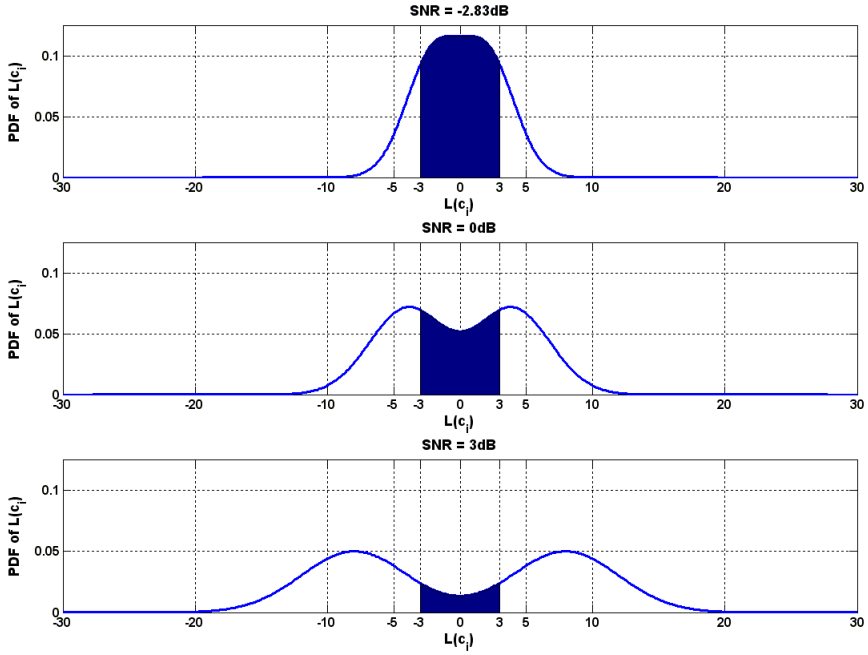



Figure 3.12: Pdf's for $L(c_i)$ over an AWGN channel with SNR $= -2.83$dB, 0dB and 3dB.

2. $E[L(c_i|e_i = +1)]$ values over an AWGN channel with threshold 1.0, 2.0 and 3.0 are obtained for each value of SNR by averaging 10000 $L(c_i|e_i = +1)$ values in each case.

3. The equivalent SNR $\tilde{K}$ is found by the process presented in Figure 3.13.

The left sub-figure in Figure 3.13 shows the $E[L(c_i|e_i = +1)]$ over an AWGN channel with the threshold 1.0. For example, a particular SNR value $K = 0$dB corresponds to $E[L(c_i|e_i = +1)] = 4.45397$. According to the right sub-figure of Figure 3.13 ($E[L(c_i|e_i = +1)]$ without threshold), $E[L(c_i e_i = +1)] = 4.45397$ in the Y-axis corresponds to the SNR value of 0.464dB in the X-axis. Hence, the estimated equivalent SNR $\tilde{K}$ for the SNR value $K = 0$dB with the threshold 1.0 is 0.464dB.

All the numerical equivalent SNR $\tilde{K}$ results estimated based on $E[L(c_i|e_i = +1)]$ are later presented in Figure 3.15 of Section 3.4.2.

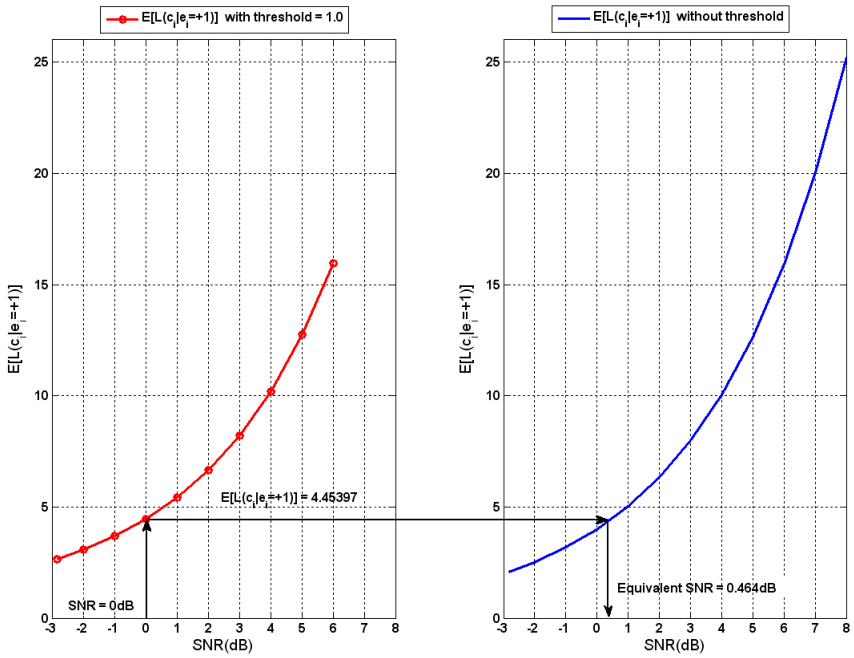



Figure 3.13: The estimation of equivalent SNR $\tilde{K}$ by using $E[L(c_i|e_i = +1)]$.

### 3.4.2 Numerical Results for Estimated Equivalent SNRs

Figure 3.14 presents the results for equivalent SNR $\tilde{K}$ values that are estimated based on the average number of reliable bits for a successful decoding $E(Nr_F)$ and the error probability of received bits for three different thresholds 1.0, 2.0 and 3.0 over an AWGN channel. It plots the equivalent $\tilde{K}$ versus its original $K$. Continuous curves present the results based on $E(Nr_F)$, while dash curves present the results based on the error probability of received bits. Note that all the simulations use MPA for Raptor decoding process. It is observed that in all cases, the equivalent SNR $\tilde{K}$ is always higher than the original $K$. This is reasonable due to the fact that the Raptor decoder is provided with more reliable bits, and thus simulating a better transmission channel. The higher the threshold is set, the more reliable the information input into the Raptor decoder and thus the higher the equivalent SNR $\tilde{K}$ compared to the original $K$. Another notice is that the two groups of estimation methods do not give identical results. As the figure shows, the theoretical estimation for equivalent SNR (based on error probability calculation) is slightly higher than the practical estimation (based on the average number of reliable bits for a successful

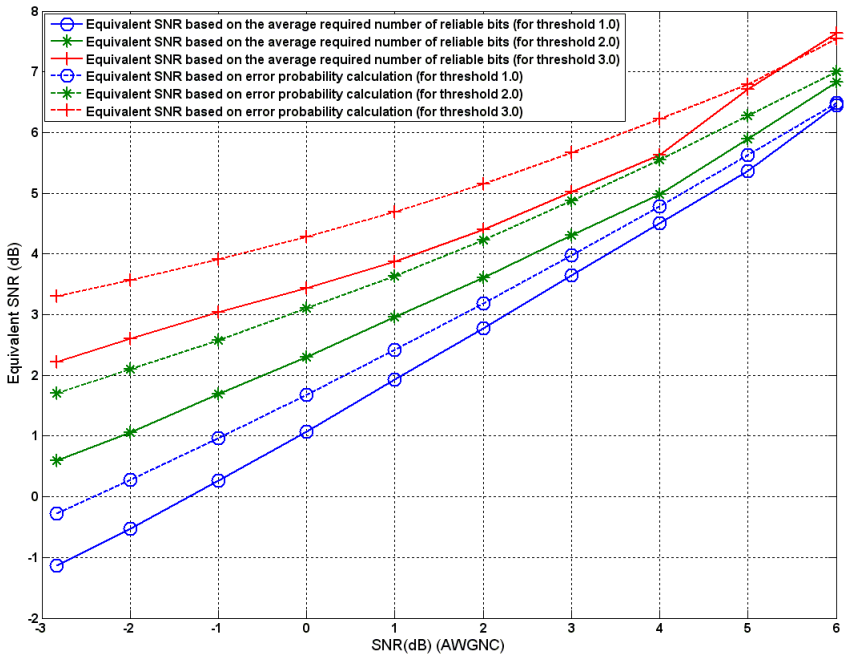



Figure 3.14: Equivalent SNR $\tilde{K}$ estimated based on the average number of reliable bits for a successful decoding $E(Nr_F)$ and error probability calculation for different thresholds.

decoding $E(Nr_F)$).

