A Fountain Code Forward Error Correction Strategy for SensLAB Applications

by

Jaco du Toit

Thesis presented in full fulfilment of the requirements for the degree

Master of Science in Engineering

at Stellenbosch University

Supervisor: Dr. R. Wolhuter

April 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2014 .....

Abstract

A Fountain Code Forward Error Correction Strategy for SensLAB Applications

J. du Toit

Thesis: BEng (E and E)

April 2014

The discovery of sparse graph codes, used in forward error correction strategies, has had an unrivaled impact on Information theory over the past decade. A recent advancement in this field, called Fountain codes, have gained much attention due to its intelligent rate adaptivity, and lend itself to applications such as multicasting and broadcasting networks. These particular properties can be considered valuable in a wireless sensor network setting as it is capable of providing forward error correction, and the added conceptual network protocol related extensions.

A wireless sensor network testbed in France, called SensLAB, provides an experimental facility for researchers to develop and evaluate sensor network protocols, aside from a simulation environment. Tremendous value can be added to the SensLAB community if an appropriate forward error correction design, such as Fountain codes, is deemed feasible for use on such a platform.

This thesis investigates the use of Fountain codes, in a binary erasure channel environment, as a forward error correction strategy for the distribution of reliable data content over the SensLAB platform. A short message length LT code using two different decoding mechanisms were developed and evaluated for possible implementation. Furthermore, a short message length Raptor code was developed by using supplementary theory and optimisation techniques that permit scalability in terms of the message size. The results favoured the Raptor code design as it performs close to near optimal while still satisfying the rateless- and universality property, at low computational complexity.

Uittreksel

A Fountain Code Forward Error Correction Strategy for SensLAB Applications

J. du Toit

Tesis: BIng (E en E)
April 2014

Die ontdekking van yl-grafiekkodes, van toepassing op foutkorreksie strategieë, het onlangs 'n ongeewenaarde impak op Informasieteorie gehad. In 'n onlangse vooruitgang in hierdie veld, genoem Fonteinkodes, word daar meer fokus geplaas op die intelligente tempo aanpassingsvermoë van hierdie kodes, wat nuttige toepassing kan inhou in multi-saai- en uitsaai netwerke. Hierdie eienskappe kan moontlik as waardevol beskou word in draadlose sensor netwerke weens die fout regstellingsvermoë en die bykomende konseptuele netwerk protokol verwante uitbreidings.

'n Draadlose sensor netwerk toetsplatvorm in Frankryk, genoem die SensLAB, bied navorsers die geleentheid om eksperimentele sensor netwerk protokolle te ontwikkel en te toets buite 'n tipiese simulasie-omgewing. Groot waarde kan bygevoeg word aan die SensLAB gemeenskap indien 'n geskikte foutkorreksie strategie ontwikkel word, soos Fonteinkodes, en as geskik beskou kan word vir hierdie platvorm.

In hierdie tesis word Fontein kodes saam met die SensLAB platvorm ondersoek, binne die raamwerk van 'n binêre verlieskanaal, om vir foutkorreksie oor die verspreiding van betroubare data in SensLAB op te tree. 'n Kort boodskap LT kode word voorgestel deur van twee verskillende dekoderings meganismes gebruik te maak.'n Alternatief, genaamd Raptorkode, was ook ondersoek. 'n Raptor kode. 'n Kort boodskap Raptor kode, wat ontwikkel is met bykomende teorie en optimeringstegnieke, word ook voorgestel. Die bykomende tegnieke bied 'n skaleerbare boodskap lengte terwyl dit tempoloos en universeel bly, en lae kompleksiteit bied.

Acknowledgements

I would like to express my sincere gratitude to my family and the following people and organisations:

Founder of Information theory:

Claude Elwood Shannon

Pioneers in sparse graph codes:

Prof. R. G. Gallager
Prof. Michael Luby
Prof. Amin Shokrollahi
Prof. David J.C. MacKay

University of Stellenbosch:

Prof. J du Preeze
Prof. A E Krzesinski
Prof. WJ Perold
Prof. M Kidd
Prof. GJ van Rooyen
Dr. R Wolhuter
Dr. HA Engelbrecht
Dr. MA Muller
Peter Hayward
Stephan Gouws
Dirk Bosman
Dirk Badenhorst
Waldo Minnaar
Marc de Klerk
Oswald Jumira
Mariska du Preez
FJ Olivier
Riaan Wiid

Rian Singels
Christiaan Brand
JP Meijers
Anita van der Spuy

University of Newcastle:

Dr. S J. Johnson

University of Bristol:

Dr. D Sejdinović

Conferences:

ICSNC, Barcelona, Spain, 2011
WiMAX Forum® Cape Town, South Africa, 2011
ICEAA-IEEE APWC, Cape Town, South Africa, 2012
ADHOCNETS, Paris, France, 2012

Industry:

INRIA
MWeb
MIH Medialab
Qualcomm
Rudi Jansen
Eugene van der Westhuizen
Antonie Roux
Jacques van Niekerk
Leon Coetzee
Valerie Wentworth

Dedications

Hierdie tesis word opgedra aan my oupa, FR Smit (25-10-1928 tot 24-08-2009).
Quia amasti me, fecisti me amabilem - (Psalm 139)

Contents

List of Figures

List of Tables

Nomenclature

Acronyms:

AL-FEC	Application Layer Forward Error Correction
ARQ	Automatic Repeat-Request
BEC	Binary Erasure Channel
BER	Bit Error Ratio
BSC	Binary Symmetric Channel
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
DVB	Digital Video Broadcasting
CD	Compact Disk
CDF	Cumulative Distribution Function
CRC	Cyclic Redundancy Check
DSP	Digital Signal Processing
FCS	Frame Check Sequence
FEC	Forward Error Correction
GE	Gaussian Elimination
GF	Galois Field
ID	Identifier
INRIA	Institut national de recherche en informatique et en automatique
IP	Internet Protocol
IPTV	Internet Protocol Television
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
LLN	Law of Large Numbers
LQI	Link Quality Indication
LT	Luby Transform
MAC	Media Access Control
MCU	Micro-Controller Unit
MBMS	Multimedia Broadcast-Multicast Services
MDS	Maximum Distance Seperable
ML	Maximum Likelihood
NP	Nondeterministic Polynomial-time

PCO	Pre-Code Only
PDD	Probabilistic Degree Distribution
PEC	Packet Erasure Channel
PEG	Progressive Edge Growth
PER	Packet Error Rate
QoS	Quality of Service
Raptor	Rapid Tornado
RISC	Reduced Instruction Set Computing
RS	Reed-Solomon
RSSI	Received Signal Strength Indication
SNR	Signal-to-Noise Ratio
SP	Sum Product
TCP	Transport Control Protocol
TDD	Time Devision Duplexing
UDP	User Datagram Protocol
WSN	Wireless Sensor Network
WiMAX	Worldwide Interoperability for Microwave Access
XOR	Exclusive-OR

Variables:

p_i	Probability Space
k	Source Data
x	Encoded Data
x^*	Received Encoded Data
k^*	Received Decoded Data
A_i	Input Alphabet
A_o	Output Alphabet
ϵ	Erasure or Error
p	Erasure Probability
y_i	Output Symbol Probability
x_i	Input Symbol Probability
Γ	Communication Channel
X	Random Binary Input Source Set
Y	Random Binary Output Set
H	Entropy
I	Mutual Information
C	Channel Capacity
N	Block Code Length
R	Code Rate
σ	Overhead
$\mathbb{F}_q$	Finite Field
q	Alphabet Size

LDPC Codes:

k	Message Bits
n	Codeword Length
r	Code Rate
m	Redundant Bits
C	Codeword
H	Parity-Check Matrix
G	Generator Matrix
x	Valid Codeword
x^*	Received Codeword
d	Hamming Distance
c_i	Stored Codeword

BEC LDPC Codes:

ϵ	Erasure or Error
p	Erasure Probability
G	Generator Matrix
H	Parity-Check matrix
I	Identity Matrix
m	Message Vector
r	Redundancy Vector
c	Codeword Vector
$\hat{x}$	Estimated Codeword Vector
z	Syndrome Vector
j	Column Weight
k	Row Weight
K	Information Length
w_c	Column Weight
w_r	Row Weight

LT Codes:

s_i	Source Data
S_k	Source Symbols
E_N	Encoded Symbols
k	Message Length
δ	Decoding Failure Probability
c	Free Parameter

N	Number of Symbols for Decoding
$\Omega(x)$	Polynomial Degree Distribution
d	Degree
v	Summation Value
$\rho(d)$	Ideal Soliton Degree Distribution
$\tau(d)$	Robust Soliton Degree Distribution
R	Luby Transform
Z	Normalising Factor

Raptor Codes:

R	Code Rate
n	Length of LDPC Pre-Code
λ_i	Edge Perspective Column Weight
ρ_j	Edge Perspective Row Weight
$\lambda(x)$	Polynomial Edge Perspective Column Weight
$\rho(x)$	Polynomial Edge Perspective Row Weight
v_i	Node Perspective Column Weight
h_j	Node Perspective Row Weight
δ	Decoding Failure Probability
ς	Undecoded Fraction of Symbols
ϵ	Raptor Code Overhead
x_0	Smallest Root
f	Linear Programming Cost Vector
A	Linear Programming Constraints Matrix
b	Linear Programming Inequality Column Vector
w_d	Optimal Degree Weights
Ω	Raptor Degree Distribution
$\Omega(x)$	Polynomial Raptor Degree Distribution
m	Amount of Symbols for Decoding
a	Average Degree
d_{max}	Maximum Degree
M	Output Message String
$E_{j,i}$	Edge Label
A_i	Set of Bits in i-th Parity-Check Equation
B_j	Set of Bits in j-th Parity-Check Equation

Chapter 1

Introduction

1.1 Background

Sensor networks received major popularity from a broad spectrum of research disciplines over the past decade, moreover from the *Information theory* community. This is accentuated by the growing interest for new insights obtained from the data captured by these sensors. The application of advanced applied statistics, such as *Machine Learning*, has revolutionised the way we think about data and as such influenced the perception of data aggregation methodologies by many scientists today. A large number of discrete sensor nodes, capable of recording data, can be organised and connected via networks composed of several hundred to several thousand entities. The captured data can comprise of measurements ranging from temperature, humidity ratios, luminosity, velocity, acceleration, seismic waves, water levels, electricity usage, and many other essential features capable of shaping our understanding of the world. Conceptually, the learning algorithms combined with this data seem eminently suitable for the discovery of new knowledge in disciplines such as: Oceanography, Epidemiology, Environmental studies, Geomatics; and even vehicle traffic and energy monitoring for future *Smart homes* and *Smart grid* technologies, to name a few [1].

In 2008 a wireless sensor grid testbed, called *SensLAB*, was developed by academia and industry partners in France. It provides a competitive and innovative experimental facility for researchers and industrial users interested in designing and evaluating large-scale sensor network applications. Stellenbosch University was selected, among others, to participate in a wide range of joint projects in collaboration with INRIA, a French public science and technology research institution, which is one of the four hosts of the SensLAB platform.

The platform infrastructure comprises of more than a thousand sensor nodes at four different geographical locations, and allows for experiments to be carried out, which can validate the algorithms in actual hardware, aside from a simulation environment. SensLAB members actively participate in developing

tools for the platform mainly in an effort to improve node energy consumption and manage information propagation by means of network coding. Until now modern strategies enabling reliable communication between these nodes have not been researched for application in SensLAB. It is, however, important that these strategies be evaluated and made available for the platform, as the testbed is widely used for industrial and scientific research projects, as a precursor to wider application.

Users of this platform is provided with a wide range of software tools to use in order to reinforce their designs, compilations and simulations. These services are open to the public through an open access multi-user web-portal where scripts can be uploaded to the platform over the internet, and results from the experiments downloaded after execution. As a result SensLAB have seen many developed packages and tools from a dedicated SensLAB community.

The platform is distributed among four host countries in France namely: INRIA Lille - Nord Europe, Strasbourg- LSiiT, INRIA Grenoble - Rhône-Alpes, and INRIA Rennes - Bretagne Atlantique. Each site hosts 256 sensor nodes with different characteristics in order to diversify application development. All experimental data is collected and stored during an experiment to support subsequent analyses. The value of such a technology is emphasised by its potential to lower the entry cost of actual operational experimentation, where conventional experimentational setups usually involve a complex and burdensome setting (a proof-of-concept evaluation usually fails due to cost implications and other intervening factors).

The interpretation of corrupted data, as a result of node communication errors, need to be reduced or eliminated in order to avoid confusing and incorrect statistical interpretations and to reduce simulation time on a scarce resource. Over the past two decades there have been significant advances in Information theory, particularly in research on error correction coding, capable of remedying such concerns. *Shannon's fundamental theory of information* galvanised the field of telecommunication by demonstrating the achievable limits for such attempts [2]. This limit known as the *Shannon capacity* has been unattainable for long, and practically implementable coding schemes able to approach this limit in a computationally efficient way, have not been successful, until comparatively recently.

Pioneers in the field of sparse graph coding theory have changed the way we think about error correction and paved the way, from vigorous theoretical constructs to practically implementable solutions, for the development of *forward error correction coding* schemes, utilisable in modern technology. *Fountain codes* are a family of sparse graph codes that improve on conventional forward error-correction codes. Rather than designing a fixed coding scheme, which is suitable only for a specific noisy communication channel, the aim of Fountain coding is to intelligently adapt to the varying or unknown channel conditions by performing close to the Shannon capacity, at low complexity [3]. The former renders these codes rateless, in the sense that the original message can be

recovered by any aggregate of an unlimited number of independent encoded symbols. These properties make Fountain codes particularly suitable for *multicasting* and *broadcasting* applications suitable for wireless sensor networks where sensor nodes are exposed to different channel characteristics at varying link conditions. Since Fountain coding has attained a striking momentum in Information theory research we attempt to investigate this paradigm, and combine its unprecedented capabilities with application to wireless sensor networks using the SensLAB platform. The rationale for utilising SensLAB is that it provides an accessible, flexible and realistic test environment. If the applicability and functionality of concepts can be proven in that setup, there is very good reason to be confident about the possible success of subsequent practical applications.

1.2 Motivation

The research team at INRIA require an investigation on advanced and robust communication link coding techniques suitable for their SensLAB platform. The implementation of such techniques should provide reliable communication over the noisy channels that these sensor nodes are exposed to. The type of research involved also required wider application in the general advancement of forward error correction technologies.

Wireless sensor network (WSN) technologies are accompanied by new technical challenges. One such challenge is to ensure successful and complete delivery of transmission messages between nodes under independently fluctuating link conditions. If message errors occur it can corrupt the received data, which render post data analytics useless. Therefore, the primary challenge is to successfully deliver error-free data between sensor nodes. Given the centrality of this task it is equally important to be as zealous on sensor node hardware limitations as on an error correction strategy. These nodes are rather limited in calculation capacity, memory availability and energy consumption [4].

Until now there has been no concrete research on the application of Fountain codes pertinent to the SensLAB platform and the general consent around what variant of Fountain code to consider for such an implementation remains relatively unclear. Moreover, considering the appropriate error correction scheme may allow for prominent protocol related extensions, which traditional error correction strategies may not provide. For this reason it is important to provide a design and testing framework in which parameters can be adjusted to support additional implementable network strategies in SensLAB.

In most cases a WSN environment will require the delivery of short messages (e.g. temperature or energy level measurements.). However, the literature, on highly developed Fountain codes, abounds with examples on large message lengths, since it was rapidly extended to video streaming applications

over IP based networks [3, 5, 6, 7]. In addition, the theory seems to suggest that the message length is directly proportional to the efficiency of the code [3]. In other words, shorter message length codes, required by WSNs, may perform poorly and require special design. This is anticipated to be the reason why the literature provides inadequate examples for the design of scalable Fountain codes specifically with regard to the message length. *Raptor codes* represent a type of Fountain code that is typically associated with large message lengths exhibited by popular degree distributions used in most designs (see appendix A.1). Fountain codes may be preferred on WSNs for the following reasons:

- It retains a residual one-to-many communication capability [8],
- It has intelligent rate adaptation that requires no feedback channel [3],
- It allows for low computational complexity and can be implemented over the application layer [3, 9].

These properties needed thorough investigation as each may provide lucrative application within the sensor node constraints.

It is implicit that the consideration for such codes on SensLAB poses much difficulty with regard to the manifested challenges. The subject of this work is first to investigate some of the literature on Fountain codes. This is followed by the design, analyses, and comparison of candidate schemes over a channel that approximates the SensLAB communication environment, while providing a scalable design. The result of such work should provide corroborating evidence for, or against, the Fountain coding ideal in SensLAB applications.

1.3 Research Objectives

The primary objective of this thesis is to investigate the feasibility of Fountain codes as a forward error correction strategy for SensLAB implementation generalised to wireless sensor networks. A simulation environment is considered in order to help determine the operational status and performance of the error correction strategy by modelling the appropriate conditions and by allowing parameter optimisation. Test cases should be identified in which different decoding approaches and different message lengths can be compared in order to identify a viable candidate strategy. This can be accomplished by designing, evaluating and comparing different candidate Fountain codes in a simulated environment, which approximates the SensLAB infrastructure. Ultimately an illustration of their expected behaviour, performance comparison and limitations is required to validate the feasibility of each candidate code in SensLAB. Providing a framework of the design steps and algorithms is important, since

the researchers at INRIA will need the design procedure when embedding the codes into the actual hardware.

The major theme of this work attempts to answer the following questions:

- What is the Fountain coding paradigm?
- Can such codes be extended to, or even considered for, the SensLAB platform? This is an important potential outcome, affecting future implementations on the platform.

We should answer these questions by investigating the supporting theory, develop appropriate coding strategies, test each code in simulation over estimated channel conditions and further evaluate the results with regard to SensLAB implementation. We should highlight the main concepts formally by presenting a comprehensive literature study on the evolution of Fountain codes in order to determine any possible practical shortcomings that may originate from the theory. Additionally, this will engage and direct the reader in understanding the results and rationale behind Fountain codes, and evidently its adoption and benefit for the SensLAB research platform, with the possible subsequent wider applications. The following section will explain the methodology and elaborate on the main objectives.