The estimation of equivalent SNR $\tilde{K}$ based on the error probability calculation gives higher values than the practical ones as shown in Figure 3.14. On the contrary, in Figure 3.15, the method based on the average soft value of received bits $E[L(c_i|e_i = +1)]$ gives lower estimated values. Figures 3.14 and 3.15 indicate that the estimation based on the error probability of received bits produces estimated values closer to the practical ones.

To further improve the accuracy of the estimated values, a Least-Square estimation approach [38] is considered. Figure 3.16 presents the modified estimated equivalent SNR $\tilde{K}$ based on the error probability of received bits, while Figure 3.17 shows similar modified results based on the average soft value of received bits. They indicate that the modified estimation method based on the error probability of received bits produces estimated values closer to the simulation results than the method based on the average soft value of received bits.

**Note on the penalty in** $E(N_F)$: Applying a threshold means fewer bits are

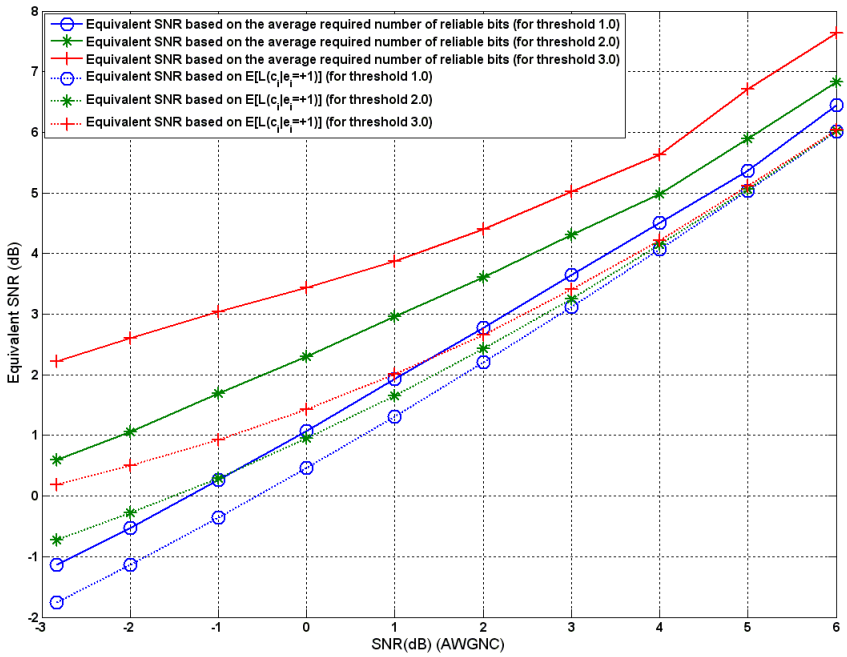



Figure 3.15: Equivalent SNR $\tilde{K}$ estimated based on the average number of reliable bits for a successful decoding $E(Nr_F)$ and the average soft value of received bits $E[L(c_i|e_i = +1)]$ for different thresholds.

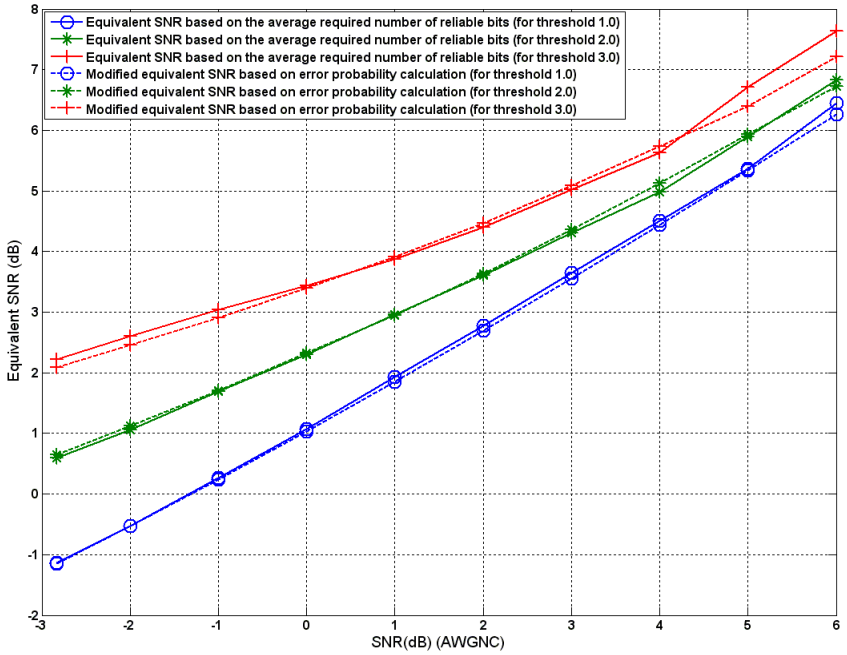



Figure 3.16: Modified estimation of equivalent SNR $\tilde{K}$ based on the error probability of received bits compared with the practical results based on the average number of reliable bits for a successful decoding $E(Nr_F)$ with different thresholds.

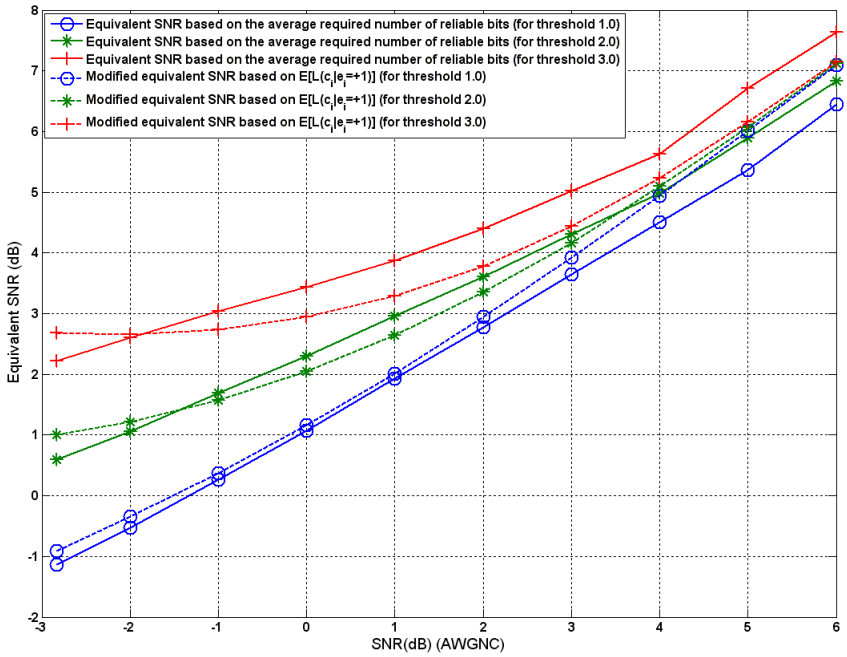



Figure 3.17: Modified estimation of equivalent SNR $\tilde{K}$ based on the average soft value of received bits $E[L(c_i|e_i = +1)]$ compared with the practical results based on the average number of reliable bits for a successful decoding $E(Nr_F)$ with different thresholds.

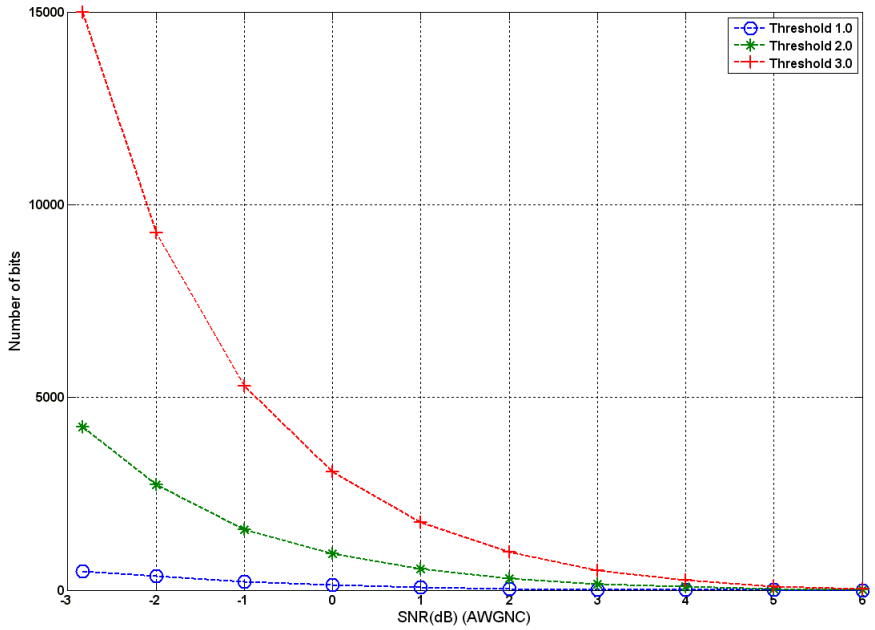



Figure 3.18: Penalty in the average number of received bits for a successful decoding $E(N_F)$ for different thresholds.

required for a successful decoding, however it requires more received bits (since the unreliable received bits are discarded). Figure 3.18 illustrates the penalty in $E(N_F)$ that must be paid as a consequence of applying the threshold. It presents the number of received bits that the Raptor decoder needs to collect *more* in three cases of thresholds 1.0, 2.0 and 3.0, compared with the case with no threshold. MPA is used in all cases. As it shows, all the difference in $E(N_F)$ is significant in the low SNR region and tends to be reduced as the SNR increases. Note that, as expected, the difference in $E(N_F)$ with the threshold 1.0 is lowest of all in the whole range of SNR values; while the difference in $E(N_F)$ with threshold 3.0 is so severe at low SNRs that it might not be suitable for some delay-sensitive applications.

## 3.5 Summary

In this chapter, methods of reducing the Raptor decoding complexity are investigated. First, PIMPA tries to reuse the results of previous decoding attempts to

recover the original message with less iteration. To further reduce the decoding complexity, LSPIMPA is proposed to be a combintation between PIMPA and a strategy of selecting an appropriate number of received bits $N_S$ to begin the first attempt so that the required number of decoding attempts is limited to minimal numbers. Finally, another method to reduce the decoding complexity, called the threshold technique, is also considered. With this technique, only received bits with absolute LLR values greater than a threshold are considered as "reliable" bits and used for decoding process. The combination of LSPIMPA and the threshold technique, named TLSPIMPA, is then investigated.

Section 3.3 compares the decoding complexities of MPA, PIMPA, LSPIMPA and TLSPIMPA. The results with PIMPA are similar to those presented in [7], in which PIMPA not only achieves a reduction of 32% to 45% in $C_{R1}$ (i.e., the number of hyperbolic functions) and 33% to 46% of $C_{R2}$ (i.e., the number of basic operations), but also requires a slightly lower average number of receive bits for a successful decoding $E(N_F)$ as compared to MPA. On the other hand, LSPIMPA is shown to be able to achieve a reduction of 90% to 92% in $C_{R1}$ and 97% in $C_{R2}$ with an increase of 1% to 3% in $E(N_F)$ as compared to MPA. On top of that, with TLSPIMPA, depending on what threshold is chosen, 3% to 25% further reduction in $C_{R1}$ is achievable; however, the increase in $E(N_F)$ is also high - an increase up to 21%. All the significant results, either reduction or increase, are concentrated in the low SNR region.

Observing that providing the Raptor decoder with reliable information (i.e., using the threshold technique) is similar to transmitting over an improved channel, Section 3.4 investigates estimation methods for this virtual improved channel. The results show that the estimation method based on the error probability of received bits give estimated values closer to the simulated ones as compared to the method based on the average soft LLR value of received bits.

# Chapter 4

# Raptor Coding Rate Performance over Nakagami-$m$ Fading Channels and a Cooperative Wireless Relay Network

In this chapter, the coding rate performance of Raptor codes over Nakagami-$m$ fading channels and in a cooperative Wireless Relay Network (WRN) is studied.

The chapter is organized as follows. Section 4.1 presents the Raptor coding rate performance over different Nakagami-$m$ fading channels. Section 4.2 first describes existing cooperative protocols over a single-relay WRN. One of these is the Time Division (TD) protocol. It was shown to relatively outperform other protocols in terms of transmission energy expenditure. The proposed cooperative protocol, named Phase-2 Simultaneous Transmission (PST) protocol, is then considered, aiming to reduce the total transmission time. Note that, the PST protocol is an adaptation for a single-relay WRN of the Asynchronous protocol which was first proposed for a multiple relays WRN in [3]. At the end, the simulation results for the PST protocol's performance in terms of the average energy expenditure $E(E_S)$ and the average transmission time $E(T)$ are presented and compared with that of the TD protocol. Finally, a summary of the chapter is given in Section 4.3.

## 4.1 Raptor Coding Rate Performance over Point-to-point Nakagami-$m$ Fading Channels

R. Palanki and J. S. Yedidia in [13] were the first to study the performance of the Raptor code (the Raptor code version mentioned in Section 2.2.4) in the Binary Symmetric Channel (BSC) and AWGN channel. They showed that the Raptor code can achieve a coding rate close to the Shannon channel capacity and the result is consistent over a range of noise levels for both BSCs and AWGN channels. O. Etesami and A. Shokrollahi in [14] also showed analogous results, confirming the capacity-approaching Raptor coding rate performance over Binary Input Memoryless Symmetric Channel (BIMSC) - the general model of the BSC and AWGN channel.

Later, J. Castura and Y. Mao [16] investigated the Raptor coding rate performance over wireless fading channels with different fading block size $B$ (the number of received bits in which the fading gain is unchanged). They showed that over Rayleigh fading channels with $B = 1000$, 5000 and infinity (quasi-static fading), the average Raptor coding rate is close to the average capacity of the channel. Further, B. Sivasubramanian and H. Leib [17, 18] confirmed that the Raptor code could deliver capacity-approaching coding rate performances over both Rician and Rayleigh fading channels and that these results held for all fading block sizes. Note that, all of the mentioned results here were obtained by simulations.

The Nakagami-$m$ fading channel models generally fit better for a wide range of fading statistics than the Rayleigh and Rician models. Specifically, the Nakagami distribution with an appropriate "shape factor" $m$ fits the measured received envelope data better than the Rayleigh or Rician distributions [24–26]. Motivated by that fact, the Raptor coding rate performance over different Nakagami-$m$ fading channels is studied in this section.