1.4 Research Methodology

We first investigate the capabilities of the sensor node hardware, and then derive a channel model. We then study the fundamental theory on channel modelling and forward error correction codes, in particular Fountain codes, and identify beneficial code properties and subsequent algorithms that may contribute to our objective. These properties and algorithms may be used to reinforce scalability in the candidate codes. A scalable design is required so that SensLAB researchers have access to the important parameter selections. Results from these codes reflecting the expected theoretical performance should also be presented. For convenience the research methodology can be divided into the following key objectives:

1. An investigation of the SensLAB platform and its hardware and software limitations.
2. A study of the fundamental theory and derivation on channel models.
3. The identification of an appropriate channel model for the SensLAB.
4. The identification of an efficient error-correction scheme for SensLAB.
5. A thorough investigation of the theoretical underpinnings of Fountain codes.

6. The identification of candidate Fountain codes conforming to the SensLAB criterion.

7. The consideration of software tools capable of creating the design framework and testing environment.

8. The design and evaluation of the candidate Fountain codes over the selected channel model.

Matlab is a numeric computing environment offering useful features (e.g. advanced matrix manipulations, random number generators, modular algorithm implementation, and plotting functionality). In addition, it includes built in functions and extra toolboxes that can simultaneously benefit the design cycle and reduce design time. For these reasons it was considered as the development and testing environment.

1.5 Summary of Results and Complementary Work

The results presented in this thesis were obtained by assuming that erroneous messages are detected and discarded by an *error-detection* mechanism outside the scope of this work. It also assumes that the sensor node hardware has limited capability in capturing quantifiable signal-reception information that may be used in other iterative *soft-decision* algorithms. As a result a particular Fountain code was considered, known as an *erasure resilient* code. A description of the principles of operation of such a code and a framework for its design was presented. Different message frame sizes was considered to illustrated the capability, efficiency and performance of the strategy at various code lengths.

The *binary erasure channel* was assumed between nodes, and considered throughout the design phase of the two Fountain codes (the *LT code* and the *Raptor code*). Two decoding strategies were identified for the LT code, *Gaussian elimination decoding* and *belief propagation decoding*. Each were evaluated and compared. For the considered strategies Gaussian elimination decoding may provide less redundancy at higher computational complexity, while belief propagation may provide higher redundancy at lower computational complexity. The Gaussian decoder is recommended for SensLAB applications that require very small message lengths, since the redundancy introduced by the belief propagation decoder becomes undesirable at such lengths. Belief propagation decoding is suitable for larger message lengths.

The Raptor code required the design of a small message length *high-rate irregular LDPC code*, which subsequently required the design of an appropriate *H-matrix*, which is a complex research topic on its own. Pre-developed H-matrices with pre-defined rates could have been selected, however, they do

not allow for a scalable solution in terms of code-rate, or an alternative design in terms of the message lengths. An approximated H-matrix design algorithm, called the *progressive edge growth*, was considered instead and used to design an H-matrix for the LDPC code. Results indicated that this technique is suitable in the design of the Raptor code for SensLAB, as the algorithm can produce scalable and functional small message length high-rate H-matrices. Supplementary to the H-matrix design the LT code part, in the Raptor code, required a distinct degree distribution. A *Linear programming optimisation* technique was used to design an optimal degree distribution for the Raptor code. Subsequently, a test method known as *Density evolution* was used to evaluate the designed degree distribution. Results from this evaluation method revealed that the Linear programming optimisation technique produced a degree distribution capable of functioning within its design constraints. All these tools were used to design and evaluate the expected functionality of the different components in the Raptor code.

These methods were used to develop three test cases for both the LT code and Raptor code in order to evaluate the error-correction characteristics as a function of three different message lengths. Three different sized message length codes (100, 500 and 1000), for the Raptor code, were produced with the above mentioned summarised design methodology. Three different sized LT codes (100, 500 and 1000) were produced using standard methods. The performance of both codes regarding overhead as a function of decoding probability, and error-correction capability over a simulated erasure channel were tested and compared. Results indicate that the *universality property* and the *rateless property* applies to both designed codes and at the larger message length (1000) both codes perform closer to the *Shannon capacity*, which reflect efficiency in terms of overhead. The LT code's performance becomes increasingly irregular in terms of the projected overhead (especially at lower message lengths) and results demonstrate that a larger overhead, in terms of encoded symbols, is required for successful decoding compared to the Raptor code. Results from the Raptor code indicate both lower redundancy and improved decoding probability, as it forms tighter bounds around the expected overhead as a function of the decoding probability. Moreover, the Raptor code also provides lower decoding complexity and is highly recommended as the most feasible error-correction scheme for SensLAB.

The provided codes are well in range of SensLAB specifications in terms of hardware and software capability, except for the larger LT codes that uses the Gaussian decoder. The transceiver modem has built-in *CRC detection* and is restricted to packetised binary outputs. From the available operating systems, the *Contiki OS* coupled with the *uIP* stack seems eminently suitable to host the designed codes. These platforms were specifically designed for networking applications, is capable of threading and is well suited for memory constraint applications. The *protothread* library coupled with the uIP stack allows for UDP implementation over the application layer, or the *Rime stack* can al-

low for the codes to be implemented through individual modules such as *abc*, *broadcast*, *unicast*, or *stunicast*. These available SensLAB options may allow for the integration of Fountain codes. Alternatively, the designed candidate codes can be modified by using the methods and techniques provided in this thesis. The presented design framework permits this kind of scalability.

The work done under this project, can be summarised as follows:

- The identification of a suitable channel model for SensLAB.
- The design of a short message length LT code usable over the estimated channel.
- A performance comparison between two different decoding strategies for the LT code.
- The design of a short message length Raptor code usable over the estimated channel.
- A performance comparison between the two codes under the estimated channel conditions.
- A scalable message frame design framework.

The following are the supporting theoretical contributions to the project:

- The utilisation of the progressive edge growth algorithm in the H-matrix design used for the inner LDPC code for the Raptor code design.
- The utilisation of a linear programming optimisation technique for the degree distribution design used in the outer LT code for the Raptor code design.
- The utilisation of Density evolution to evaluate the designed inner LT code degree distribution for the Raptor code design.

All results were obtained within a simulated environment using Matlab. A design framework was set up wherein all the parameters can be adjusted. This enables the INRIA researchers to evaluate the expected performance of the codes while tailoring it to SensLAB applications in terms of hardware limitations, network strategies or MAC preferences. The decision to use the progressive edge growth algorithm coupled with the linear programming optimisation enables the additional scalability that allows for ample design combinations for many networking applications. Furthermore, the design of the Raptor code subsequently resulted in the design of a scalable LDPC code, which can also be considered as a standalone forward error correction strategy for SensLAB. The results for this code indicate an error correcting capability

of up to 10% erasures at high probability ($\approx 95\%$) in all three designs. Depending on the requirement for a feedback channel: the developed LDPC code will require full ARQ capability, while the developed LT and Raptor code will require limited or no ARQ.

1.6 Outline of Thesis

In this section, an overview is presented of the main topics considered in the proceeding chapters. In chapter 2, a study of the SensLAB architecture and its corresponding limitations, suitable channel models and a comprehensive evolution of Fountain codes is presented, while focusing on LT codes and Raptor codes. The main contributing theories and concepts of Fountain codes are explained, and a background of traditional coding strategies is also discussed.

The design of two Fountain codes is presented in chapter 3. A comparison between the Gaussian elimination and belief propagation decoding for the LT code is also presented. In chapter 4, the performance results of three different sized LT codes and Raptor codes are illustrated. Furthermore, the effect of various parameters related to the LT code and Raptor code designs is discussed and presented. Finally, in chapter 5, the research findings is discussed and summarised. The thesis is concluded by recommendations for future work.

Please note that some symbols and variables are duplicated in other sections in an attempt to follow standard notation. We refer the reader to the nomenclature to avoid unnecessary confusion.

Chapter 2

Literature Review

2.1 Background in Information Theory

The basis for the entire field of Information theory had its origins in the paper by Claude Shannon in 1948 entitled "A Mathematical Theory of Communication" [2]. Shannon did most of his pioneering work during the most part of the 1940's and 1950's when he published several papers that extended his original idea to the Nyquist-Shannon sampling theorem, the Shannon-Hartley theorem, and the fundamentals of theoretical cryptography. He is also known for the early development of integrated circuits, computers and Artificial Intelligence. The impact of his research on the real world is far more widespread today than it was during that time. Historically, one can look back and say Shannon created a challenge for engineers regarding digital technology.

Shannon's landmark paper benefits this work in terms of the described concepts of noise over a communication system, which is still a practical issue in modern communication systems. In the introduction section of this paper, he wrote:

"The fundamental problem of communication is that of reproducing at one point either ***exactly*** *or* ***approximately*** *a message selected at another point."*

Intuitively, a message can be transformed by the addition of extra "information" in such a way that, even if the message is corrupted by noise, there will be sufficient redundancy in it to reproduce the originally intended message. However, this brings forth two critical design considerations from both a theoretical and a practical orientation. What kind of redundancy mechanism is needed, and how much redundancy will be effective? These questions will lead towards the development of an appropriate *forward error correcting* (FEC) coding scheme capable of realising the reconstruction process of a damaged message to a confidently inferred one.

According to Shannon a FEC coding scheme can be quantified by an assigned ratio — called the information rate or coding rate. This ratio describes

the portion of meaningful information within a FEC encoded message. The qualitative question looks for actual coding schemes most effective with regard to the available device resources. In this case, the SensLAB communication systems will demand low complexity encoding and decoding algorithms that can perform practically and efficiently, at a high success rate. Therefore, it is important to identify coding schemes with the largest possible information rate, compiled with a very small probability of decoding error, and a resource efficient encoding and decoding algorithm.

In this study, a question under discussion is whether a FEC strategy, which conforms to these underlying principles, can be considered for the SensLAB platform. The focus of this work will be on a newer set of error correcting codes, which are known to be *rateless* or *self-adapting*. For such codes the coding rate may be non-deterministic and trivial. Conceptually, these codes may introduce additional protocol related features to the SensLAB network coding community.

The following sections will give a formal overview[1] of the Information theory fundamentals used to derive a channel model. The selected channel model will be explained with some important implications, which will form the basis for the rest of this research. The main theoretical premise behind such models is the concept of *Information Entropy*, since the modelling of channel characteristics is crucial in the design for a compensating decoding algorithm in the FEC code.

2.1.1 Random Variables and Entropy

The introductory section of this thesis touches on the subject of quantifiable uncertainty. Shannon wrote his master's thesis on the subject of Boolean algebra during the late 1930's, and it is considered to be "one of the most important master's theses ever written". This laid some of the groundwork for his theory on *Information Entropy* and it will also be the subject of this section.

In this section, we will formally introduce the concept of Information Entropy and its probabilistic nature with regard to channel characteristics. This concept can be explained by considering an entity, whose value is subject to random variation. We refer to this entity as a *random variable*. Consider that we have a binary string of length n bits, each of which is chosen independently, randomly deciding between 0 and 1; there would be 2^n possible combinations from which to choose. Therefore, n binary digits can indicate any one of 2^n possible outcomes. In this case, the uncertainty or entropy of the binary string is defined to be n, since $n = log_2(2^n)$, which quantifies the expected information contained in the string.

[1] It is beyond the scope of this thesis to provide complete proofs or an in-depth discussion. Only relevant concepts are explained.

More generally, we can say that information entropy is log_2(number of possible strings) or in other words, the number of "on or off" states needed, on average, to determine the outcome of an experiment [10, 11, 12].

Given a random variable X, let the possible values of X be $\{x_1, x_2, ..., x_n\}$ and set the probability of each to $p_i = Pr(X = x_i)$, for $1 \leq i \leq n$. We will only be concerned with the probabilistic domain $\{p_1, p_2, ..., p_n\}$ and not with the actual nature of X. As such, the entropy of X can be defined as follows:

$$H(X) = \sum_{i=1}^{n} p_i log_2 \frac{1}{p_i} = -\sum_{i=1}^{n} p_i log_2 p_i. \tag{2.1.1}$$

An especially interesting and important case occurs when $n = 2$, when only two possible outcomes exist for the random variable X, with probabilities denoted by p and $q = 1 - p$ for each case. This can be referred to as the *Binary entropy function* of a *Bernoulli process*. For this case the entropy can be rewritten as:

$$H(p) = H(p, 1-p) = plog_2 \frac{1}{p} + qlog_2 \frac{1}{q} = -(plog_2 p + qlog_2 q). \tag{2.1.2}$$

A graph of equation 2.1.2 is shown below in figure 2.1. In most literature this is known as the *Shannon entropy function*. This quantifies the expected "value" of the information contained in a message. Furthermore, it portrays a measure of the information received (or uncertainty removed) upon learning the outcome of an experiment or trial, which has just two possible outcomes [11].

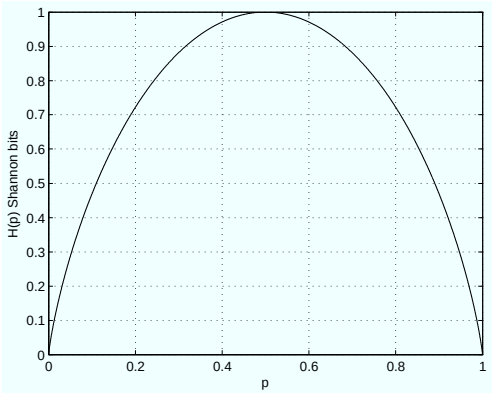

Figure 2.1: Entropy of a Bernoulli process for p and q.

The outcome is known where $H(p)$ is 0, in either case, at $p = 0$ and $p = 1$ the uncertainty is at its minimum. This implies that no information is needed to describe the outcome, since the message is certain. We only receive maximum information upon learning the outcome; when the outcome is completely random, which occurs at $p = \frac{1}{2}$, and thus requires 1 bit to communicate that message. In this case, knowledge of the outcome or message

is completely uncertain. Entropy can therefore be a measure of translated information, usually indicated by units called Shannon bits [2]. Shannon linked his source entropy theorem with his noisy-channel coding theorem to produce the channel capacity theorem, which was a ground breaking result and is also used to derive a channel model for the SensLAB.

2.1.2 Channel Models

The primary focus of this thesis is concerned with the transmission of reliable data over wireless sensor networks in the SensLAB environment. These networks represent information exchange in digital form, which means that there are only a finite number of possible symbols to characterise such information. In such networks information can be passed through multiple nodes each of which possess uniquely varying link conditions from the source to one or more recipient nodes. These links, in information theoretical terms, are communication channels that can sustain different error characteristics. The transfer of information over such settings is a physical process and is therefore, subject to natural occurring noise (resulting in signal degradation).

Many, if not all, books on Information theory explaining error-correction strategies have a similar presentation of a communication channel, as shown in figure 2.2. This topology is a basic illustration for the flow of information. The data source initiates an input messages k to be encoded into x. Ultimately, a generated message has an analogue or digital form and goes through a transmitter, which can perform some operations on the signal.

The two primary operations in the transmitter architecture are physical signal modulation and digital message encoding. Modulation[2] is the process of varying a signal in a particular way to convey a message.

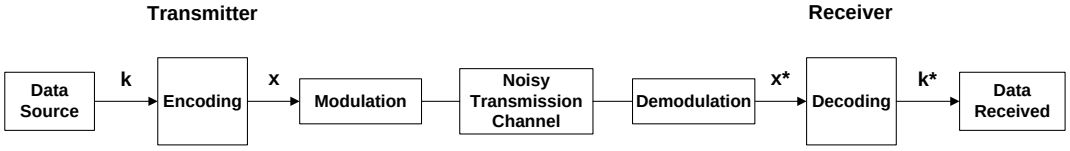

Figure 2.2: Transmission channel with simplified transmitter and receiver topology.

Real communication channels are non-deterministic, since errors might be introduced into the stream of transmitted information in the presence of random noise. Fading effects, shadowing, multipath and many other physical propagation conditions can introduce random noise in such a stream. Either signal modulation, channel coding or composite techniques are suggested to remedy the loss of transmitted information. Before considering such techniques

[2]For more information on physical layer (PHY) techniques, we refer the reader to [13, 14, 15].

it is important to consider the expected channel characteristics of the SensLAB platform. Different channel models exist, most commonly the conventional binary symmetric channel (BSC), and more recent, the binary erasure channel (BSE).

The way a channel can be described is by the set of symbols it accepts as inputs and the set of symbols it accepts as outputs, respectively called the input and output alphabet. This is denoted by A_i and A_o. Information is in the form of messages represented by binary digits. The presented channel models can each be derived from Shannon's Bernoulli entropy equation, as noted in the previous section. In accordance with the approach by Tirronen [16], the channel models can be defined as follows.

Definition 2.1: Binary symmetric channel. *A binary symmetric channel has input alphabet* $A_i = \{0, 1\}$ *and output alphabet* $A_o = \{0, 1\}$. *Transmission fails (bit flips) with probability* p, *or symbols are transmitted without error with probability* $1 - p$.

The BSC is a simply yet elegantly defined channel model used by many FEC designers, since many error characteristics can be reduced to a BSC representation. It is particularly useful to describe such error characteristics when noise is assumed to be additive white Gaussian noise (AWGN). The error probability p can be calculated by using an appropriate complementary error function, depending on the type of modulation used [10, 12, 13, 15]. For this model the probability p is also known as the crossover probability. However, one shortcoming of this model is that it assumes the errors are independent, which is however, not always the case, as the errors could occur in bursts. In addition, the value of the output symbol is utilised within *soft-decision* decoding algorithms. However, to support such decoding schemes probabilistic information is required from the radio unit.

Definition 2.2: Binary erasure channel. *A binary erasure channel has input alphabet* $A_i = \{0, 1\}$ *and output alphabet* $A_o = \{0, 1, \epsilon\}$. *A symbol is transmitted correctly with probability* $1 - p$ *or the output is erased, indicated by the symbol* ϵ, *with probability* p.

The BEC was first introduced by Elias [17] in 1955. For this channel, the symbol ϵ represents the case when a message is damaged, and the value of such a message becomes inconclusive and is discarded. Figure 2.3 depicts both the BSC and BEC models, and also shows the conditional probabilities of possible output symbols y_i given the input symbol x_i. For the BEC, it is not possible to have an inverted output (or bit flip) symbol: in the case of an error ϵ is generated as the output symbol.

In the case of the BSC, both the input and the output alphabet have the same set of variables, which may seem simpler than the BEC model, but in

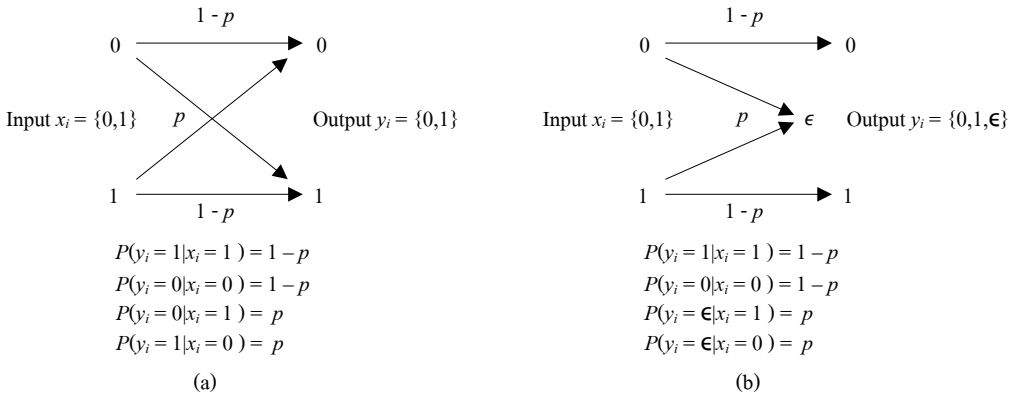

Figure 2.3: The binary symmetric channel model (a), and the binary erasure channel model (b).

fact is much more complicated. The complication becomes apparent since the exact position of the error is undetermined, and should be accounted for in other layers. Some noisy channels may also behave like erasure channels, when entire messages are dropped after a decoder has failed to recover the message [10].

The aim of this research is to develop a FEC coding scheme for a given channel model that best suits the SensLAB platform. The SensLAB radio[3] units do not allow for easily accessible probabilistic signal information, and the consensus view seems to be that soft-decision algorithms will not work well without the availability thereof. Therefore, in terms of the objectives (1 and 2), mentioned in section 1.4, we design the FEC coding strategy for particular implementation over the BEC and focus on utilising *hard-decision* decoding algorithms instead.

In the next section, we will elaborate on the Shannon channel capacity for each of the derived channel models to highlight their theoretical performance differences. This will give us an idea on how to approach the channel coding strategy and more importantly the decoding algorithm for the BEC.

2.1.3 Shannon Channel Capacity

So far we have discussed and formulated two channel models and now need to investigate the expected theoretical performance of each. A performance upper bound (channel capacity) can be obtained by using Shannon's noisy-channel coding theorem. This is an indication of the limiting information rate achievable by any attempted FEC code at arbitrarily small error probability. In operational terms, we are interested in finding ways of using the channel in such a way that all communicated bits are recovered with negligible probability

[3]More detail on the radio unit hardware architecture will follow in section 3.2.

of error at the highest possible data rate. We can use this measure as an indication of how well our designs perform.

The two presented channel models are assumed to be restricted to a *discrete memoryless* random binary source of input X over a channel Γ, with output Y. For each x in X there is an output y in Y, determined by the forward probabilities $Pr(y|x)$. If these probabilities stay unchanged over time and are independent, i.e., they correspond to independent random variables, the channel is called memoryless.

By calculating the joint probability the capacity of the channel can be defined as: Capacity $= \max Pr(X)I(X:Y)$, where $Pr(X)$ denotes the probability distribution of the input X. The channel capacity can be defined as the maximum mutual information between the input and the output. In other words, we take the maximum over all possible input probability distributions, $Pr(X)$, of the mutual information $I(X:Y)$. By applying this argument to the previously discussed channel models, the capacity for each is given as follows:

From Definition 2.1 the BSC capacity can be calculated by letting p equal the error probability. If x denotes the probability that 0 is input, $1-x$ is the probability that 1 is input. Therefore, from the entropy as calculated in equation 2.1.2 we have:

$$H(p) = -(xlog_2x + (1-x)log_2(1-x)). \tag{2.1.3}$$

Setting $q = 1-p$, the probability that a zero is output is $x(1-p)+(1-x)p = \alpha$ and the probability that a one is output is $(1-x)(1-p)+xp = \beta$. Therefore, equation 2.1.3 can be rewritten to $H(y) = H(\alpha,\beta) = -(\alpha log_2\alpha + \beta log_2\beta)$. All possible values for (X,Y) are $\{(x,y)\} = \{(0,0),(0,1),(1,0),(1,1)\}$. Then by rewriting each of the conditional probabilities we obtain: $H(X,Y) = H(x(1-p), xp, (1-x)p, (1-x)(1-p))$, and can then express the mutual information shown in equation 2.1.4, where $H(p)$ is the binary entropy function stated previously.

$$I(X,Y) = 1 + plog_2p + qlog_2q = 1 - H(p) \tag{2.1.4}$$

When $p = \frac{1}{2}$, the channel is purely random and the capacity $C = 0$. If the channel is completely reliable ($p = 0$) or completely unreliable ($p = 1$), the capacity is at maximum $C = 1$. This capacity demonstrates the upper bound to accurate communication over the BSC channel.

From Definition 2.2 the BEC capacity can be calculated similarly by using the same equations and reasoning above. It can be shown to have the linear relationship, $C = 1 - p$.

The maximum amount of information that can be transferred through the two different channels is indicated in figure 2.4 below. The complete proofs can be found in [10, 11].

The capacity of the BSC compared to the BEC in figure 2.4 indicates that, for $p > 0.5$, the BSC transfers more bits incorrectly than correctly, since the

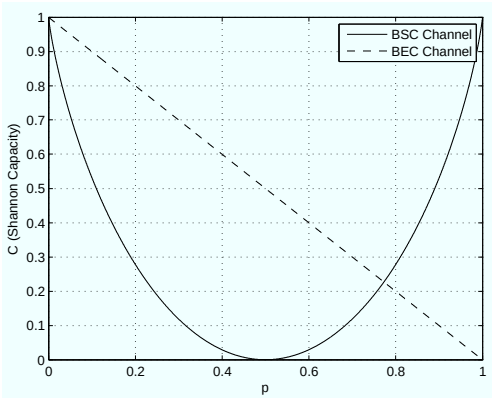

Figure 2.4: Capacities of the binary symmetric channel and binary erasure channel.

information gets inverted with probability $1 - p$. For $0 < p \leq 0.5$, the BEC capacity exceeds that of the BSC.

By careful examination it can be argued that corrupted binary information is still useful and should not be discarded (as random guessing may produce correct results half of the time), as is the case in the BEC model. As a rebuttal to this point, it can be shown by example that a soft-decision decoding scheme in a simple Hamming (7,4) code is capable of correcting up to two errors, while the same algorithm coupled with hard-decision decoding is capable of fixing only one. For such a case the channel noise can be assumed AWGN and the model a BSC model. There is no doubt that the BSC has its advantages in this regard. However, it should be coupled with an appropriate decoding scheme over lower layers supporting its output alphabet in order to realise such capabilities.

This graph gives us an idea of the expected capacity and behaviour between the two different channel models as a function of the expected error probability. In the case of the BEC, information of error positions is of great importance for the decoding process, unlike soft-decision decoders using error probability generally implemented over the BSC models. The BEC coding schemes are normally referred to as *erasure-resilient* codes. These codes and their available reliability assuring mechanisms, will be the topic of the next section.

2.1.4 Objectives of Channel Coding

There are two main strategies to consider regarding channel coding. One such strategy comprises a process that requires allocating extra bits (controlled redundancy) to the original message, which is used to reconstruct the damaged bits within it. This strategy is known as forward error correction (FEC) and is known to add a coding gain to an un-coded system (measured in bit error rate). The other popular strategy is known as automatic repeat-request (ARQ), and instead sends a resend request to the transmitter if an error is

detected. Combined FEC and ARQ solutions are normally used in practice if a feedback channel exists. Although feedback is required for reliable communication, we are particularly interested in utilising a FEC strategy without it, in an effort to preserve the sensor node resources.

With every channel we can associate a channel capacity C. This capacity defines the rate R, at which reliable communication can be obtained, such that information can be transmitted across a channel, at rates less than C, with an arbitrarily small error probability [10, 12]. Shannon's noisy-channel coding theorem can be formulated as follows:

Theorem 1: *Associated with each discrete memoryless channel, there is a non-negative number C (channel capacity) with the following property: For any $\epsilon > 0$ and $R < C$, there exists a block code of length N and rate $\geq R$ and a decoding algorithm, such that the maximal probability of block error is smaller than ϵ.*

Shannon hereby proved that the reliability of communication is not only dependent on the noise, but that reliable communication can be obtained within the capacity of a channel, by adding an appropriate channel code. From this theorem it is clear that each coding scheme can be assigned to a certain information rate, which states, in a natural way, what fraction of the transmitted information is useful.

The most important capacity approaching property of erasure-resilient codes is the number of output symbols it requires to recover the original message — where *any* portion of the encoding equal to the length of the message is sufficient to recover the message. This is called a *maximum distance separable* code (MDS) [18]. It can be defined more formally in Theorem 2.

Theorem 2: *A standard linear maximum distance separable code can be used to convert a message of length k into a transmission of length n, with a code rate R from which the message can be recovered from any portion of length greater than k. A code of dimension k is universal with respect to the decoding algorithm, if for any erasure rate $p < 1$, and for any overhead $\sigma > 0$ the given decoding algorithm can decode $k(\frac{1}{1-p} + \sigma)$ of the code with high probability* [19].

For an efficient decoding scheme, Elias showed that it is possible to decode with an exponentially small error probability using *maximum likelihood* decoding [17]. These algorithms solve systems of linear equations in polynomial time, using schemes such as *Gaussian elimination*, where the failure probability is usually a function of the overhead. Some traditional codes will be discussed in the next sections, while particular attention is given to the advent of sparse graph coding theory, where we will later test and demonstrate that it is possible to approach these limits in a computationally efficient way. To a large extent the performance of codes varies depending on the nature of the

information transfer channel. Suffice it to say, a selection between one of the presented channel models is adequate.

In terms of the objectives (1, 2 and 3), discussed in section 1.4 of this thesis, erasure-resilient codes are considered as it affords simplicity and elegance to the design, while still being a valid coding strategy over the selected channel model for the SensLAB. It was identified that the two most important complications to consider for the erasure-resilient codes in SensLAB, is the computational complexity of the encoding and decoding algorithms, and the overhead σ, necessary for successful decoding. Larger decoding overheads may contribute to increased energy consumption as it can increase symbol operations in the processor. The investigation and identification of proper FEC techniques, coupled with low complexity algorithms, will be the main focus of the following section.

2.2 Forward Error Correction

In the subsequent sections we shall explain the concepts and analysis of the different encoding and decoding principles of linear block codes and a few important traditional error correcting codes relevant to our objectives. For further reading we suggest [10, 11, 12] for a complete characterisation of earlier developed FEC schemes. The literature of Mackay [10, 20] touches on the theoretical concept of Fountain codes by using matrix representations and its corresponding operations. Concretely, the pioneers of practical Fountain codes, Luby and Shokrollahi, focus more on decoding reliability at lower computation. These are also issues concerning SensLAB.

In the proceeding sections we explain the concepts and analyse the encoding and decoding complexities of linear block codes, Low-density parity-check codes and Fountain codes, with regard to our selected channel model. We start by investigating traditional block codes, since the more advanced Fountain codes may be concatenated with them; and these block codes may be transformed into Fountain codes giving them other capabilities in SensLAB.

2.2.1 Linear Block Codes

Linear block codes are fixed length channel codes. Each block is n bits long with dimension k over a finite field $\mathbb{F}_q$, where $q = 2^l$ represents the alphabet. A $[n, k]_q$ code for short, represents a k dimensional linear subspace of the standard vector space $\mathbb{F}_q^n$. In other words, there are k information bits, and m redundant bits, where $m = n - k$. The elements of the code are called codewords, which is represented by C. Linear block codes are a special class of codes with linear dependencies between these codewords. To the code C there corresponds an *encoding map*, which is an *isomorphism* of the vector space $\mathbb{F}_q^k$ and C [21, 22]. This mapping function is used by the transmitter to

encode a vector of k elements into a codeword in $\mathbb{F}_q$. The rate of the code is defined as $r = \frac{k}{n}$, which is a measure for the amount of real information inside each codeword.

Furthermore, linear block codes can be described by their generator matrix G, and parity check matrix H. For such codes the sum of any two codewords is a codeword. Being a linear vector space, there is some basis, and all codewords can be obtained as linear combinations of the basis. The coding operations are usually done by simple matrix multiplication. All linear combinations of the rows of G must produce a codeword x, and satisfy equation 2.2.1.

$$G \times H^T = 0 \tag{2.2.1}$$

This implies that a codeword is orthogonal to each row of H. If x is a valid codeword then, if multiplied by the parity check matrix H, it must give an all-zero vector, as shown in equation 2.2.2.

$$x \times H^T = 0 \tag{2.2.2}$$

Another important property of linear block codes is the so called minimum Hamming distance d of the code, which is the minimum distance between two distinct codewords, which can be expressed as a $[n, k, d]_q$ code. In other words, it measures the minimum number of substitutions (steps) required to change one transmitted binary string into the received string.

A decoding error can be defined as the decoder in figure 2.2 selecting the wrong codeword x from the received codeword x^*. Depending on the nature of the errors inflicted on the codeword, the receiver then executes appropriate algorithms in an effort to decode the received word. Decoder failures happen when the received data differs from all valid codewords by a specified distance.

Classical block codes are generally decoded using *maximum likelihood* (ML) detection, for the BSC channel this is equivalent to finding, for a given vector of length n over $\mathbb{F}_2$, a codeword that has the smallest Hamming distance from the received codeword. Given a received vector x, the decision rule that minimises the probability of error is to find a codeword c_i, which maximises $P(c = c_i|x)$. A codeword, which is selected on the basis of maximising $P(x|c)$, is said to be selected according to the ML criterion given in the following equation.

$$P(x|c) = \prod_{i=1}^{n} P(x_i|c_i) \tag{2.2.3}$$

It has been shown that ML decoding for the BSC is NP-complete[4], and it is very likely that we would never have an efficient practical decoding algorithm regardless of the available computational resources [23, 24]. This implies a search over 2^k codewords for decoding, growing exponentially as k increases.

[4] A non-deterministic algorithm may lead to an exponential growth, since a choice between two alternatives can create multiple copies of itself.

In contrast, for linear codes ML decoding complexity on the BEC is polynomial, since it can be reduced to solving a system of linear equations. For this reason, it can be argued that the BEC channel model has great algorithmic advantage, even in traditional block coding schemes and affords favour considering SensLAB.

For a complete graphical representation, or detailed discussion on other traditional linear block codes, we refer the reader to [11, 12, 25]. The next section will consider a more practical FEC, which is popular in most communication and storage technologies.

2.2.2 Reed-Solomon Codes

We are interested in finding erasure-resilient coding schemes for the BEC channel model. A classic coding scheme fitting this criterion is called *Reed-Solomon* (RS) codes, discovered in 1960, by I. Reed and G. Solomon [26, 27]. In coding theory these codes are also known to be a MDS code, which means that an (n, k) RS code can reliably reconstruct the original k message symbols, over an alphabet of size $q = 2^l$, when any k out of n transmitted encoded symbols are received. These codes typically have a fixed code rate $\frac{k}{n}$.

The RS code has an error-correcting ability determined by its minimum distance or, equivalently, by $n - k$, which is the measure of redundancy in the block. When the error locations are not known in advance, a RS code is able to correct up to $\frac{(n-k)}{2}$ erroneous symbols.

Research in networking has suggested using implementations of RS codes for reliable data distribution. However, RS codes have the disadvantage that they are practical only for small settings of k, n, and q [3]. Although this may favour our specifications, in terms of small message length codes for SensLAB, the main disadvantage for considering this code is its high computational complexity. Standard decoding algorithms for RS codes are of the order $O(k \cdot (n-k) \cdot log(n))$. Another drawback arises if erasure rates change. We need to be able to estimate the erasure probability p prior to transmission and select rates according to additional information. This property can render the code inefficient when the expected channel conditions are discordant with the actual channel characteristics — additional channel information is not available.

RS codes use large Galois fields (GF), e.g. 2^{16}, as their input alphabets, and thereby automatically achieve a degree of burst-error tolerance [10]. It has become standard practice to concatenate these codes with others, e.g. Convolution codes, and give further protection by means of bit interleaving [15]. The concatenated RS codes used on digital compact discs are able to correct large bursts of errors.

Despite these disadvantages in terms of the SensLAB requirements, these codes are employed in a variety of commercial products. RS codes are widely used in storage, such as the compact disk (CD). Many other applications in-

clude a concatenated version of RS codes, which also finds common use in space and satellite communications, Worldwide Interoperability for Microwave Access (WiMAX) and Digital Video Broadcasting (DVB) [15, 28].

2.2.3 LDPC Codes

Low-density parity-check (LDPC) codes were first proposed in the PhD thesis of Gallager at MIT in 1962 [29]. Evidently, Gallager's work was far ahead of his time due to computational resource constraints. These codes have experienced a remarkable comeback in the last decade as demonstrated by a number of researchers including: Luby, Shokrollahi, Mitzenmacher, Spielman, Richardson, and Urbanke.

LDPC codes are block codes, which uses a parity-check matrix that contain only a very small number of binary non-zero entries (sparse matrix). In these codes it is the sparseness of the H matrix that guarantees a decoding complexity that increases linearly with the code length n. LDPC codes have excellent distance properties, which make the probability of an undetected error very small; this distance also increases linearly with the code length. Iterative decoding was originally conceived by Gallager and is also referred to as probabilistic decoding (soft-decision decoding) or codes on graphs, to identify the same area. The first efficient iterative decoder for LDPC codes was introduced by MacKay in [30]. Since then the decoding process has seen many manipulations and other optimisation techniques to obtain more efficient *message-passing* algorithms [31, 32].