In the simulations presented here, all the bits are transmitted using the BPSK modulation scheme. The transmission channels are assumed to be fast fading channels ($B = 1$) where the channel gain $h_i$ for each received bit is independent with each of the others. At the receiver, the MPA is used for decoding. The interval between two consecutive decoding attempts is chosen to be $I = 50$ bits. The average number of received bits for a successful decoding $E(N_F)$ is obtained by averaging 100 $N_F$ values for each channel SNR value. Finally, the average Raptor coding rate is

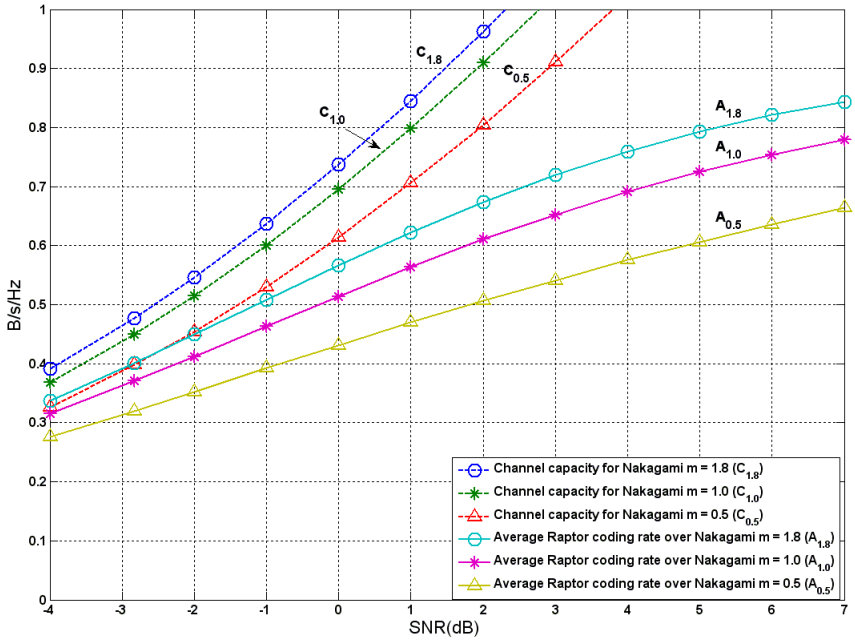



Figure 4.1: Average Raptor coding rate over Nakagami-$m$ fading channels.

represented by $k/E(N_F)$. Since the spectral efficiency of BPSK is 1 (b/s/Hz), the overall rate performance of the point-to-point link using BPSK and Raptor codes under consideration is also represented by $k/E(N_F)$.

Figure 4.1 shows the Raptor coding rate (b/s/Hz) versus SNR over three different Nakagami-$m$ fading channels with $m = 1.8$, 1.0 and 0.5. These $m$ values reflect different fading conditions of the channel. Note that $m = 1.8$ and 1.0 are equivalent to Rician and Rayleigh fading channels, respectively (Section 3.2.2 of [39]), while $m = 0.5$ presents a severe fading condition. Here, the theoretical channel capacity $C = \log_2(1 + \text{SNR})$ is also plotted for the range up to 1 (b/s/Hz) for comparison.

Using the results in Figure 4.1, Figure 4.2 plots the difference $D_m(\text{SNR}) = C_m(\text{SNR}) - A_m(\text{SNR})$ where $C_m$ denotes the theoretical (average) channel capacity, $m$ is the parameter of the Nakagami-$m$ channel and $A_m$ denotes the average Raptor coding rate.

The results in Figure 4.2 indicate the good performance of the Raptor code in the low SNR region: as the channel SNR decreases, the average Raptor coding rate approaches the average channel capacity. For example, at SNR $<$ 0dB, the achievable rate of the BPSK-Raptor coding scheme is less than 0.2 (b/s/Hz) away from the

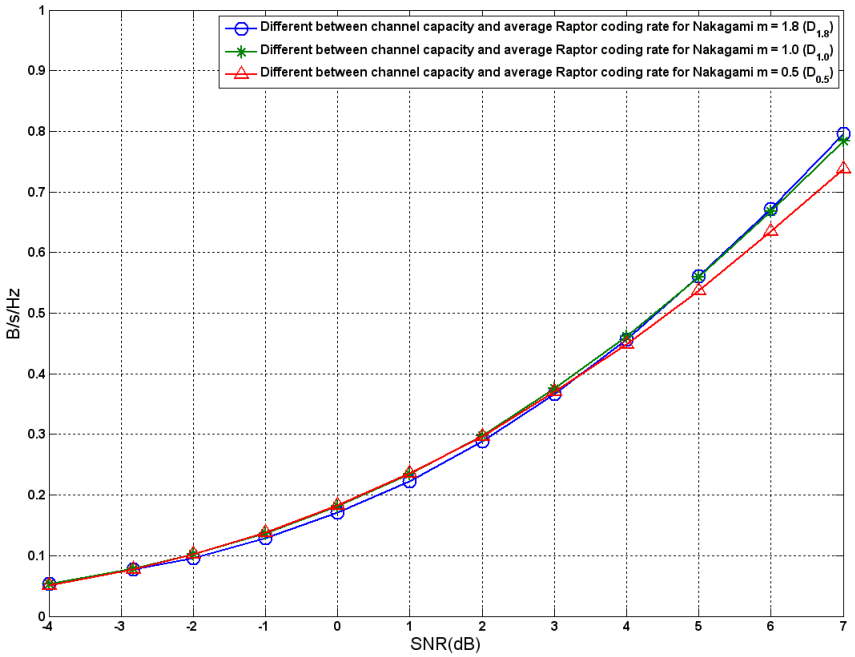



Figure 4.2: Difference between the average channel capacity and the average Raptor coding rate versus the SNR of Nakagami-$m$ fading channels.

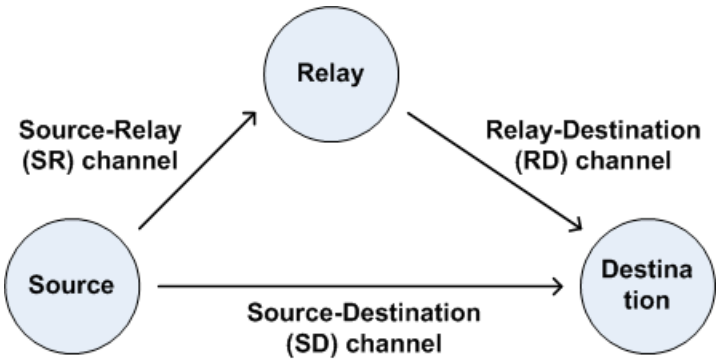



Figure 4.3: Model of a single-relay WRN.

channel capacity. As SNR increases, the average rate achieved by the BPSK-Raptor coding scheme is limited by the BPSK and the coding rate, and hence is far from the channel capacity.

## 4.2 Raptor Coding Rate Performance in a Single-Relay Wireless Relay Network

Figure 4.3 shows the cooperative single-relay WRN under consideration, in which one relay helps a source to transmit information to a destination. The cooperative transmission interval can be divided into two phases. In Phase 1, the source broadcasts the data stream to both the relay and the destination through the Source-Relay (SR) and Source-Destination (SD) channels, respectively. The relay listens to the SR channel and tries to decode the message. In Phase 2, after it has recovered the message successfully, the relay assists the source in transmission to the destination through the Relay-Destination (RD) channel.

There are several different two-phase cooperative protocols proposed for relay transmission strategy. Section 4.2.1 will discuss the two-phase cooperative protocols used in our study. Section 4.2.2 describes simulation configurations, scenarios and parameter settings. Section 4.2.3 presents the simulation results in terms of the average energy expenditure $E(E_S)$ and the average transmission time $E(T)$.

### 4.2.1 Two-phase Cooperative Protocols under Consideration

Fountain codes in WRNs have been considered. In [3], performance of fountain codes in a collaborative multiple-relay WRN was examined for the following asynchronous two-phase protocol:

Phase 1: The source broadcasts to all the relays; the relays listen to the source while the destination is idle as the source-destination (SD) link is assumed to be unavailable.

Phase 2: The relays that successfully recovered the source message in Phase 1 re-encode the source message into a different encoded bit streams and asynchronously transmit them to the destination by using the Code Division Multiple Access (CDMA) scheme [40] so that multiple received signals at the destination are distinguishable.

In [28], fountain codes over the cooperative single-relay WRN as shown in Figure 4.3 were studied for three following two-phase cooperative protocols.

A. Distributed Space-Time Code (DSTC) protocol

Phase 1: The source encodes the message using a Raptor code and broadcasts the encoded signal. Both the relay and the destination listen. After the relay has recovered the message, it encodes the message in the same way as the source.

Phase 2: The Alamouti scheme [41, 42] is used for the cooperative transmission between the source and the relay. Specifically, the output bits from the source and the relay are synchronized over time to form consecutive pairs and then transmitted sequentially: if the source transmits a pair of bits $s_1$ and $s_2$, the relay will transmit a pair of bits $-s_2^*$ and $s_1^*$ at the same time. Note that, this cooperation requires that the source and the relay generate exactly the same encoded version of the message.

At the destination, these pairs of bits are received and decoded to extract the Raptor encoded data. Finally, the destination uses a Raptor decoding algorithm to decode the encoded data and recover the message.

B. Time Division (TD) protocol

Phase 1: The TD protocol has the same Phase 1 with the DSTC protocol, that is the source broadcasts and both the relay and the destination listen.

Phase 2: Unlike the DSTC protocol, for the TD protocol, the source stops transmitting and the relay alone communicates with the destination. Note that in this case, the relay can encode the message different from the source and transmit that encoded bit stream to the destination.

The destination recovers the message by decoding received signals from the source in Phase 1 and from the relay in Phase 2.

C. Two Hop (TH) protocol

Phase 1: Phase 1 is slightly different in the TH protocol compared to the DSTC and the TD protocols. Specifically, only the relay listens to the source in Phase 1 while the destination is idle. This scenario usually happens as the SD channel is unavailable due to the long distance between the source and the destination.

Phase 2: The relay alone transmits the encoded signal to the destination in Phase 2 of TH protocol and it can encode the message different from the source. In contrast to the two previous protocols, here, the destination recovers the message by decoding received bits collected *solely* from the relay in Phase 2.

Study results in [28] indicate that the TD protocol outperforms the other two protocols in most cases. For an unlimited number of bits that the destination can receive, the TD protocol performs better than the DTSC and TH protocols in terms of energy expenditure. On the other hand, for an upper bound on the number of bits that the destination can receive, the TD protocol provides a smaller number of times in which the destination fails to recover the message than the DTSC and TH protocols. Based on this conclusion, the TD protocol will be considered in our following study.

In addition, another two-phase cooperative protocol for a single-relay WRN using Raptor codes, namely Phase-2 Simultaneous Transmission (PST) protocol, is also considered as follows. Note that, the PST protocol is a modified version for a single-relay WRN of the Asynchronous protocol [3] mentioned above.

Phase 1: The source broadcasts its encoded signal to the relay and destination (i.e., similar to the TD protocol).

Phase 2: Both the source and relay can simultaneously transmit (different) encoded signals to the destination in an asynchronous manner. An appropriate transmission scheme (e.g., CDMA) is assumed so that multiple received signals are distinguishable at the destination.

In the PST protocol, since the relay encodes the message in a different way from the source, the source and the relay transmit, in general, different encoded signals to the destination. The destination simply collects all of them for the Raptor decoding. As a result, in Phase 2, the destination using the PST protocol receives about twice the amount of encoded signals as compared to the TD protocol. Such an advantage can lead to a good performance of the PST protocol in terms of the average transmission time $E(T)$.

### 4.2.2 Set-up for Simulations

#### 4.2.2.1 Coding and Transmissions

For the encoding process, at the source and the relay, the same Raptor encoder mentioned in Section 2.2.4 are implemented; however, the random encoding manner ensures that encoded bits output from the source and the relay are different. Decoding processes at the relay and the destination use MPA and start after the fraction $k/N_S$ becomes a little lower than the underlying average channel capacity. Recall that $N_S$ is the number of received bits that the Raptor decoder has collected in the first decoding attempt. The number of received bits between two consecutive decoding attempts is chosen as $I = 50$ bits. Note that, at the destination, $I$ is counted for received bits from both the source and relay.

Both the source and the relay use the BPSK modulation scheme for transmission. The transmission energy and time duration for each output bit at both the source and the relay are assumed to be $E_b$ and $T_b$, respectively. All transmission channels in the single-relay WRN - the SD, SR and RD channels - are assumed to be fast Rayleigh fading channels (i.e., the channel gain $h_i$ is independent and different for each transmitted bit).

#### 4.2.2.2 Figures of Merit

The following figures of merit are considered.

The average energy expenditure $E(E_S)$ of a transmission over a single-relay WRN is defined as the average transmission energy that the source and the relay have spent to transmit the message to the destination, i.e.,

$$E(E_S) = E(N_F) \times E_b \tag{4.1}$$

where $E(N_F)$ is the average number of bits that the destination receives from both the source and the relay to successfully recover the message.

The average transmission time $E(T)$ of a transmission over a single-relay WRN is defined as the average amount of time during which the transmission of a message happens: it starts when the source begins its broadcast and ends when the destination recovers the message successfully.

The $E(T)$ calculation is slightly different between the TD and PST protocols. For the TD protocol, at each phase of the transmission, there is only one transmitter: it is either the source or the relay. Therefore, the average transmission time $E(T)$ for the TD protocol can be calculated as

$$E(T) = (E(N_{SD1}) + E(N_{RD2})) \times T_b \tag{4.2}$$

where $E(N_{SD1})$ is the average number of bits that the source transmits to the destination in Phase 1 and $E(N_{RD2})$ is the average number of bits that the relay transmits to the destination in Phase 2.

For the PST protocol, the source also transmits in Phase 2 and it continues transmitting until the destination successfully recovers the message; therefore the transmission time is equal to the period in which the source transmits and $E(T)$ can be calculated as

$$E(T) = E(N_{SD12}) \times T_b \tag{4.3}$$

where $E(N_{SD12})$ is the average number of bits that the source transmits to the destination in both Phase 1 and Phase 2.

The values of $E(N_F)$, $E(N_{SD1})$, $E(N_{RD2})$ and $E(N_{SD12})$ are all obtained by averaging 100 simulation values.

#### 4.2.2.3 Simulation Scenarios

For all simulation scenarios, the SR channel always has higher SNR than the SD channel precluding the case of the destination recovering the message before the relay does. Three simulation scenarios are presented in Table 4.1.

| Scenario | SNR of the SD channel | SNR of the SR channel | SNR of the RD channel |
|---|---|---|---|
| RD-better-than-SD | $\gamma$ | $\gamma + 5$ | $\gamma + 5$ |
| RD-equal-SD | $\gamma$ | $\gamma + 5$ | $\gamma$ |
| RD-worse-than-SD | $\gamma$ | $\gamma + 5$ | $\gamma - 5$ |

Table 4.1: Three simulation scenarios.