A good graphical representation of LDPC codes are the linear codes represented by a *Tanner graph* in figure 2.5. The Tanner graph is associated with the H matrix and comprises of two sets of vertices. The first set is the variable nodes for the codeword (also called message nodes) of length n, and the second set corresponds to the set of check nodes m, which represents the parity-check constraints (also referred to as factor nodes). The check nodes correspond to all the parity constraints that need to be met to provide a valid codeword c_n. An edge connects a variable node to a check node if that bit is included in the corresponding parity-check equation and so the number of edges in the Tanner graph corresponds to the number of ones in the H matrix. Encoding and decoding are accomplished locally, by sending messages along the edges of the Tanner graph and these messages are then processed locally at each node.

Encoding

A codeword for transmission can be constructed by matrix multiplication from equation 2.2.4.

$$c = k \times G \tag{2.2.4}$$

The generator matrix G will most likely not be a sparse matrix in the LDPC design, and the encoding complexity can be of the order $O(N^2)$. In some cases

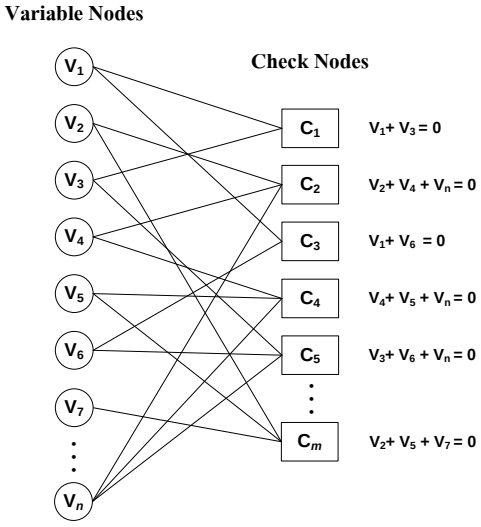

Figure 2.5: Tanner graph of a LDPC code.

a good approach for parity-check matrices is to avoid constructing G at all, and instead to encode using back substitution with H.

Decoding

Efficient decoders can be realised for LDPC codes, in the usual case a class of message-passing algorithms will be used. At each round, messages are passed on the edges from variable nodes to check nodes, and from check nodes back to variable nodes. The messages passed from the variable nodes to check nodes are computed based on the "observed" values of the variable nodes and some of the messages passed from the neighbouring check nodes to that specific variable node. The message sent from a variable node to a check node should not account to the message sent in the previous round from the check node to the variable node. This also applies to messages passed from check nodes to variable nodes. These message-passing algorithms are also known as iterative decoding algorithms and can continue back and forth until some stopping condition is reached.

In decoding algorithms, such as *bit-flipping* or *hard-decision*, the messages passed are binary (0,1) and in others, such as *belief propagation* (BP) decoding, the messages can be probabilities, which represent a level of belief about the received bits in the codeword (soft-decision). Received information is mapped to probabilities by using *log-likelihood ratios* (LLR), enabling the use of the *sum-product* (SP) algorithm. The use of the LLR allows simplified calculations at the variable and check nodes, which are reduced to sum and product operations. This is particularly useful to reduce complex multiplication operations within a processor.

In our selected channel model (the BEC channel) the erroneous messages are assumed to be completely dropped. We will only consider such channels and their supporting algorithms. For the BEC, if a parity-check equation exists, which includes only one erased bit (one unsolved variable), the correct value for the erased bit can be determined by choosing the value that ultimately satisfies the condition $H \times c^* = z$, where $z = 0$.

The biggest feature of LDPC codes in consideration for SensLAB is its simplicity in terms of sparse matrix multiplication imposing a low complexity decoding algorithm. However, the construction of an efficient H-matrix is a difficult research topic and it can produce codes that perform close to Shannon's channel coding theorem if well optimised [12, 33]. These types of code are based on irregular LDPC graphs, which are known to outperform the regular LDPC graphs on systems of practical size [34]. It is, however, important to note that the LDPC codes used in this thesis do not need to be capacity achieving. A number of pre-designed matrices are available, but all exhibit a pre-defined coding rate and selections are limited. The reason for this will become clear in section 2.3.3 and a detailed discussion will follow in chapter 3.

For the sake of discussion, LDPC codes are powerful codes, which have been adopted in many communication systems including: DVB and WiMAX [15, 28]. In the following section we will investigate new generalisations of Gallager's LDPC codes, independent of channel information and code rates, called Fountain codes. For a well-documented reference on LDPC codes we refer the reader to [35].

2.3 Digital Fountain Codes

The set of Fountain codes described in this section is one of erasure-correcting codes, presented in the work by Michael Luby et al [3, 19, 36, 37]. This work shows remarkable improvements based on LDPC codes that require either no feedback (no ARQ) or almost none depending on the specific application (with regard to SensLAB). Such codes, the most well-known being Luby Transform (LT) codes [3], were the first practical realisation of record-breaking sparse graph codes for the BEC. Recall that Elias showed that the capacity of an erasure channel is $1-p$ and that a random linear code can be used to transmit over the erasure channel at any rate $R < 1 - p$ [17].

Fountain codes are *rateless*, since the number of encoded symbols generated from the source data is potentially limitless. These codes have the advantage of not requiring *a priori* knowledge of specific channel conditions, which lends itself to application in non-deterministic networks such as SensLAB (and require no pre-configured code rate in the encoder).

The concept of Fountain coding can be explained as follows: Some data source of length M is partitioned into $k = \frac{M}{l}$ input symbols and the encoded symbols are then generated independently on the fly. The encoder is

a metaphorical fountain that produces an endless supply of water drops (encoded packets). Each water drop contains l encoded bits (from a source file of size kl bits), and any recipient will hold a glass under the fountain, collecting random drops until his glass is full. The decoder can then try and recover the source data from any set of encoded symbols N. In the usual case an aggregate, N, only slightly longer than the source data length k is necessary for successful recovery. Decoding algorithms typically considered for Fountain codes include *belief propagation* (BP) and *Gaussian elimination*[5] (GE). Like LDPC codes, the concept is well illustrated by means of a graph, as we will illustrate in the following sections.

Formally, a binary Fountain code of designed dimension k is the image of a linear map $\omega : \mathbb{F}_2^k \to \mathbb{F}_2^{\mathbb{N}}$ in which the encoded symbols are binary sequences. Here ω denotes the coding map, whereby we can identify a Fountain code by its statistical properties and thereby calculate and predict its performance. The Fountain codes discussed in this thesis have the property that their coordinates ω are created independently at random, as this induces a property of uniformity on the generated encoded symbols. These codes are governed by a *probability distribution* Ω on the vector space $\mathbb{F}_2^k$.

More precisely, let $(\Omega_1, \Omega_2, ..., \Omega_k)$ be a distribution on $\mathbb{F}_2^k$ and let $(s_1, s_2, ..., s_k)$ be the input vector, so that Ω_d denotes the probability that the value d is chosen from a *probabilistic degree distribution* (PDD) $(\Omega_1, \Omega_2, ..., \Omega_k)$. Every coordinate of the Fountain code is obtained by sampling from this distribution. The output value $(v_1, v_2, ..., v_n)$, for each encoded symbol, is the product of d corresponding source neighbours, chosen uniformly at random. We call this a Fountain code with parameters $(k, \Omega(x))$. The degree distribution can be denoted by $\Omega(x) = \sum_{d=1}^{k} \Omega_d x^d$, which is in *polynomial* form. The expectation of a distribution in this form is given by the derivative at one $\Omega'(1)$, and we will always assume that $\Omega_0 = 0$ in this thesis. This is one of two ways we will describe a degree distribution in this thesis. The encoding operation defines a bipartite sparse graph connecting encoded symbols to source symbols. These graphs are a straightforward generalisation of Tanner graphs; however, functioning in a different manner from the previous graph explaining the LDPC code [38]. Figure 2.6 illustrates a Fountain code between k input nodes on the left, and n encoding nodes on the right. Encoding nodes are also referred to as *output nodes*. Even though the block-length of a Fountain code is potentially infinite, only a prefix of the code is used in practice. The prefix length, hereafter called the overhead, depends on the channel's average erasure rate, and the type of algorithm used in the decoder. Decoding algorithms play a very important role in the overall coding efficiency and, since a small overhead is desirable, we will pay much attention to a low average length overhead. *Overhead-failure* curves are used to determine the efficiency of such codes [8].

The goal of this section is to lay down some of the theoretical foundations

[5] In the case of the erasure channel the ML decoding algorithm amounts to GE [19].

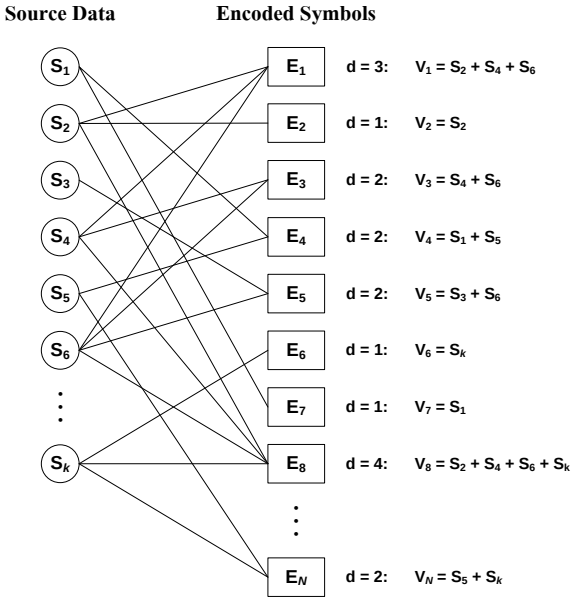

Figure 2.6: Sparse distributed representation of a Fountain code

needed for the design and analysis of a Fountain code strategy for SensLAB. The next section will investigate a very simple Fountain code, called a Random linear Fountain code. These codes, coupled with GE decoding, are known to produce low coding overheads.

2.3.1 Random Linear

In accordance with the approach by Mackay [20] the Random linear Fountain code can be explained using simple matrix notation (that demonstrates the underlying concept and decoding considerations).

Consider the encoding of a file (message) of size k, represented by source symbols $s_1, s_2, ..., s_k$. A symbol here is the elementary unit that is either transmitted intact or erased by the channel (the BEC channel). Assuming a symbol is composed of a whole number of bits, each encoded symbol e is set to the bitwise sum (modulo 2) of the source symbols for which $G_{nk} = 1$. The generator matrix G assigns a degree to each encoded symbol from a uniform distribution given by the polynomial $\Omega(x) = \frac{1}{2^k}(1+x)^k$. Each set of k random bits generated by G can be represented as a new column entry in a growing matrix by equation 2.3.1.

$$e_n = \sum_{k=1}^{K} s_k G_{kn} \tag{2.3.1}$$

If the receiver has collected N encoded packets (or N columns of length k), we would like to try to decode the message by using the inverse of matrix G, where G is defined as the random k-by-N binary matrix. In the usual case N

is slightly larger than the message length k. Recovery of the source symbols is possible only if the rank of this matrix is k. Formally, if G is invertible (modulo 2), G^{-1} can be computed by using GE, and the receiver can recover the source message vector s_k as demonstrated in equation 2.3.2.

$$s_k = \sum_{n=1}^{N} e_n G_{nk}^{-1} \tag{2.3.2}$$

The probability of successful decoding depends on the received matrix G, in which a random binary k-by-k matrix must be invertible. The probability of this happening can be determined by calculating the product of k probabilities, each of them the probability that a new column of G is linearly independent of the preceding columns, which roughly converges to 0.289 for any $k > 10$ [20]. This means that a random code for the erasure channel is not theoretically perfect when trying to recover k source symbols from a set of k received symbols. Some additional encoded symbols are required to decrease the decoding failure probability δ. Conceptually, this fundamental example illustrates the presence of coding overhead.

For larger values of k, it is shown that the decoding failure probability is upper bounded by $\delta \leq 2^{-E}$, where E is the small number of redundant encoded symbols in $N = k + E$ [20]. For $N > k$, the Random binary Fountain code has a rapidly decreasing probability of failure as a function of overhead, since we are interested in finding a k-by-k invertible matrix within a k-by-N matrix. Figure 2.7 illustrates the overhead-failure curve for $k = 100$, compared to the upper bound $\delta(E)$. The two graphs look identical for $E > 5$, and the probability of failure approaches zero after an overhead of about $\sigma \geq 0.1$. The redundancy can therefore be predicted for a given failure probability δ, where the total number of received encoded packets necessary for complete decoding is estimated to be $N \approx k + log_2 \frac{1}{\delta}$.

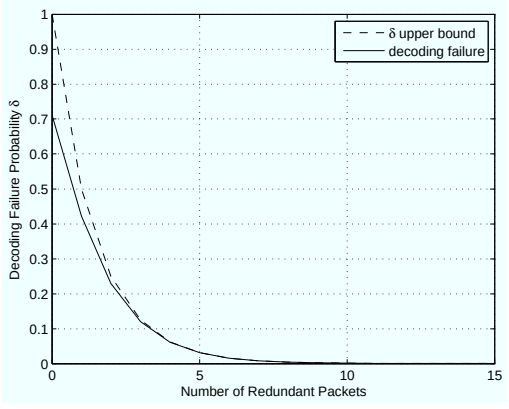

Figure 2.7: Random linear decoding performance as a function of overhead ($k = 100$).

The biggest problem with the Random linear code is the great complexity of the encoding and decoding processes. Since a uniform distribution over k is used in the encoding process, each encoded packet is expected to have an encoding cost of approximately $\frac{k}{2}$. The decoding cost of this code amounts to $O(nk^2)$ operations used in the GE decoding process. However, the theory provided suggests that these codes can get arbitrarily close to the Shannon limit as the file size k increases; the only practical problem thus far seems to be efficient encoding and decoding algorithms implementable in SensLAB. An important notion in this context is that the Random binary Fountain code can theoretically achieve a good overhead-failure curve at smaller message lengths as indicated in figure 2.7.

In summary, the performance of the Random linear Fountain code can be described by:

- The number of encoded symbols required to attain probability $1 - \delta$ of success is roughly $N = k + log_2\frac{1}{\delta}$.
- The expected encoding cost per encoded symbol, at most, is $\frac{k}{2}$ (on average half of the symbols are added up).
- Decoding cost $N \cdot k^2$ ($\approx k^3$ for large k), which is the cost of using GE decoding.
- Applying the inverse to the received matrix would have a cost approximately $\frac{k^2}{2}$.

These properties illustrate very high computational complexity in terms of encoding and decoding. We now focus our attention on the development of more efficient Fountain codes, which simultaneously improves the encoding and decoding computations, while focusing on the implications with regard to overhead.

2.3.2 Luby Transform

This section will demonstrate the first practical and efficient realisation of a linear error correcting Fountain code, published by Michael Luby in 2002 [3]. Special attention will be given to the probabilistic analysis of degree distributions created to support a particular message-passing algorithm used in the decoder. A major advantage of the Luby Transform (LT) codes is their exceedingly small encoding and decoding complexities.

Encoding

Recall from the previous section, the process of generating encoded symbols:

- Randomly choose the degree d of the encoding symbol from a degree distribution.
- Choose uniformly at random d distinct input symbols as neighbours of the encoding symbol.
- Calculate the value v of the encoding symbol, which is the exclusive-or of the d neighbours.

Decoding

In the case of the BEC, the belief propagation (BP) decoder takes on a combinatorial form, and we can consider this decoding algorithm as a sum-product algorithm, with exception, where all messages are either *completely certain* or *completely uncertain*. This particular algorithm is best described by a decoding graph, corresponding to the relationship between *source symbols* and, *any set* (fragmentary ensemble), of received symbols. The BP decoder proceeds iteratively, and recovers one source symbol at each step. Upon reception of N encoded packets, the decoding algorithm tries to decode the source message of length k. The decoding steps are illustrated in figure 2.8.

In this example, there are six source symbols and six received symbols. At the first iteration, we start at a check node with degree-one (one edge connection). The only check node connected to one variable node is E_3, and we set that source symbol S_1 to the value of E_3, since it is an exact copy of the value of the encoding symbol. We then discard the check node E_3 and add the value of S_1 to all other check nodes it is connected to (in this case E_2 and E_5). After this we can discard all edges emanating from S_1.

At the start of the second iteration, E_2 now has degree-one and we repeat the procedure until the source message is completely decoded, or no more degree-one checks are available, which implies that the decoding algorithm failed. The decoding recovery rule, hereafter referred to as the LT process, is formally described in Definition 2.3 below.