These scenarios are chosen to investigate the effects of the RD channel quality on the performance of cooperative protocols in terms of $E(E_S)$ and $E(T)$ as compared to that of the SD channel. For example, the TD protocol is expected to reduce the energy expenditure when the RD channel is good, i.e., the RD channel is better than the SD channel, but does it still keep that good performance when the RD channel is worse than the SD channel? Similar questions are asked for the PST protocol's performance in transmission time. Simulation results in Section 4.2.3 give answers to these questions. Note that the 5dB difference presented in these scenarios is just a subjective value used to demonstrate the better or worse in the transmission channel quality.

### 4.2.3 Simulation Results and Discussions

Figures 4.4 - 4.6 show the performance in terms of the average energy expenditure $E(E_S)$ and the average transmission time $E(T)$ of the TD and PST protocols using the Raptor code over a single-relay WRN for all three scenarios: RD-better-than-SD, RD-equal-SD and RD-worse-than-SD, respectively. Note that in each figure, the performance of the point-to-point transmission, i.e., the source transmits data to the destination without any assistance from the relay, is also included to serve as the benchmark.

In Figure 4.4 for the RD-better-than-SD scenario, it is observed that, over the whole range of SNR of interest, the TD protocol has the least average energy expenditure $E(E_S)$ while the average transmission time $E(T)$ of the PST protocol is smallest. Specifically, the TD protocol achieves a reduction up to 6% in $E(E_S)$ as compared to the PST protocol and a reduction up to 8% in $E(E_S)$ as compared to the point-to-point transmission. On the other hand, the PST protocol has a reduction of

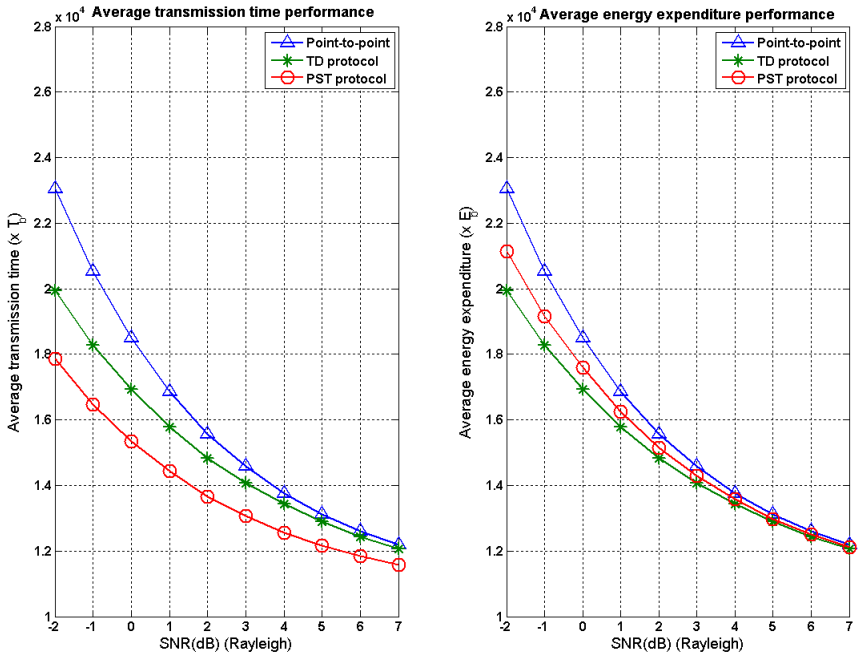



Figure 4.4: Average energy expenditure $E(E_S)$ and average transmission time $E(T)$ of the TD protocol, the PST protocol and the point-to-point transmission over a single-relay WRN with fast Rayleigh fading channels - the RD-better-than-SD case.

4% to 10% in $E(T)$ as compared to the TD protocol and a reduction of 5% to 22% in $E(T)$ as compared to the point-to-point transmission. These performances can be explained as follows.

For energy expenditure, the PST protocol outperforms the point-to-point transmission due to the fact that the source is assisted by the relay in Phase 2 and that the RD channel is better than the SD channel. On the other hand, the TD protocol has less average energy expenditure $E(E_S)$ compared to the PST protocol due to the transmission on the RD channel alone, in Phase 2 of the TD protocol, being more energy-efficient than the transmission on both the SD and RD channels in Phase 2 of the PST protocol. To elaborate this point, the transmissions of two arbitrary bits over the RD channel alone and over both the RD and SD channels are considered. Two bits transmitted through the RD channel alone require an energy amount of $2E_b$. That same amount is also required for two bits with each of them transmitted over the SD and RD channels. However, the bit transmitted over the RD channel has higher reliability than that over the SD channel. As a result, the destination then

will require fewer bits in the TD protocol than in the PST protocol.

For transmission time, the TD protocol performs better than the point-to-point transmission due to the fact that the relay, instead of the source, transmits to the destination in Phase 2 of the TD protocol and the RD channel is better than the SD channel. On the other hand, with the PST protocol, the transmission of both the source and the relay in Phase 2 shortens the transmission time even more.

In Figure 4.5 for the RD-equal-SD scenario, the right sub-figure shows all three curves lie on top of each other, indicating the same average energy expenditure $E(E_S)$ for both protocols and the point-to-point transmission. Because the RD and SD channels have the same SNR in this case, it makes no difference in terms of energy expenditure whether the source transmits to the destination alone or with the help of the relay in Phase 2. However, the transmissions from both the source and the relay in Phase 2 of the PST protocol still reduce the average transmission time $E(T)$, as presented in the left sub-figure of Figure 4.5. Specifically, the TD protocol and the point-to-point transmission have the same $E(T)$ while the PST protocol has a reduction from 5% to 18% in $E(T)$ as compared to both of them.

Unlike the above scenarios, in Figure 4.6 for the RD-worse-than-SD scenario, the TD protocol has higher the average energy expenditure $E(E_S)$ than the PST protocol, and the point-to-point transmission has the smallest $E(E_S)$ of all. The explanation is that when the RD channel is degraded, the bits transmitted from the relay do not help the destination to decode faster while increasing the total energy spent for transmission. On the other hand, in the left sub-figure of Figure 4.5, the average transmission time $E(T)$ of the PST protocol is still smaller than that of the point-to-point transmission. This result indicates the advantage of the PST protocol in transmission time even when the RD channel is worse than the SD channel.

**Conclusions**: Over a single-relay WRN, whenever the SR channel is good, i.e., the relay can recover the message before the destination, the PST protocol has advantages in the average transmission time. The PST protocol, therefore, is suitable for delay-sensitive applications, i.e., live conference or Voice over IP. On the other hand, if the objective is minimizing the average energy expenditure of the transmission, the TD protocol is a better choice than the PST protocol. But this is only true in the case when the RD channel is better than the SD channel. If the RD channel quality is degraded, the TD protocol even spends more energy than the PST protocol.

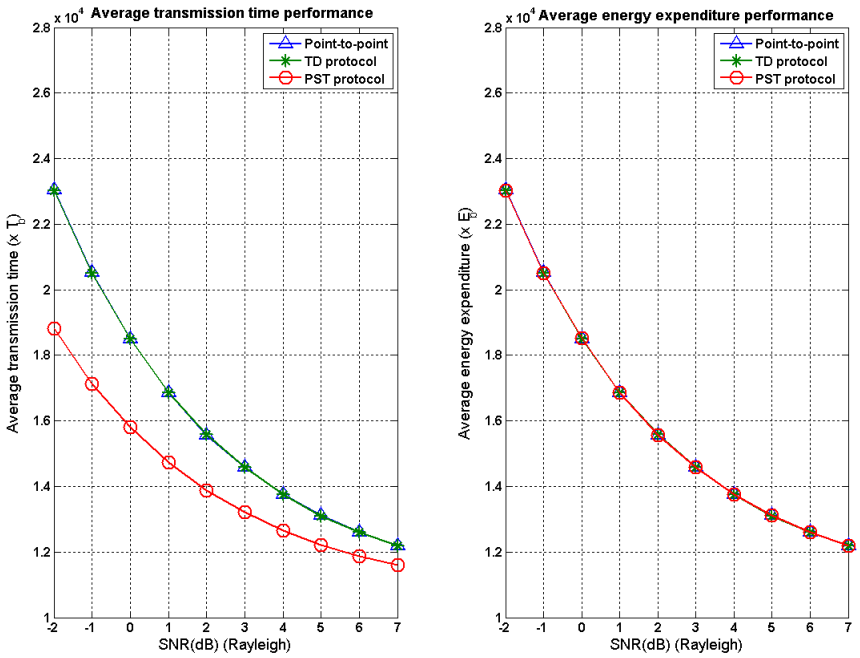



Figure 4.5: Average energy expenditure $E(E_S)$ and average transmission time $E(T)$ of the TD protocol, the PST protocol and the point-to-point transmission over a single-relay WRN with fast Rayleigh fading channels - the RD-equal-SD case.

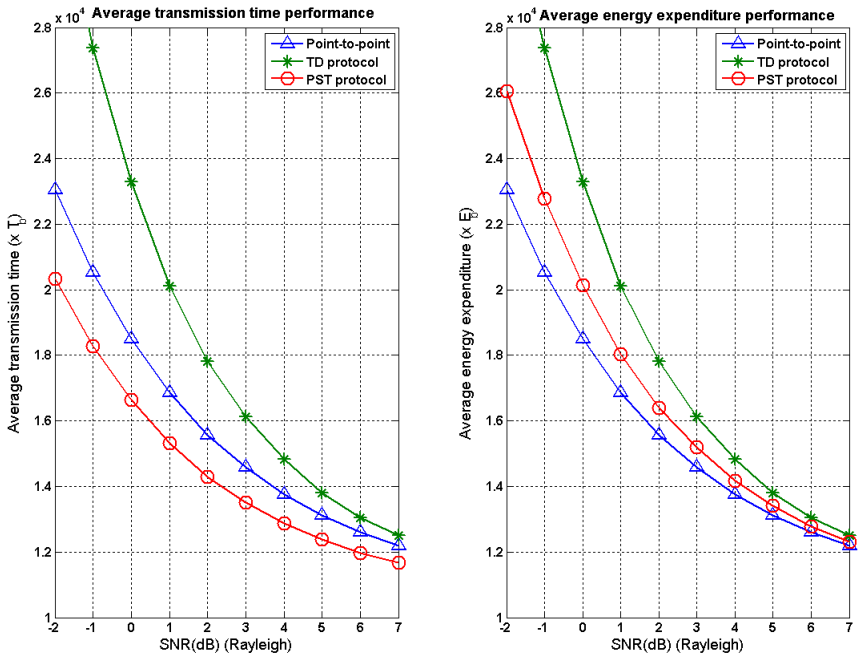



Figure 4.6: Average energy expenditure $E(E_S)$ and average transmission time $E(T)$ of the TD protocol, the PST protocol and the point-to-point transmission over a single-relay WRN with fast Rayleigh fading channels - the RD-worse-than-SD case.

## 4.3 Summary

In this chapter, first, Section 4.1 presents the Raptor coding rate performance over Nakagami-$m$ fading channels. The simulation results confirm that the Raptor code can adapt its coding rate close to the average channel capacity in the low SNR region when the fading condition varies.

Second, Section 4.2 investigates the Raptor coding rate performance over a single-relay WRN and its impacts on the average energy expenditure $E(E_S)$ and the average transmission time $E(T)$ of the TD protocol and PST protocols. The PST protocol targets to reduce the average transmission time $E(T)$ as compared to the TD protocol, by having both the source and the relay transmit encoded signals to the destination in Phase 2. The reduction in transmission time comes from that fact that the receiving rate at the destination is doubled as the source and the relay transmit simultaneously (different) encoded signals. Three simulation scenarios are considered: the SD channel is better than the RD channel; the SD channel is equal to the RD channel; the SD channel is worse than the RD channel. The simulation results show that the PST protocol reduces the average transmission time $E(T)$ in all scenarios, while the TD protocol has lower average energy expenditure $E(E_S)$ than the PST protocol only when the RD channel is better than the SD channel.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

This thesis addresses three problems related to the employment of Raptor codes in wireless transmissions: (i) reducing Raptor decoding complexity; (ii) the Raptor coding rate performance over Nakagami-$m$ fading channels and (iii) the Raptor coding rate performance in a single-relay WRN.

For the first problem, Previously Inherited Message Passing Algorithm (PIMPA) is the widely-used decoding algorithm that was designed to reduce the decoding complexity of Raptor codes by re-using the decoding result from the previous attempt. Chapter 3 of this thesis proposes a modified version of PIMPA called Late Start PIMPA (LSPIMPA) to further reduce the Raptor decoding complexity. LSPIMPA begins the first decoding attempt only after an appropriate number of received bits have been collected so that the required number of decoding attempts is reduced and thus the decoding complexity is also reduced. The simulation results show that LSPIMPA can achieve a reduction of 86% in average decoding complexity while only cause an increase of 1% to 3% in the average number of received bits as compared to PIMPA.

Moreover, the threshold technique of reducing the Raptor decoding complexity is considered. Specifically, observing that, feeding the Raptor decoder with only reliable received bits in the threshold technique is similar to simulating a better channel quality, the issue of estimating this better channel quality is investigated. Three estimation methods are then proposed: (i) Estimation based on the average

number of reliable received bits required to recover the message; (ii) Estimation based on the error probability of received bits; (iii) Estimation based on the average of LLR value of received bits. The second method gives closer estimates to the simulation results.

For the second problem, this thesis provides simulation results on Raptor coding rate performance over Nakagami-$m$ fading channels, with $m = 0.5$, 1.0 and 1.8. The results show the consistently good performances of the Raptor code over different kinds of fading channels: when the fading condition varies, the Raptor code can adapt its coding rate consistently close the underlying channel capacity at low SNR.

For the third problem, two figures of merit of the Raptor coding rate over a single-relay WRN is considered in this thesis: (i) the average energy expenditure $E(E_S)$ and (ii) the average transmission time $E(T)$. The PST protocol is considered to reduce $E(T)$ by allowing both the source and the relay simultaneously transmit to the destination after in the second phase of the cooperative interval. The performance of the PST protocol is compared with that of the TD protocol, which only has the relay transmit to the destination in the second phase, in terms of $E(T)$ and $E(E_S)$. The simulation results show that the PST protocol often outperforms the TD protocol in terms of $E(T)$ while the TD protocol only had advantages in $E(E_S)$ when the RD channel is better than the SD channel. Hence, the PST protocol may be more suitable than the TD protocol for delay-sensitive applications.