Definition 2.3 (The LT process). *All source symbols are initially unrecovered. If there is at least one output symbol with exactly one neighbour, then that neighbour can be recovered immediately and the output symbol released. The set of encoded symbols of reduced degree one after step i is called the ripple at step i. We say that an encoded symbol is released at step i if its degree is larger than one before step i, and it is equal to one after step i, so that recovery of the source symbol at step i reduces the degree of the encoded symbol to one. At each subsequent step one source symbol in the ripple is processed: The value of the recovered source symbol is exclusive-or'ed into any remaining output symbols which also have that source symbol as a neighbour and removed as a neighbour from each output symbol which has it as a neighbour. The number of edges (degree) is decreased by one to reflect this removal. If some of these*

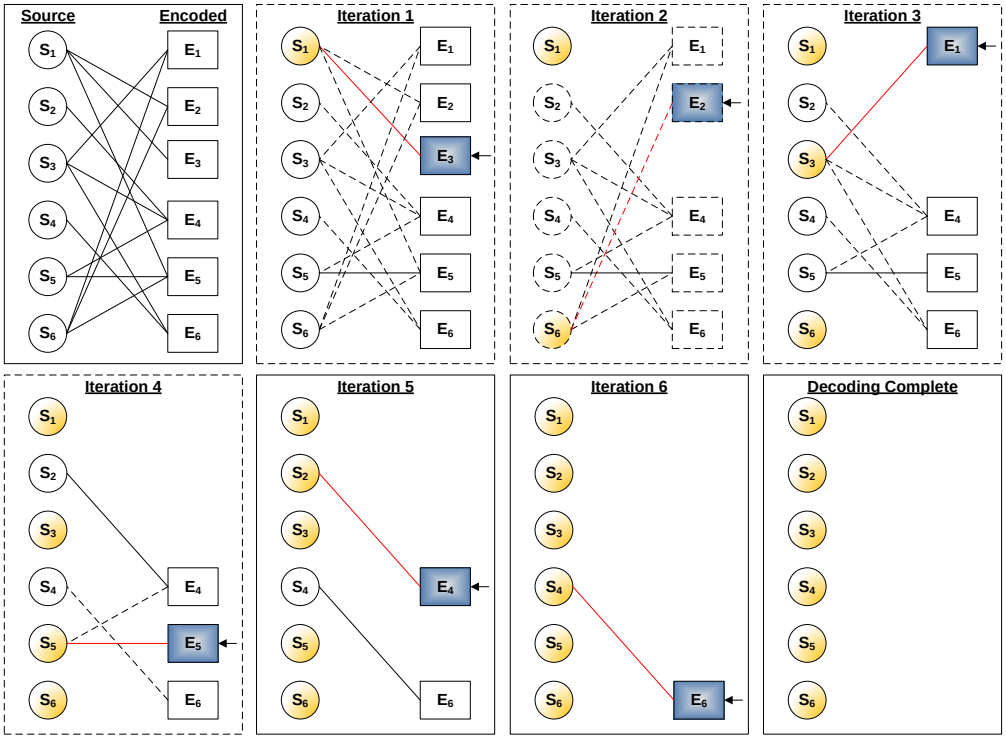

Figure 2.8: Belief propagation decoding example for LT codes ($k = 6$ and $N = 6$)

neighbours were previously unrecovered, the ripple size will grow, while other of these neighbours may already have been in the ripple, thus causing no growth in the ripple. The process ends when the ripple size is zero. The process fails if there is at least one unrecovered source symbol at the end, or succeeds if all source symbols are recovered in the end [3, 8].

Degree Distribution

Luby demonstrated a theoretically optimal solution to this problem, which can also be affiliated with the classical *balls in bins problem*[6] [3]. A high probability of collisions seems likely, when multiple balls cover the same bin (here the edges emanating from the encoded symbols represent the balls thrown to the bins, which represent source nodes) and, therefore, many more balls (more than the number of bins) must be thrown to cover all the bins. For the design of LT codes, this inevitably demands a properly designed degree distribution to ensure that the LT process releases output symbols incrementally to cover each source symbol. Conceptually, the goal of the degree distribution is to slowly

[6]The process of throwing balls into bins can be viewed as a special case of the LT process where all encoding symbols have degree-one, which are all released and thrown initially.

release output symbols as the process evolves to keep the ripple small, preventing redundant coverage, while releasing the encoding symbols fast enough to keep the ripple from disappearing prematurely before the process ends. The main objectives in the design of such probabilistic distributions are to ensure:

- Enough encoded symbols with high degree, to guarantee connections to all source symbols.
- Enough encoded symbols with low degree, to ensure that the decoding process can start, and keep going.
- The average degree of the encoding symbol should be as low as possible, to keep the number of symbol operations low.
- The number of output symbols N, necessary for successful decoding of the original message, should be as low as possible to keep the overhead factor minimal.

Ideally, to avoid redundancy, we would like the received graph to have the property of just one check node with degree-one at each iteration. In other words, source symbols are added to the ripple at the same rate as they are processed. In expectation, this behaviour can be achieved by the *Ideal soliton* distribution[7] depicted in figure 2.9 and equation 2.3.3.

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for } d = 1 \\ \frac{1}{d(d-1)} & \text{for } d = 2, 3, ..., k \end{cases} \tag{2.3.3}$$

The Ideal soliton distribution displays ideal behaviour in terms of the expected number of encoding symbols required to recover the original message data. However, it performs poorly in practice, since the slightest variation in its expected behaviour can cause the ripple to disappear prematurely. The expected behaviour seldom (almost never) matches the actual behaviour, when sampling finite times from the distribution (especially for small values of k). It is quite fragile, since only one degree-one symbol is expected from a large number of k. Despite its undesirable performance, its description and analysis captures many of the crucial elements in the design of a more robust version. Since the decoding process fails due to fluctuations around the expected value, the intuition is to increase the number of output symbols of degree-one, by slightly modifying the ideal distribution. From these points Luby proceeded to construct a very powerful degree distribution with a few modifications to the Ideal soliton distribution. It is called the *Robust soliton* distribution.

Two extra parameters were introduced to create the Robust soliton distribution. It was designed so that the expected ripple size stays roughly $\sqrt{k} \cdot ln(\frac{k}{\delta})$

[7] The inspiration for the name Soliton distribution comes from a self-reinforcing solitary wave, that maintains its shape while travelling at a constant speed. One where dispersion balances refraction perfectly.

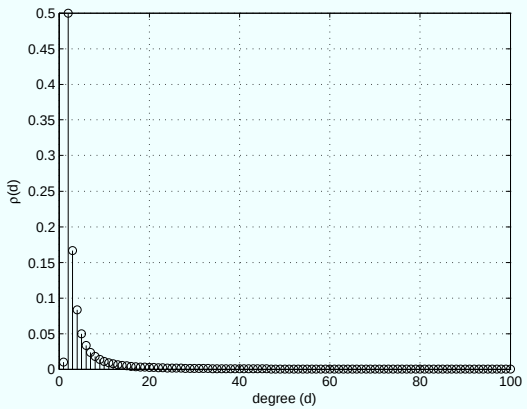

Figure 2.9: The Ideal soliton degree distribution for $k = 100$.

throughout the LT process. This is what Luby could prove at the time, based on using only BP and analytical tools, where the intuition was that the probability that a random walk of length k deviates from its mean by more than $\sqrt{k} \cdot ln(\frac{k}{\delta})$ is at most δ. This ensures that the ripple size stays large enough at each decoding step so that it never disappears completely, and that few released output symbols are redundantly covered by input symbols already in the ripple. The Luby Transform is defined by $R \equiv c \cdot \sqrt{k} \cdot ln(\frac{k}{\delta})$, where c is some suitable constant of order one, and δ reflects the decoding failure probability. The formal proof is given in [3]. The Robust soliton distribution is illustrated by equation 2.3.4 and an example is shown in figure 2.10.

$$\tau(d) = \begin{cases} \frac{R}{dk} & \text{for } d = 1, ..., \frac{k}{R} - 1 \\ \frac{R}{k} ln(\frac{R}{\delta}) & \text{for } d = \frac{k}{R} \\ 0 & \text{for } d = \frac{k}{R} + 1, ..., k \end{cases} \tag{2.3.4}$$

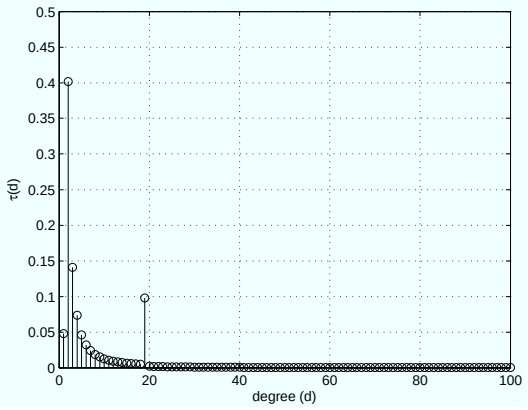

Figure 2.10: The Robust soliton degree distribution for $k = 100$, $\delta = 0.5$ and $c = 0.1$.

The Robust soliton distribution is different from its ideal counterpart in two major respects. These are the small-d end of τ, ensuring that the decoding process starts with a reasonable ripple size, and the larger spike at $d = \frac{k}{R}$ which lies at a relatively high degree in the distribution. This spike is a necessary element which helps to ensure that all source packets are connected, keeping the ripple large enough throughout each decoding step. Intuitively, the expected number of output symbols required to reach the receiver to ensure that the decoding can run to completion, has now increased to $N = k \cdot Z$, where Z is the normalising factor $\sum_{i=1}^{k} \rho(i) + \tau(i)$ between the two distributions. A complete theoretical analysis on the properties of the Robust soliton distribution is given in [3] where pessimistic estimates were used to prove that the total number of output symbols, that would suffice for complete recovery of an input message, was simplified to

$$N = k + O(\sqrt{k} \cdot ln^2(\frac{k}{\delta})), \tag{2.3.5}$$

and the average degree of an encoded symbol was shown to be approximated by $O(ln(\frac{k}{\delta}))$. Figure 2.10 illustrates an example of the Robust soliton distribution with parameters $k = 100$, $\delta = 0.5$ and $c = 0.1$.

The complexity of BP, prominent in the decoding of LT codes, is essentially the same as the complexity of the encoding algorithm, i.e. exactly one symbol operation is performed for each edge in the bipartite graph between the source symbols and the encoded symbols, during both encoding and decoding. Therefore, the computational complexity of this algorithm is linear in the average degree of the degree distribution multiplied by the size of the source block [3, 22].

In summary, the performance of the LT Fountain code can be described by:

- The number of output symbols required to have probability $1 - \delta$ of successful decoding, with respect to BP decoding, is roughly $N = K \cdot Z$, this is also given by $N = k + O(\sqrt{k} \cdot ln^2(\frac{k}{\delta}))$.
- The expected encoding cost per packet is $O(ln(\frac{k}{\delta}))$.
- The expected decoding cost is $O(k \cdot ln(\frac{k}{\delta}))$, where $N \approx k$.

Much of the previous work studying the various performance aspects of LT codes and their applications [39, 40, 41] has implicitly accepted the Robust soliton degree distribution as sufficient and optimal. This is a sound assumption from the theoretical proofs presented. However, many other studies have presented efforts to derive an optimal form of the degree distribution, decreasing the overhead and thereby reducing the total number of symbol operations required [42, 43, 44, 45, 46, 47].

So far we have introduced Fountain codes with remarkable degree distributions, and exceptionally low computational cost. These codes can exhibit

theoretical performances matching the information theoretic bounds with respect to the overhead factor σ, as we discussed in section 2.1.2 et seq. This statement, in particular, will be investigated to see if it applies to smaller message length codes.

Luby's main theorem proved that there exists a value of c such that, given N received packets, the decoding algorithm will recover the k source packets with probability $1-\delta$. This places some bounds on the probability of decoding failure as a function of reception overhead. These distributions lead to universal LT codes; however, the codes cannot have linear time encoding algorithms, since the distributions have dependencies on the data length k.

The provided theory suggests that the decoding probability is highly dependent on the selection of an appropriate degree distribution and its parameters. The decoding probability, overhead and complexity leaves possible entry for optimisation considerations. In the next section we introduce a set of Fountain codes that may provide constant encoding and decoding complexity, and more stable bounds on the decoding failure probability, which is of practical concern regarding a feedback channel in SensLAB.

2.3.3 Raptor

Soon after the realisation of the first practically implementable Fountain codes, Shokrollahi published a novel improvement in 2006, and in [22] he demonstrated the existence of Raptor codes with universal capacity-approaching performance. In this section, we will discuss a class of Raptor[8] codes, which form an extension of LT codes, with constant encoding and decoding cost. In particular, this class was considered for the SensLAB, since it exhibits a scalable design in terms of objective 6 in section 1.4.

The main contribution of Shokrollahi's work was the discovery of a weakened LT code. He proved that for any constant $\epsilon > 0$ one can construct a Raptor code for a source block (message) of length k symbols and that, on average, the number of symbol operations per generated encoded symbol is $O(log(\frac{1}{\epsilon}))$. The estimated number of symbol operations required to decode the source block then becomes $O(k \cdot log(\frac{1}{\epsilon}))$, and for an overhead of $\epsilon \cdot k$ the failure probability is bounded by $\frac{1}{k^c}$ for a constant $c > 1$, which is independent of ϵ. These codes behave similarly to LT codes, as they can also similarly produce a potentially infinite stream of encoded symbols — such that any subset of these symbols of size $(1 + \epsilon)k$ is sufficient to recover the original symbols with high probability.

Encoding

[8] Rapid-Tornado (Raptor) codes were invented by Amin Shokrollahi in 2000/2001, initially motivated by the objective of improving the encoding and decoding complexity of LT codes.

These codes are a concatenation of an inner LT code with a very high rate outer linear code, typically an LDPC code. The idea behind this Raptor code is to initially encode the source message by using a traditional code, and then to encode it using a Fountain code. In the case of LT codes, the decoding graph needs to have roughly $k \cdot log(k)$ edges in order to make sure that all input nodes are connected with high probability. Raptor coding relaxes this condition and require that only a *constant fraction* of the input symbols be covered by edges, while the unrecovered symbols are treated as erasures, and corrected by the outer code. Therefore it is important to design an appropriate outer code, able to correct a certain fraction of erasures produced by the inner LT code. A Raptor code ensemble can be represented by a triplet $(k, C, \Omega(x))$. Here C denotes a linear code (n, k) of block length n and dimension k, with $\Omega(x)$ the degree distribution generator on the set $\{1, 2, ..., n\}$. The input symbols of a Raptor code are the k symbols used to construct the codeword in C consisting of n number of *intermediate symbols*. Raptor codes also use a degree distribution, however, capped at a certain integer value d_{max} as $k \to \infty$. The encoding cost of Raptor codes can be defined as $\frac{E(C)}{k} + \Omega'(1)$, where $E(C)$ is the number of arithmetic operations sufficient for generating a codeword in C from dimension k [22].

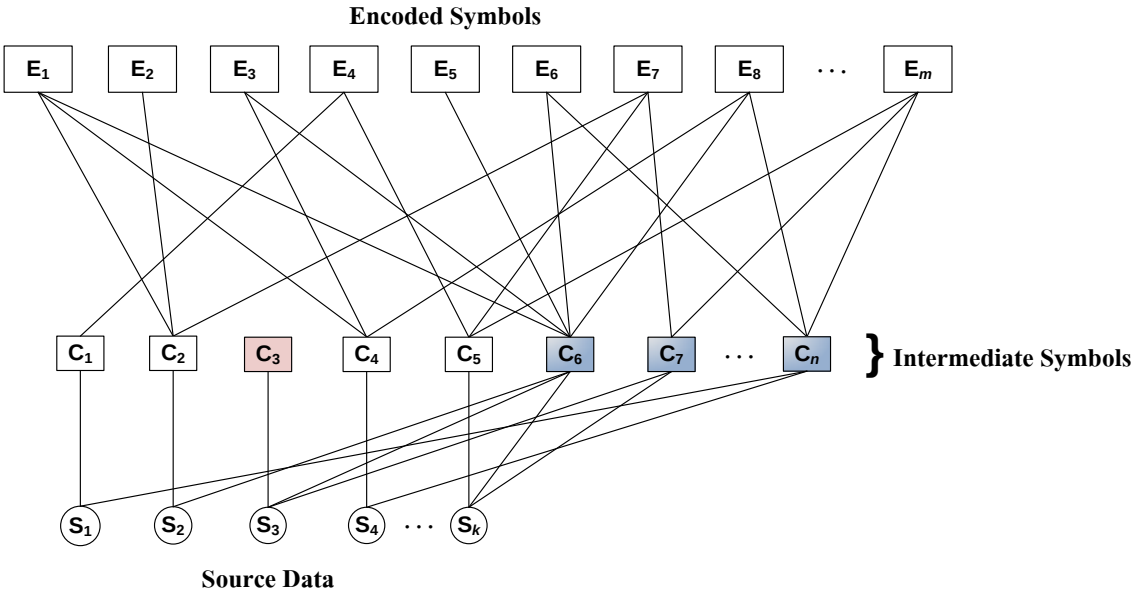

Figure 2.11: Schematic diagram of a Raptor code

Decoding

The inner LT decoder in the Raptor code can recover a certain fraction $(1-\varsigma)n$ of the outer LDPC code of length n.

The high rate LDPC code, also called the pre-code, will typically have some redundancy, depending on the code rate imposed on it. This will aim to recover the rest of the symbols after the LT decoder has terminated. The intuitive advantage of pre-coding is that the redundancy amongst the intermediate symbols allows recovery of all the intermediate symbols if most of the n are known. In some cases it may also recover from erasures induced by the channel.

A Raptor code is illustrated in figure 2.11, where the intermediate symbols are partitioned into the source symbols $C_1 - C_5$, and the constraint symbols $C_6 - C_n$ (blue). The encoded LT symbols are indicated by $E_1 - E_m$, where m encoded packets are required to recover at least $(1 - \varsigma)n$ source symbols from the inner LT code. The ϵ parameter is used here to indicate the LT code overhead, which should be designed so that decoding with an overhead of $n + \epsilon \cdot n$ is successful with high probability.

From the diagram it is clear that S_3 cannot be recovered by the LT code alone, however, it is present in the redundant pre-code symbols C_6 and C_7, and can, therefore, be recovered. These constraint symbols will hereafter be referred to as repair symbols.

Degree Distribution

The output degree distributions used in Raptor codes are fairly similar to the Soliton distribution described previously, since they follow the LT process in the decoder. However, they need to be optimised with regard to the weight on degree-one, and capped at a certain maximum degree value. Modifications to this distribution have to ensure complete decoding of $(1 - \varsigma)n$ symbols, with high probability. The general way to approach this seems to be to select a predefined degree distribution from appendix A.1. However, these distributions were designed for larger message lengths. In order to design unique degree distributions we need to understand the following Lemma. The following Lemma, which was proved in [22] from standard analytical arguments, guarantees complete decoding of $(1-\varsigma)n$ and gives the key result on developing unique degree distributions.