## 5.2 Future Work

In this work, all transmissions are assumed to have fading block size $B = 1$, i.e., the channel gain $h_i$ is independent for each bit transmitted over the channel. This model is suitable for relatively fast fading scenarios, e.g., mobile transmissions from a car moving at high speed. There are slower fading scenarios with $B > 1$, e.g., wireless transmissions in office buildings. Further work to study the impact of slow fading on the performance of Raptor codes over Nakagami-$m$ point-to-point links or WRN would be of interest. Note that, with $B > 1$, the threshold technique should be modified so that it is adaptive to the channel SNR. For example, if the channel SNR is low for all $B = 5000$ bits, the magnitude of soft values of all these bits may be all less than the chosen threshold and so they are all ignored for the decoding.

This process is wasteful if the low channel SNR persists for several blocks, and thus modifications to the threshold technique should be considered when investigating slow fading scenarios. Moreover, simulations in the $B > 1$ scenarios may be costly as compared to the research in this thesis. For example, with $B = 1$ and over Nakagami-$m$ fading channel with $m = 1.0$ and SNR = 2.0dB, the receiver requires 15559 bits on average to successfully recover the message. On the other hand, with $B = 5000$, i.e., a block of 5000 bits having the same value for the channel gain $h_i$, the number of the simulations required is 5000 times higher than those in this thesis.

# Bibliography


[1] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1528 – 1540, oct. 2002.

[2] D. MacKay, "Fountain codes," *IEE Proceedings - Communications*, vol. 152, pp. 1062 – 1068, dec. 2005.

[3] A. Molisch, N. Mehta, J. Yedidia, and J. Zhang, "Performance of Fountain Codes in Collaborative Relay Networks," *IEEE Transactions on Wireless Communications*, vol. 6, pp. 4108 – 4119, nov. 2007.

[4] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity. Part I. System description," *IEEE Transactions on Communications*, vol. 51, pp. 1927 – 1938, nov. 2003.

[5] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity. Part II. Implementation aspects and performance analysis," *IEEE Transactions on Communications*, vol. 51, pp. 1939 – 1948, nov. 2003.

[6] Y. Liu, "A low complexity protocol for relay channels employing rateless codes and acknowledgement," in *Proc. IEEE International Symposium on Information Theory*, pp. 1244 – 1248, jul. 2006.

[7] K. Hu, J. Castura, and Y. Mao, "Reduced-complexity decoding of raptor codes over fading channels," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1 – 5, dec. 2006.

[8] A. AbdulHussein, A. Oka, and L. Lampe, "Decoding with early termination for raptor codes," *IEEE Communications Letters*, vol. 12, pp. 444 – 446, jun. 2008.

[9] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st ed., jun. 2003.

[10] S. Papaharalabos, P. Sweeney, B. Evans, P. Mathiopoulos, G. Albertazzi, A. Vanelli-Coralli, and G. Corazza, "Modified sum-product algorithms for decoding low-density parity-check codes," *IET Communications*, vol. 1, pp. 294 – 300, jun. 2007.

[11] V. Chandrasetty and S. Aziz, "A reduced complexity message passing algorithm with improved performance for LDPC decoding," in *Proc. International Conference on Computers and Information Technology (ICCIT)*, pp. 19 – 24, dec. 2009.

[12] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551 – 2567, jun. 2006.

[13] R. Palanki and J. Yedidia, "Rateless codes on noisy channels," in *Proc. International Symposium on Information Theory (ISIT)*, p. 37, jun. - jul. 2004.

[14] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, pp. 2033 – 2051, may 2006.

[15] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Communications Letters*, vol. 10, pp. 46 – 48, jan. 2006.

[16] J. Castura, Y. Mao, and S. Draper, "On rateless coding over fading channels with delay constraints," in *Proc. IEEE International Symposium on Information Theory*, pp. 1124 – 1128, jul. 2006.

[17] B. Sivasubramanian and H. Leib, "Fixed-rate Raptor codes over Rician fading channels," *IEEE Transactions on Vehicular Technology*, vol. 57, pp. 3905 – 3911, nov. 2008.

[18] B. Sivasubramanian and H. Leib, "Fixed-rate Raptor code performance over correlated Rayleigh fading channels," in *Proc. Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 912 – 915, apr. 2007.

[19] T. Mladenov, S. Nooshabadi, and K. Kim, "MBMS Raptor codes design trade-offs for IPTV," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 1264 – 1269, aug. 2010.

[20] L. Al-Jobouri, M. Fleury, and M. Ghanbari, "Raptor coding of H.264 data-partitioned video over a WiMAX channel," in *Proc. IEEE International Conference on Consumer Electronics (ICCE)*, pp. 341 – 342, jan. 2011.

[21] M. Uppal, Z. Yang, A. Host-Madsen, and Z. Xiong, "Cooperation in the low power regime for the MAC using multiplexed rateless codes," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4720 – 4734, sept. 2010.

[22] H. Zhang and G.-S. Kuo, "Raptor code for downlink cooperative wireless cellular networks," in *Proc. IEEE Vehicular Technology Conference (VTC)*, pp. 1 – 5, sept. 2008.

[23] T. Schierl, S. Johansen, C. Hellge, T. Stockhammer, and T. Wiegand, "Distributed rate-distortion optimization for rateless coded scalable video in mobile Ad hoc networks," in *Proc. IEEE International Conference on Image Processing (ICIP)*, vol. 6, pp. 497 – 500, oct. 2007.

[24] N. Nakagami, "The m-distribution, a general formula for intensity distribution of rapid fading," in *Statistical Methods in Radio Wave Propagation* (W. G. Hoffman, ed.), Oxford, England: Pergamon, 1960.

[25] L. Rubio, N. Cardona, S. Flores, J. Reig, and L. Juan-Llacer, "The use of semi-deterministic propagation models for the prediction of the short-term fading statistics in mobile channels," in *Proc. IEEE Vehicular Technology Conference (VTC)*, vol. 3, pp. 1460 – 1464, 1999.

[26] L. Rubio, J. Reig, and N. Cardona, "Evaluation of nakagami fading behaviour based on measurements in urban scenarios," *AEU - International Journal of Electronics and Communications*, vol. 61, no. 2, pp. 135 – 138, 2007.

[27] J. Castura and Y. Mao, "Rateless coding for wireless relay channels," *IEEE Transactions on Wireless Communications*, vol. 6, pp. 1638 – 1642, may 2007.

[28] X. Liu and T. J. Lim, "Fountain codes over fading relay channels," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 3278 – 3287, jun. 2009.

[29] A. Ravanshid, L. Lampe, and J. Huber, "Signal combining for relay transmission with rateless codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 508 – 512, jul. 2009.

[30] R. Nikjah and N. Beaulieu, "Achievable rates and fairness in rateless coded relaying schemes," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 4439 – 4444, nov. 2008.

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

[32] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28 – 41, jan. 2004.

[33] T. Richardson and R. Urbanke, "The renaissance of Gallager's low-density parity-check codes," *IEEE Communications Magazine*, vol. 41, pp. 126 – 131, aug. 2003.

[34] A. Wiesel, J. Goldberg, and H. Messer-Yaron, "SNR estimation in time-varying fading channels," *IEEE Transactions on Communications*, vol. 54, pp. 841 – 848, may 2006.

[35] D. Pauluzzi and N. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," *IEEE Transactions on Communications*, vol. 48, pp. 1681 – 1691, oct. 2000.

[36] A. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2117 – 2120, may 2009.

[37] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering (2nd Edition)*. Addison-Wesley, 2 ed., aug. 1993.

[38] G. Strang, *Linear Algebra and Its Applications*. Brooks Cole, feb. 1988.

[39] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[40] T. S. Rappaport, *Wireless Communications: Principles and Practice (2nd Edition)*. Prentice Hall, 2 ed., jan. 2002.

[41] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1451 – 1458, oct. 1998.

[42] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, pp. 1456 – 1467, jul. 1999.