Lemma 1: *Suppose that LT decoding can recover at least* $(1 - \varsigma)n$ *symbols, with an overhead of* $m = (1 + \frac{\epsilon}{2})n + 1$ *received symbols, where* $\epsilon > 0$. *It follows that:*

$$\varsigma = \frac{\epsilon}{4(1+\epsilon)}, \tag{2.3.6}$$

$$d_{max} := \left\lceil \frac{4(1+\epsilon)}{\epsilon} \right\rceil, \tag{2.3.7}$$

$$\mu := \frac{\epsilon}{2} + \frac{\epsilon^2}{4}, \text{ and} \tag{2.3.8}$$

$$\Omega(x) := \frac{1}{(\mu+1)}(\mu x + \sum_{i=2}^{d_{max}} \frac{x^i}{i(i-1)} + \frac{x^{d_{max}+1}}{d_{max}}). \tag{2.3.9}$$

From the description of the LT code provided earlier, this implies that the degree distribution has to change to produce an average degree of $O(log(\frac{1}{\epsilon}))$. Each output symbol is generated using $O(log(\frac{1}{\epsilon}))$ operations, and the original symbols are recovered from those collected with $O(n \cdot log(\frac{1}{\epsilon}))$ operations. This lowers the computational cost of the LT encoding and decoding algorithms, constituent in Raptor codes, to a constant. The LT code's decoding process is therefore weakened and can recover only a certain fraction of all the message symbols.

To construct good Raptor codes, it is important to get an optimal interplay between the choice of pre-code parameters and the LT code distribution.

Raptor codes is summarised as follow:

- Raptor codes have an average encoded symbol degree of $O(log(\frac{1}{\epsilon}))$.
- The decoding complexity is of $O(n \cdot log(\frac{1}{\epsilon}))$.
- The degree distributions are designed by using linear programming optimisation.
- The finite length asymptotic behaviour is a measure used to define good distributions, and to tweak the coding parameters. This ensures decoding of the fraction ς of intermediate symbols of n with high probability.
- Raptor codes should be designed with some high rate pre-code, to ensure successful decoding with high probability. The LT decoder should then be able to decode the unconnected message symbols with any $m = (1 + \frac{\epsilon}{2})n + 1$ received symbols.

A crucial design consideration for this class of Raptor codes is the interplay between the pre-code C and the LT code. This can be considered from two sides. LT codes form a special subclass of Raptor codes, one class concerns a trivial pre-code C, i.e., $n = k$; where no redundant symbols are added to the intermediate block. Here the degree distributions are important in their design. At the other extreme, there are sophisticated pre-code only codes for which the degree distribution Ω is trivial, i.e., they assign a probability of 1 to degree-one, and 0 probability to all other degrees. The Rapor code considered for SensLAB lie somewhere between these two extremes, where a non-trivial (high rate) pre-code and a non-trivial (optimised) degree distribution needs to be utilised.

2.4 Conclusions

This chapter presented a literature study that concerns the main points discussed in section 1.4 of this thesis. An investigation of the SensLAB platform and its hardware limitations was performed. Three primary concerns regarding the implementation of an error control strategy for SensLAB was identified. These include inadequate channel information, low memory availability, modest computation and low energy consumption. A study of the fundamental theory and derivation on a channel model that can be used to accommodate these constraints was presented. There are two main arguments that can be advanced to support the selection of a BEC channel model for the SensLAB. Firstly, it is often much simpler to test coding concepts using this channel model, as the actual error positions are known and the received packets can be assumed to be error-free. Secondly, on the basis of the currently available resources, it seems fair to suggest that no individual bit error probabilities can be obtained from the sensor node radio unit. A presentation of the BSC and the BEC was presented to demonstrate the advantages and disadvantages of both, including their respective estimated theoretical limits and behaviour to random noise.

The identification of an efficient error-correction scheme for SensLAB involves a focused approach with regard to the encoding and decoding mechanism. Literature covering several FEC strategies, based on advanced iterative decoding methods, was introduced as a mechanism to increase reliability of data transmission using approximated localised solutions. These strategies were selected based on their algorithmic simplicity and computational advantages compared to other analytical approaches. Conventional methods were also explained, e.g., linear block codes, RS codes and LDPC codes. These codes use a "mapping schemes" to convert a message into a codeword, given a pre-determined code rate. A fixed channel code rate may lead to bandwidth wastage if the erasure rate is overestimated, or simply failure, when the erasure rate is underestimated in SensLAB. In addition, these implementations require feedback resources such as ARQ, additional design complexity, and their parameter restrictions make them unattractive for devices that exhibit low resources (energy). The implementation of a Fountain coding strategy was considered instead, since there is growing support for the claim that it is computationally inexpensive and resource flexible. For a traditional block code, the structure of the code is determined prior to transmission. On the other hand, in practice, Fountain codes can be generated on the fly with a "code rate" arbitrarily close to the channel's erasure probability. In addition, their rateless property may introduce other protocol related extensions for broadcasting and multicasting applications, using the existing SensLAB software. For these reasons it seems reasonable to consider a Fountain code FEC strategy for SensLAB.

The LT code was presented, as well as some important decoding algo-

rithms, e.g., GE and BP. An extended LT code, called the Raptor code, was also explained. These codes are all usable over the BEC channel model and can be classified as MDS codes (according to section 2.1.4). For MDS codes the performance depends on the overhead (number of symbols necessary for complete decoding). For this reason we need to be vigilant on large coding overheads in our design — especially at lower message sizes.

The presented literature failed to explain how such codes can be extended to a scalable variant in terms of their message length. This is a requirement for the SensLAB community, since message sizes can vary depending on the intended application or the data transfer size. Our options regarding such functionality are rather limited especially for a Raptor code implementation. One feasible option is to use an approximated algorithm that can construct H-matrices for the outer LDPC code, and an optimisation technique to construct degree distributions for the inner LT code.

The proceeding chapters will entail the design and evaluation of the candidate Fountain codes over the selected channel model. A scalable LT code with two different decoding strategies will be designed and tested. A scalable Raptor code with a unique degree distribution and a unique H-matrix will be designed and tested. The designed degree distribution and H-matrix will be evaluated separately, since it is based on heuristics outside the scope of general Fountain code literature.

Chapter 3

Design and Implementation

3.1 Introduction

In the preceding chapter an analysis of the *SensLAB* infrastructure, theory on channel models and viable *forward error correction* (FEC) strategies was presented. The *binary erasure channel* (BEC) was selected as a suitable channel model for the SensLAB and will be used as a principal element in the subsequent design. Moreover, two types of *Fountain code* were identified as possible candidates for SensLAB: a *LT code* and *Raptor code*.

This chapter will focus on the design of these two codes by considering the SensLAB resource limitations; and describe their respective design and analyses. Subsequently, the accommodating SensLAB software and chosen stack constrain the message length to a maximum of 1024 bits for an application layer implementation. Separate components in the designs will be evaluated before it is considered for integration in the final codes. A theoretical comparison will be presented with regard to the decoders, to validate their performance and characteristics.

A short message length LT code, using the *Gaussian elimination* (GE) decoder, will be designed and its theoretical upper bound will be compared to actual results in order to determine its validity for SensLAB. A short message length LT code, using the *belief propagation* (BP) decoder, will be designed and compared to its theoretical boundary conditions. These evaluations will help explain the fundamental differences between the two decoding methods and provide insight into the considerations for actual implementation over SensLAB.

A detailed description of the iterative BP algorithm is given to illustrate the importance of the *probabilistic degree distribution* (PDD). Upper and lower bounds of the parameters used in the *Robust soliton* distribution is evaluated and explained, which are important design considerations with regard to symbol overhead in SensLAB. Pseudo code is given for both the encoding and the decoding processes.

The short message length Raptor code is designed by using an appropriate *Low-density parity-check* (LDPC) pre-code and an optimised output degree distribution, for which the decoding cost could be chosen to be a constant. The conventional pre-designed degree distributions (see appendix A.1) cannot be used for this specific implementation and require a combination of design methods and testing techniques outside the scope of typical Fountain code literature. Subsequently, the LDPC code *H-matrix* is designed by using an approximated algorithm, called the *progressive edge growth* (PEG). An evaluation of the performance data from the *H*-matrices are also presented. A *Linear programming* (LP) optimisation technique is used to develop the optimised output degree distributions (the PDDs), and the result of which is evaluated. The combination of these two approaches results in a scalable Raptor code for the SensLAB community.

Three different message length codes (for the LT code and Raptor code) will be designed to illustrate the scalability of each code. The message sizes[1] will be 100, 500 and 1000. The presented decoders will be designed for each code, over these three different message sizes.

The chapter will be concluded by explaining the design steps and functions used in the simulator. Note that we are not interested in the channel information that is used in soft-decision decoding, and will therefore not use signal modulation in the simulator.

3.2 SensLAB Architecture

In this section we will examine the SensLAB node hardware and available software. We will consider the resources and limitations for both and identify those options that accommodate our FEC coding scheme. The main hardware components on the *WSN430* sensor board used in SensLAB comprise of the following:

- CC2420 2.4 GHz IEEE 802.15.4 ZigBee-ready RF transceiver
- MSP430 Ultra-low power 16-bit MCU

The exact individual bit-error probability, calculated from a received constellation (depending on the modulation[2]), is not readily available from these radio units — and estimating such values is not worth investigating considering the additional computation it may introduce. Our choice for using a BEC channel model and an appropriate erasure-resilient FEC scheme (using appropriate decoding) trivialises the necessity for such information. Concretely, it is

[1] As previously mentioned the maximum message length should not exceed 1024 bits, in order to accommodate additional header information from other routing schemes.

[2] The IEEE 802.15.4 modulation format uses Offset-quadrature phase shift keying (O-QPSK).

worth mentioning that the CC2420 can characterise the quality of a received packet from a received signal strength indication (RSSI) measured by using the available MAC software. This is referred to as a link quality indication (LQI) [48]. From the schematic it is recommended that the RSSI value is used in tandem with an average correlation value for each incoming packet to enable more accurate estimations. In the frame check sequence (FCS) the detection of corrupt data packets can be determined by using this RSSI-correlation measure with a cycle redundancy check (CRC). The LQI calculation is based on a packet error rate (PER) measure as a function of the correlation or the RSSI-correlation calculation; and limited to a value ranging from 0 - 255. The *MDMCTRL0.AUTOCRC* register should be set to enable this functionality, since the FEC scheme will use only the frames without errors. These layers are only concerned with the correctness of a frame and not the individual bit sequence itself. This is where the FEC scheme will play an important role in the upper layers.

The micro-controller unit (MCU) is an ultra-low power using a reduced instruction set computing (RISC) microprocessors that is ideal for portable applications. The MCU has 48kB ROM, 10kB RAM and the board provides an additional 1MB flash memory. The FEC code implementation may require expandable flash memory or a different MSP430 controller with more RAM. These options are available for SensLAB without compromising the layout [49]. The FEC coding scheme should be adapted to these constraints — with regard to the message length (the message length determines the vectors and matrices that determines memory allocation).

The available operating systems are:

- Contiki
- FreeRTOS
- RIOT
- TinyOS

From the available operating systems *Contiki* and *RIOT* seems like feasible candidates to host the FEC strategy. These two have seen many TCP, UDP and mesh network protocol implementations that can provide application layer capability for the implementation of the FEC scheme. In particular, the UDP capability seems useful for the Fountain code FEC scheme, since no feedback (no ARQ handshaking) is required. In addition, an extra FEC layer implementation in the Contiki stack will benefit the SensLAB community. More details can be found at [50] where an illustration of Contiki's protocol stack is explained.

Contiki benefits our design, since it was primarily developed for networking applications and is suited for memory constraint micro-controllers, it has an

event-driven thread-like multitasking (using the *protothread* library) functionality that can be used for running the decoding algorithm separately. Contiki also provides the *Cooja* simulator on which the FEC code can be run and tested. Furthermore, it provides a lightweight TCP/IP stack called *uIP* where it implements RFC-compliant TCP and UDP that is compatible with IPv4 and IPv6 implementations. Additional functionality of lower layers of the IPv4 stack enables a mesh-under configuration by using the *Rime* communication stack. Rime can provide mesh routing and route discovery, since the uIP stack uses it to forward packets on the network. Many of these extensions can be used in combination with the designed FEC scheme to develop powerful and feature-full applications for SensLAB.

The uIP stack has been ported and tested successfully on the WSN430 platform and as such it seems like a reasonable transport layer mechanism to use for the implementation of the FEC scheme. The only constraint is the maximum message size of 128 octets for UDP communication [50]. For this reason the FEC design should be limited to 1024 bits — even less than 1024 bits when considering an implementation over the application layer in combination with other routing layers.

3.3 LT Code Design

This section will describe the LT code encoder and decoder design in Matlab. Practical implementation considerations regarding SensLAB will also be discussed. We begin by looking at each of the components inside the encoder and decoder. In figure 3.1 an implementation of the LT code using BP decoding is depicted. When constructing the encoded packet E, two different framing methods can be used depending on the application requirements and the available protocol's packetisation payload length. In this case, we consider a comprehensible visual representation that explains the encoding and decoding process. Each encoded packet is constructed by including the degree value (number of connected neighbours) and all the neighbouring symbol values (actual values). For a practical implementation over SensLAB it is advised to replace these values with a random number generator's seed. Such a configuration will allow for reduced packetised information, since there is a possibility that the PDD will generate a large degree equal or close to the message length k, which might be larger than the maximum transport packet length — this may worsen if a larger Galois field is implemented. In this design the LT code is written for illustrative purposes with memory allocation as shown in the diagram.

Our encoder encodes E as shown in this figure, where each packet can be represented by a vector $(1 \times (k + 3))$. The encoded packet consists of:

- i — the packet number,

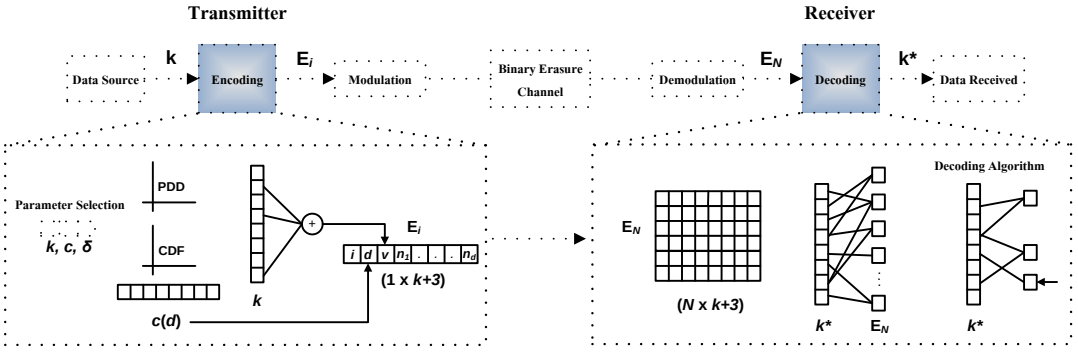

Figure 3.1: Transmission channel with a LT code implementation.

- d — the packet degree (number of connected neighbours),
- v — the packet value (sum of connected neighbour values),
- n — the packet neighbours (connected neighbour indices).

The packet degree d is randomly selected from the cumulative distribution $c(d)$, which is obtained from the PDD. This is necessary to enable the use of only one random number generator. A d number of neighbours n are randomly selected from a uniform distribution and added to the packet. The value v is calculated as the exclusive-or of the d number of neighbours. The PDD is constructed with the following hyperparameters:

- k — length of the message,
- c — free parameter,
- δ — decoding failure probability.

In the decoder, the received encoded packets is stored in a matrix $(N \times (k + 3))$. If the random seed implementation is considered for SensLAB, it is important that the decoder has the same PDD parameters as the encoder, e.g., c, δ, and k. The encoding process is demonstrated in Algorithm 1.

The receiver collects encoded packets in any order and constructs a random sparse graph for the decoder to decode [3]. The receiver can wait for a certain number of encoded packets N before attempting to decode the message k^*, or attempt decoding upon reception of a degree-one packet. The latter may improve memory availability in the sensor node hardware, since the decoded packets (released symbols) can be freed in memory.

The diagram in figure 3.2 demonstrates a basic flow of the encoding and decoding components in the LT code simulator. This implementation of the

[3] It is important to realise that it is not necessary for the received bipartite graph to be an exact reconstruction of a particular set of transmitted encoded symbols, as is the usual case for LDPC codes with fixed code rates.

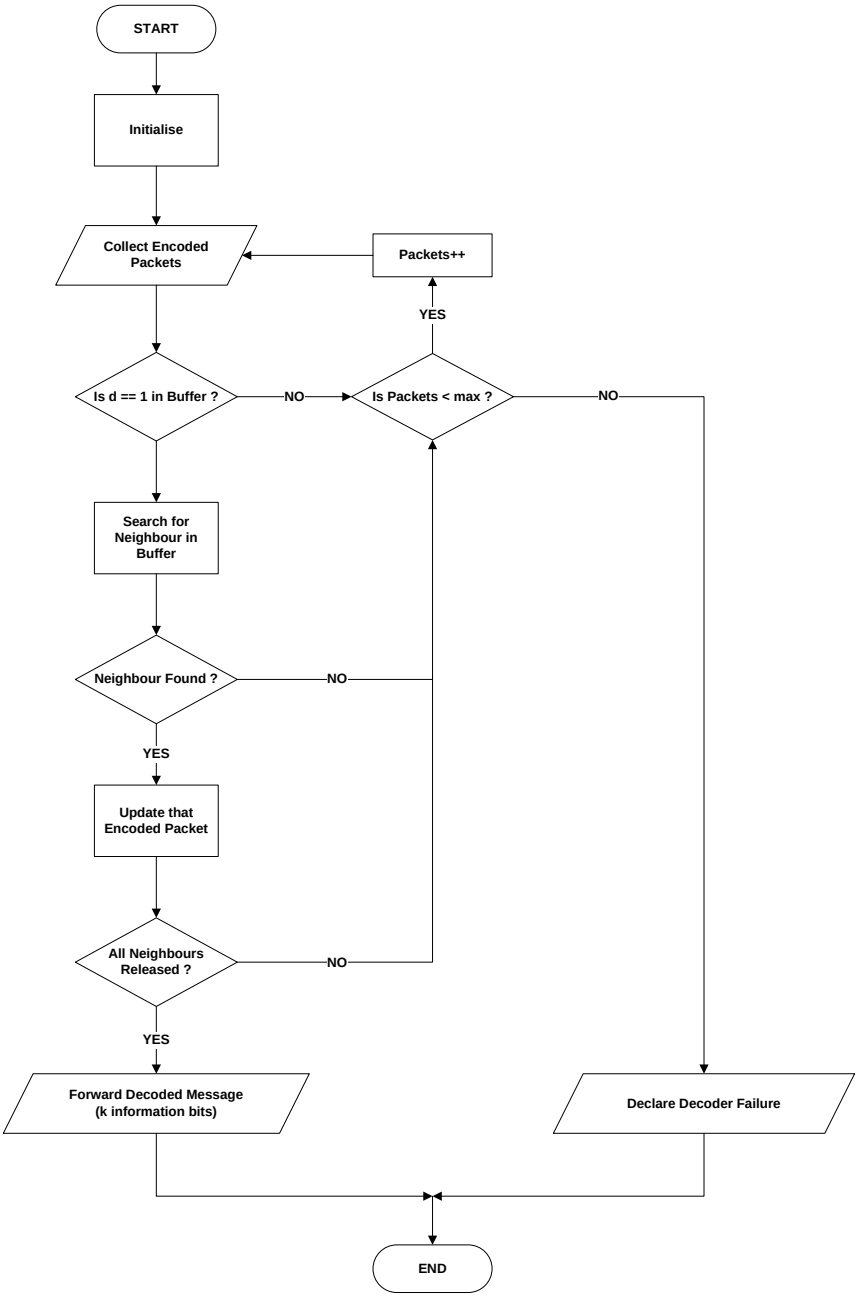

Figure 3.2: Top-level design flow of the iterative belief propagation message passing decoder.

decoder allows for an easier evaluation of decoding success. The evaluation of decoding success as a function of packet overheads can also be realised by removing the maximum packets reached test condition. We are particularly interested in these evaluations for SensLAB with the focus on overhead.

The decoder configuration may change for actual implementation and is highly dependent on the particular SensLAB application requirements. We

Algorithm 1 LT Encoding

```
Select PDD parameters
Calculate CPD
repeat
    choose degree d from PDD p(d)
    choose uniformly at random d input symbols n(i_1), ..., n(i_d)
    for all i do
        v ← v ⊕ n_i update value
    end for
    send packet
until stop bit received or maximum N sent
```

are interested in the decoder's performance in order to establish its expected packet overheads, which this configuration allows. The receiver collects a predetermined number of encoded packets N before attempting to decode the message, this enables us to evaluate the decoding success rate. The decoding probability at a specified overhead will be an important evaluation for a scalable version in SensLAB. The overhead-failure behaviour implies that: for a given value of k, the probability of decoding failure is independent of the reception index sequence, and depends only on the received number of encoded symbols N. The probability of failure is a function of the overhead, and from the theory we expect the failure probability to decrease quickly with an increase in the received symbols N. How this affects short message length versions of LT codes will be of primary concern in the results section.

In terms of scalability, the LT code message length can be adjusted by changing k. Subsequently, the length of the degree distribution and cumulative distribution should also be adjusted to this value. The two decoding strategies will be presented in the following sections.

3.3.1 Belief Propagation Decoding

The theory discussed in section 2.3 seems to suggest that the BP decoding algorithm can provide a computationally inexpensive decoding solution for the LT code over the BEC channel. This section describes how it is used for the SensLAB. In particular, we design the BP decoder to evaluate its overhead performance for short message length implementations.

In Bayesian network terms BP decoding is performed as "inference" on the "received nodes", and the algorithm used is a class of the sum-product version of the BP algorithm used in Bayesian networks [1]. This particular decoder utilises the BP algorithm by "carrying" local messages between nodes to solve a complex global function, which requires only simple processing compared to other techniques. Effectively, complex analytical methods used to solve such sets of equations can be replaced by iterative methods (converging algorithms)

such as the BP algorithm.

The objective of the decoder is to recover the original source data, s, from $s = e \cdot G^{-1}$ as described in section 2.1.1. Some decoding methods such as using matrix inversion will bring a high computational cost especially when the generator matrix G is large. Decoding LT codes over the BEC can be done by using a special BP process in which a received symbol has only two states, either completely corrupt or completely correct.

This particular variant of the BP algorithm deals with the optimisation of a rather complicated global function of a large number of variables (the matrix shown in figure 3.1). By solving subsets of the unknown variables the construction of the decoding algorithm becomes much easier to manage and, therefore, less computationally expensive. This factorisation process can be represented by a bipartite graph as discussed in section 2.3.

Figure 3.3 demonstrates the BP decoding steps for a binary input $s = [1\ 1\ 0\ 1\ 0\ 1]$, to explain how the BP algorithm is used in the design for SensLAB. Initially, the decoder was designed at small scale to validate its functionality. In this example it is represented by a bipartite graph between $k = 6$ source symbols on the left, and $N = 6$ received encoded symbols on the right. Encoded symbol E_i is connected to source symbol S_j. The BP decoder

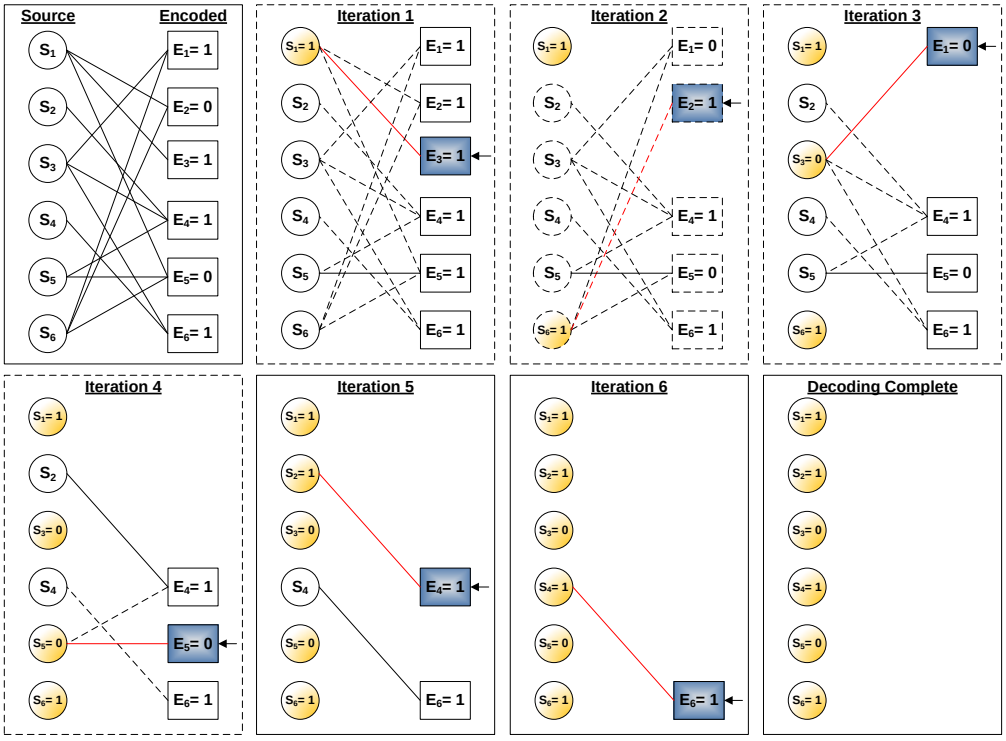

Figure 3.3: Toy example of LT belief propagation decoding for $k = 6$ and $N = 6$.

repeats the following until failure occurs in Step (1), or the decoder stops after successful decoding in Step (4):

- Step 1: Find an encoded symbol, say with index i, of degree 1; let j be the index of its unique neighbour among the source symbols. If there is no such degree 1 encoded symbol, then decoding fails prematurely.
- Step 2: Decode by setting $S_j = E_i$.
- Step 3: Let $i_1, ..., i_l$ denote the indices of encoded symbols connected to source symbol j; set $E_{i_s} = E_{i_s} + S_j$ for $s = 1, ..., l$, and remove source symbol j and all edges emanating from it.
- Step 4: If there are unrecovered source symbols, then goto Step (1). Else STOP.

If this process is repeated k times without failure then decoding completes with all k source symbols recovered. Essentially, the complexity of the BP decoding algorithm is the same as the complexity of the encoding algorithm, since exactly one symbol operation is performed for each edge (connection) between the source symbols and the encoded symbols. The computational complexity of the BP decoding is linear in the average degree of the PDD multiplied by the size of the source block k. This emphasises the importance of the PDD in a SensLAB implementation. In Algorithm 2 the pseudo code for the designed BP decoder is presented.

Algorithm 2 Belief Propagation LT Decoding

```
1: repeat
2:   if d == 1 then
3:     S_j ← E_i
4:     for all j in buffer do
5:       d ← d − 1 reduce degree
6:       v ← v ⊕ S_j update value
7:     end for
8:   end if
9: until all input symbols recovered or decoding failed
```

The evaluation of this algorithm should represent the behaviour of a metaphorical fountain producing water-drops, and a glass under this fountain catching any random drops should be usable when the glass is full. This concept can be tested by evaluating the BP decoder's decoding behaviour on processed encoded symbols. The typical behaviour of the LT code using the Robust soliton degree distribution is illustrated in figure 3.4 (a) - (d) for a message length $k = 100$ on the left, and (e) - (h) for $k = 1000$ on the right. The Robust soliton distribution with its particular parameter selection used in the

encoding process is shown in (a) and (e). The decoding failure probability in both cases is set to $\delta = 0.5$ and the free parameter $c = 0.1$. The cumulative distribution $c(d)$ is also indicated, as it is from this graph that the degrees are independently sampled. This is shown in (b) and (f). A histogram of the actual received degree distribution $r(d)$ is also provided in (c) and (g) to compare with the expected degree distributions given above.

When comparing the shape (density distribution) of the actual distribution $r(d)$ to the expected distribution $p(d)$, it is clear that many of the weights fluctuate and do not represent the exact distribution. This is an inherent problem of LT codes using a combination of BP decoding and a PDD. Furthermore, this is what Luby identified as the root cause of the Ideal soliton distribution's failure in practice (see section 2.3.2). More on this paradox and its related design considerations will be discussed in section 3.4 and chapter 4.

When comparing (g) to (e) we see a smoother resemblance, since the sampling set is much larger. This also explains why the LT code becomes more efficient for larger message lengths k. Finally, the fountain analogy is illustrated in (d) and (h), where complete decoding is possible if enough packets are received. The decoder is run greedily as encoded packets arrive and immediately starts to decode upon reception of the first degree-one packet — it decodes all k only when a sufficient number of packets N is received. The vertical axis shows the number of packets decoded as a function of the number of received packets.

This graph illustrates the conceptual characteristic of a true Fountain code and validates the functioning of our designed BP decoder. It also provides insight into the complexity (critical energy usage distribution) of the type of decoding algorithm typically required for implementation on SensLAB. When ample independent packets are received, most of the symbol operations will occur at the end of the decoding process. This is clearly illustrated in the figure.

It is clear that for BP decoding N must be greater than k to ensure successful decoding of the original message. Since it is a random code, the total number of packets required for successful decoding will seldom be the same for all transmitted messages. The expected number of encoded packets required for successful decoding can be estimated with equation 2.3.5.

The preliminary evaluations presented in this section demonstrates that the overhead[4] σ decreases substantially with the increase of the message length k. In the next section we will discuss the importance of the PDD and demonstrate the influence of the two parameters c and δ.

[4]Recall from Theorem 2 in section 2.1.4 that the overhead fraction is represented by σ.

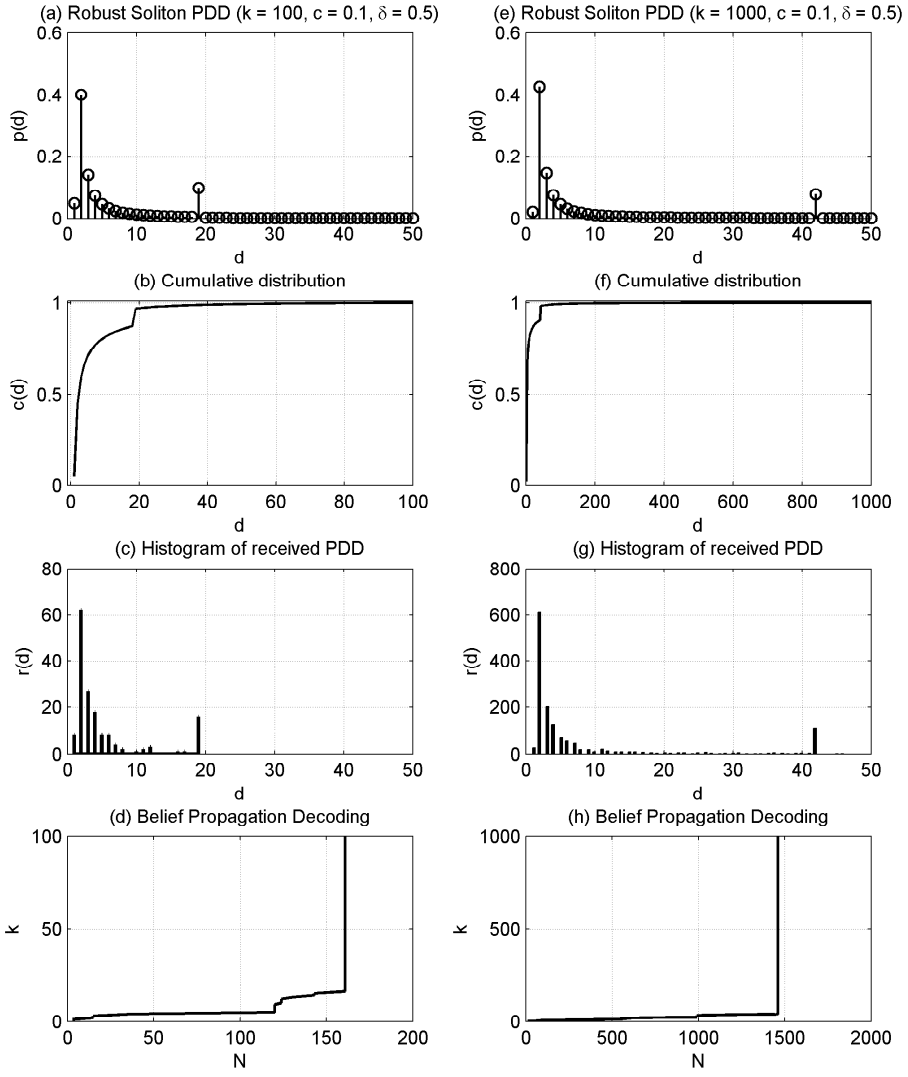

Figure 3.4: Typical LT code performance for the Robust soliton distribution with $c = 0.1$ and $\delta = 0.5$.

3.3.2 Degree Distribution

The PDD often used in the LT code is the Robust soliton distribution, and was used to establish the first practical realisation of a true Fountain code. In the last decade, research on LT codes has provided many different derivations on the design of an optimal PDD, in an effort to reduce packet overheads. However, these results provided no confirmatory evidence of significant reductions; and it is worth mentioning that many attempts have adapted the Robust soliton. As such, it is decided to use the Robust soliton PDD for this design. Subsequently, it is necessary to analyse the theoretical constructs of this PDD to find the optimal parameter pair for the implementation in SensLAB. Concerns for SensLAB implementation regarding this degree distribution are as follows:

- There might not be any encoded packets of degree-one during the decoding process, which will lead to decoding failure.
- There might be too many encoded packets of degree-one, at some intermediate step, resulting in large packet overheads.
- Some source symbols might not be connected if the average degree is too small, also resulting in decoding failure.

The correct interplay between the parameters in the Robust soliton PDD, from equation 2.3.4, is crucial for the LT code design, since it affects the decoding failure probability and the packet redundancy. Bounds on these parameters are given in equation 3.3.1 to help with the design of an optimal LT code for SensLAB. When selecting parameters beyond these constraints, the high-end spike of the degree distribution at $\frac{R}{S}$ will move outside the maximum degree length (greater than $d = k$) and will give out-of-bounds errors. Bounds on the parameter c was first proposed in [51] and presents a novel criterion in the parameter selection process.

$$\frac{1}{k-1} \cdot \frac{\sqrt{k}}{ln(\frac{k}{\delta})} \leq c \leq \frac{1}{2} \cdot \frac{\sqrt{k}}{ln(\frac{k}{\delta})} \tag{3.3.1}$$

Theoretical bounds of the Robust soliton is shown in figure 3.5. The decoding failure probability curve δ is clearly indicated, presenting the expected area within which the decoder may operate (given in equation 2.3.4). The left column (a), (d), and (g) illustrate the expected number of degree-one packets. The expected number of higher degree packets, at the spike $\frac{k}{R}$, is indicated in the middle column (b), (e), and (h). In the right column the expected number of encoded packets is shown, indicating the estimated redundancy.

The top row (a) - (c) shows the expected behaviour for smaller data lengths of $k = 100$. In each row k is increased. This gives us an idea of how the decoder will perform when utilising the BP algorithm with the Robust soliton PDD.

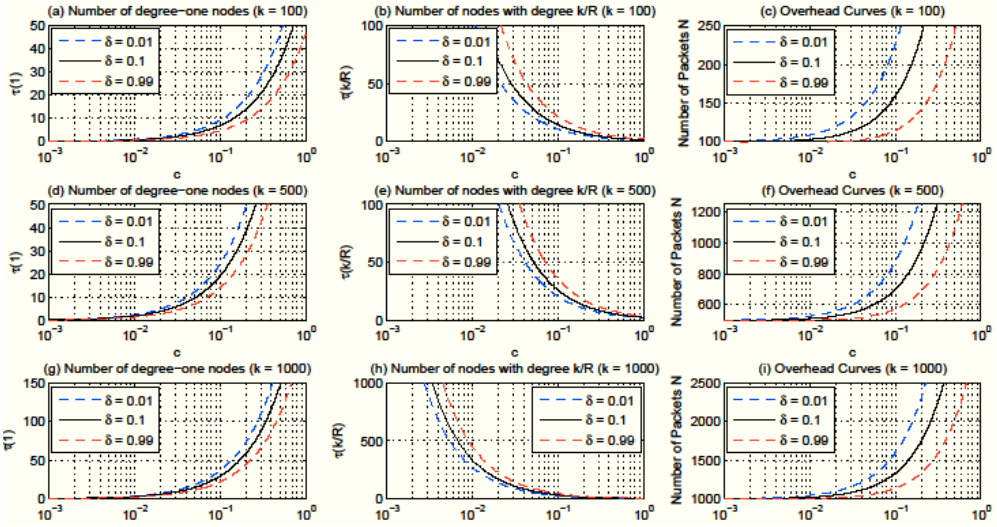

Figure 3.5: Expected performance for LT codes using the Robust soliton distribution with the two parameters c and δ.

In accordance to [52], it was shown that the optimal parameter selection for c is ≈ 0.1 in all cases where $k = 100, 500$ and 1000. Furthermore, it was indicated that the decoding failure probability can increase to a value greater than 1. We will focus on the supporting theory and stick with standard notation provided by the original developers of Fountain codes, and as such, not exceed parameter values unsupported by their respective theoretical limits ($\delta \in [0, 1]$).

The presented parameter boundaries provide the necessary information on how the Robust soliton PDD is expected to provide ample degree-one and higher degree symbols for the BP decoder. The bounds on decoding failure provide the first insight into the estimated number of symbols necessary for successful decoding. We will be using a conservative value, $c = 0.1$ (this value also falls within the boundaries of c for each of the three message lengths), for our designs and investigate the decoding probability, δ, in the results section. The optimal parameter pairs, for each message length, can be calculated using statistical methods, which is beyond the scope of this thesis. The presented parameter pairs are very close to optimal compared to other research and is feasible enough to consider for SensLAB.

The only concern with using the BP decoder is its large overheads at smaller message sizes. For this reason GE decoding was considered for smaller message sizes in SensLAB. The following section will discuss such a design.

3.3.3 Gaussian Elimination Decoding

Another familiar decoding technique used in Fountain codes, and one of the first to demonstrate the proof of concept, is Gaussian elimination (GE) de-

coding. The Gaussian decoder affords consideration with regard to the implementation in SensLAB, since it may provide low packet overheads, according to the literature.

In principle the Gaussian elimination decoding technique attempts to solve a set of linear equations (independent) produced by a random binary matrix. In this case the PDD is uniform and require no pre-calculation, since it is non-parametric (except for the variability of the message length k).

To demonstrate the performance of this method, it is necessary to investigate the invertibility probability of such random binary matrices, as explained and motivated in section 2.3.1. In order for the decoder to decode the original source message, it must be able to compute the inverse of the generator matrix (G^{-1}). From the literature it is known that the invertibility probability of random n-by-n binary matrices goes to 1 as $n \to \infty$ [53]. Furthermore, the theory shows that the invertibility probability monotonically increases as the size of the binary matrix increases (when it is augmented by the addition of more column entries, resulting in a matrix with dimensions k-by-N). The upper bound provided in figure 2.7 showed that this conjecture is valid.

A simple test was performed by using the BEC model (with random erasures) in order to validate these claims. By simulation, theoretical results are compared to actual results to investigate the GE decoding behaviour and performance for SensLAB. Random binary matrices were constructed and tested to see whether they could deliver a full rank of length k, which must be the same length as the message length (or the row length). An extra random binary column was added each time and the invertibility probability re-calculated. These tests produced overhead-failure curves, which can be compared to the more common BP decoder. The GE decoding test is illustrated in Algorithm 3. The number of frames represents the number of times the experiment is repeated.

Figure 3.6 illustrates the actual decoding compared with the expected decoding probability, and indicates the corresponding correlation. For this simulation the average success rate was calculated from a thousand random matrices for each point on the graph. It is clear that an overhead of $\sigma = 0.10$ ensures a very high decoding probability close to 1.

In figure 3.7 and figure 3.8 the same test was performed. For these message lengths the average success rate, indicated by each point, was calculated using fewer random sets of matrices (in an effort to reduce computation time). In each graph the actual performance is well anticipated by the theoretical upper bound and for larger message lengths the overhead percentage drops significantly. The relationship is strictly monotonic and not linear, so a Spearman correlation measure is used to verify the correlation (ρ close to 1 indicates strong positive correlation). The result of this measure is indicated in table 3.1. Despite the promising low overheads and the strong correlation to the theoretical upper bounds, GE is computationally expensive and not recommended for larger message lengths in SensLAB. It can, however, be considered for

Algorithm 3 Gaussian elimination decoding

```
for n = k to N do
  for i = 1 to frames do
    create random binary G matrix
    create random binary s message
    encode the message e = G × s
    calculate matrix rank r
    if r == k then
      calculate G^-1
      decode message e* = G^-1 × e
    else
      message cannot be decoded
    end if
  end for
  if e* == s then
    message successfully decoded
  end if
end for
```

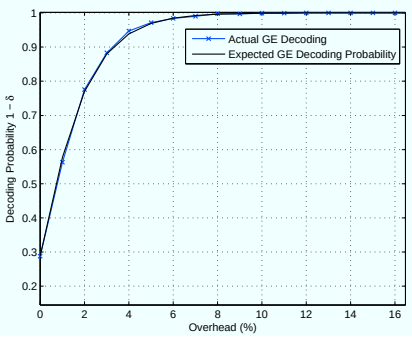

Figure 3.6: Actual Gaussian elimination decoding compared to the expected theoretical bound ($k = 100$).

smaller size messages ($k \leq 100$). In such cases the *math.h* library can be used, since it is compatible with the Contiki OS. This will enable the computation and manipulation of linear algebraic equations such as Gaussian elimination. The results section will elaborate on the performance of the different decoding overhead-failure results. Moreover, the BP decoding performance can be considered as an near optimal result for Fountain codes in terms of overhead performance.

The presented GE overhead-failure performance indicate a high correlation to the actual random matrix tests, and can therefore be used as a near-optimal performance indication (baseline indication with regard to redundancy) compared to the other designed Fountain codes.

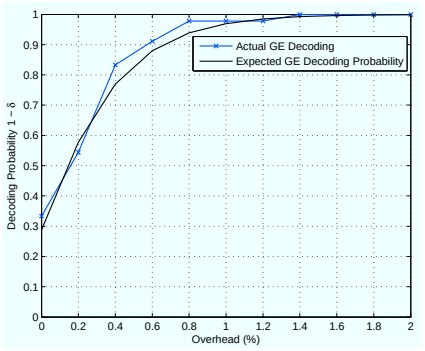

Figure 3.7: Actual Gaussian elimination decoding compared to the expected theoretical bound ($k = 500$).

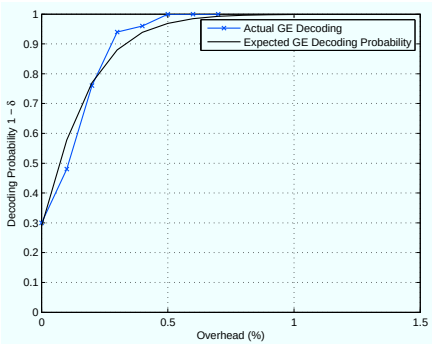

Figure 3.8: Actual Gaussian elimination decoding compared to the expected theoretical bound ($k = 1000$).

Table 3.1: Summarised evaluation of GE decoding.

k	100	500	1000
ρ	0.9989	0.9958	0.9429

3.4 Raptor Code Design

This section will entail the complete design of a scalable Raptor code that can be considered for SensLAB. This particular Raptor code consists of an outer pre-code and an inner LT code. The two most important design considerations for this Raptor code is the selection of an appropriate pre-code, and the design of an complimentary degree distribution such that their combination will allow for lower overall complexity. As mentioned previously in section 2.3.3, we consider a Raptor code that lie somewhere between a high-rate pre-code and an optimised PDD. As such both should be considered non-trivial.

In the last decade, research has provided ample support for the assertion

that LDPC codes are efficient as they can perform close to the Shannon limit. Moreover, the consensus view seems to be that a set of LDPC codes that uses an *irregular H-matrices* can outperform regular ones. For these reasons a LDPC code, which utilises an irregular H-matrix was considered for this design.

Typical LDPC codes propagate probabilistic channel information in the decoding algorithm. As such, our LDPC code needs to be modified to accommodate a BEC implementation for SensLAB. The design methodology for an appropriate erasure-resilient LDPC encoder and decoder and their parameter selections will be illustrated. Additionally, an equally important consideration is the design of an effective Raptor degree distribution capable of elegant interplay between erasure delivery and erasure correction. The correct interplay between all these parameters and the selection of an optimal degree distribution is what determines the performance of the Raptor code. A diagram of the different Raptor code components are shown in figure 3.9. In this illustration,

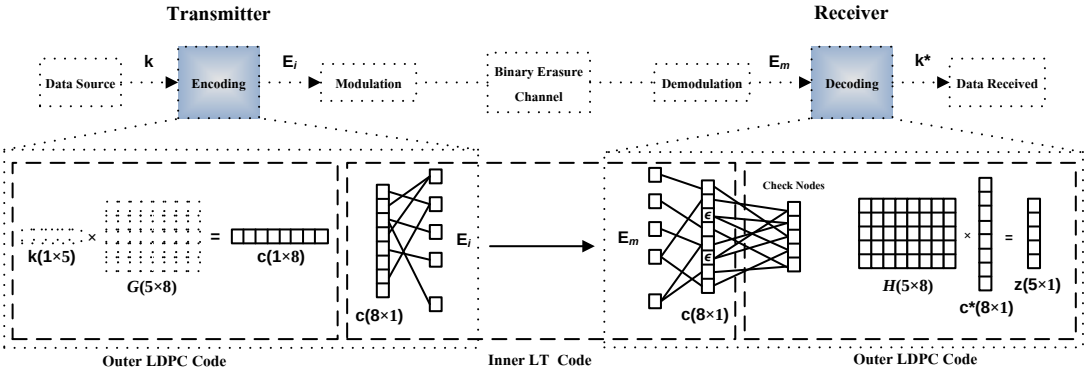

Figure 3.9: Transmission channel demonstrating a Raptor code implementation.

the message k (of length[5] 5) is encoded by multiplication of the G-matrix (the G-matrix is derived from the H-matrix not shown in the figure). This produces the encoded message c (with 3 extra bits). The pre-coded message c is then encoded by the LT code and transmitted as E. The BP decoder attempts to reconstruct E (with missing symbols) and then proceeds to decode by using the LDPC decoder to produce c^*. The final decoded message is presented by z.

In order to proceed with the design we need to evaluate the conditions for designing a Raptor code. We will use the LT code described in section 2.3.2, and a suitable pre-code described in section 2.2.3. The conditions for these codes are:

- The code rate R of the LDPC pre-code C_n needs to be $\frac{1+\frac{\epsilon}{2}}{1+\epsilon}$.

[5]Note that these values are for illustrative purposes only.

- The LT code should be able to decode the disconnected fraction of symbols given by: $\varsigma = \frac{(\frac{\epsilon}{4})}{(1+\epsilon)} = \frac{(1-R)}{2}$.

The following two sections will focus on the specific parameter selections and the evaluation of each component in the Raptor code, i.e, the LDPC pre-code and the LT code.

3.4.1 LDPC Pre-code

In figure 3.9 the Raptor code implementation is illustrated. Firstly, the source message is encoded by the LDPC code using the generator matrix. This converts the original message to a codeword of length n. Our G-matrix construction will allow for a systematic pre-coding of c. This means that the original message k is part of c. Secondly, this codeword is passed to the LT encoder and encoded as illustrated in figure 3.1. The LT decoder collects enough symbols and tries to decode the fraction of connected symbols and sends the result (with a fraction of un-decoded symbols) to the LDPC decoder. The LDPC decoder then tries to correct the missing residual symbols.

The received symbols from the inner LT code are always completely correct (if the appropriate CRC check register is enabled in the SensLAB CC2420 radio unit), the task of the decoder is to determine the value of the unknown symbols. In the case of Raptor codes, the outer LDPC code will attempt to infer the missing fraction of symbols ς, which were not connected by the inner LT code. Hereafter the missing fraction of symbols not connected by the LT code will also be referred to as erasures.

An erasure-resilient LDPC code is used to complete the decoding process and its function can be described as follows: if a parity-check equation exists that includes only one erased bit, the correct value for the erased bit can be determined by choosing the value that satisfies even parity [35]. This description highlights the fundamental operations of the parity check equations, and care should be taken to insure that these operations can complete without intractability. These checks make up the H-matrix in the LDPC code.

Finding good sparse H-matrices for our particular implementation is not practical. Most pre-designed options are not scalable in terms of our requirements with regard to message length, code-rate and regularity. Subsequently, a good H-matrix needs to have no stopping sets or girth sizes (smallest cycle) larger than 4.

Parity-check Matrix

The construction of a high performing LDPC code requires the design of a sparse parity-check matrix H and then determining a generator matrix G. The way to design such a matrix is to start by assigning weights to it that describes the number of non-zero elements represented by the row weight w_r

and column weight w_c. Suffice it to say, the distribution of these weights can classify the LDPC code into two categories: a regular LDPC code, where the row and column weights are constant throughout H, and an irregular LDPC code with random constructions of w_c and w_r.

One way to create such a matrix is by using an existing algorithm complying to rules that ensures a high performance LDPC design. This design will require a different characterisation of the H-matrix in terms of its symbol connections. Moreover, a degree distribution can be used to characterise such a parity-check matrix. For an irregular parity-check matrix the fraction of columns can be designated by the symbol i in v_i, and the fraction of rows of weight i by h_i. The set v and h is collectively referred to as the *degree distribution* of the H-matrix. For this design the degree distribution should be converted to an edge perspective; by using an edge-perspective degree distribution polynomial. The fraction of the edges connected to a degree-i variable node is expressed as λ_i; and ρ_j is the fraction that are incident to a degree-j check node. Formally, this can be expressed by:

$$\sum_i = \lambda_i = 1, \tag{3.4.1}$$

$$\sum_j = \rho_j = 1. \tag{3.4.2}$$

The functions used to describe the degree distributions are given in polynomial form in equation 3.4.3 and 3.4.4, and can also be converted back to a node-perspective by using equation 3.4.5 and 3.4.6.

$$\lambda(x) = \sum \lambda_i x^{i-1} \tag{3.4.3}$$

$$\rho(x) = \sum \rho_j x^{j-1} \tag{3.4.4}$$

$$v_i = \frac{\frac{\lambda_i}{i}}{\sum_i \frac{\lambda_i}{i}} \tag{3.4.5}$$

$$h_j = \frac{\frac{\rho_j}{j}}{\sum_j \frac{\rho_j}{j}} \tag{3.4.6}$$

An appropriate polynomial degree distribution, that conforms to the parity check rules and the additional erasures introduced by the LT code, is crucial for the design of a H-matrix in a Raptor code. In [22] Shokrohalli's proof rest on the assumption that a particular distribution can be used in order to demonstrate the workings of a Raptor code. The mentioned distribution has a message edge degree $\lambda(x) = \frac{(2x+3x^2)}{5}$, subsequently represented by a right-regular Tanner graph. He claimed that it should be sufficient to generate a LDPC parity-check matrix for a Raptor code. For this polynomial the column distribution is given by $v_2 = \frac{1}{2}$, $v_3 = \frac{1}{2}$, and the row distribution is regular

$h_i = 1$, for some $i > 0$. This particular polynomial was considered as a guideline in the design of the parity-check matrix.

To construct such parity-check matrices an algorithm, called the progressive edge growth (PEG) algorithm, proposed by David MacKay was used [54]. This algorithm was considered, since it can potentially allow for a scalable message length and it provides the calculation of the G-matrix. The inputs to this algorithm include: code block length n, code-rate r, column weight polynomial v, and row weight polynomial h. The H-matrix of dimension $n(1-r)$-by-n is calculated by first under filling the rows and columns of the matrix and then adding rows and columns until the appropriate lengths are reached. The algorithm then checks for dependencies and refills entries if they exist. The PEG algorithm is based on mathematical assumptions and random constructs, as such it needs to be considered for evaluation before final integration in the Raptor code.

Erasure-resilient Decoding

An erasure-resilient decoder is required for the LDPC code, since the typical soft-decision decoder will not suffice. In this case, the messages passed along the Tanner graph edges are straightforward (binary). The variable node sends the same outgoing message M to each of its connected check nodes.

An example in [35] was used to demonstrate the decoding algorithm design for the SensLAB LDPC decoder. The example starts by considering an arbitrary irregular H matrix shown in equation 3.4.7 (similar to the irregular matrices we need to generate). This is a binary code with $m = 3$ parity-check constraints and has a codeword length of $n = 6$.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.7}$$

A code C consists of all six strings $c = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6]$, which satisfy all three parity-check equations. The constraints are written in matrix form as indicated by equation 3.4.8. The stopping criterion is satisfied if the maximum number of iterations are reached or if the syndrome is equal to '0', as shown in Algorithm 3.

When a check node receives only one 'ϵ' (erasure) message, only then it can calculate the value of the unknown bit; by choosing the value that satisfies parity. The check nodes send back different messages to each of their connected bit nodes. The message direction is indicated by labelling the edges as $E_{j,i}$ for the message from the j-th check node to the i-th variable node. This declares the value of the i-bit '1', '0' or 'ϵ' as determined by the j-th check node. If the variable node of an erased bit receives an incoming message which is '1' or '0' the bit node changes its value to the value of the incoming message. This

process is repeated until all the bit values are known, or until some maximum number of decoder iterations was reached and the decoder fails.

Figure 3.10 represents this iterative process. Here we use the notation B_j to represent the set of bits in the j-th parity-check equation of the code.

For the code in this example[6] we have $B_1 = \{1, 2, 4\}, B_2 = \{2, 3, 5\}, B_3 = \{1, 5, 6\}, B_4 = \{3, 4, 6\}$. Similarly the notation A_i is used to present the parity-check equations that check on the i-th bit of the code. We have $A_1 = \{1, 3\}, A_2 = \{1, 2\}, A_3 = \{2, 4\}, A_4 = \{1, 4\}, A_5 = \{2, 3\}, A_6 = \{3, 4\}$. Algorithm 3 outlines the message-passing decoding (in this case it uses an erasure correcting BP algorithm) on the BEC with the received message string $c^* = [c_1^*, c_2^*, ..., c_n^*]$ as the input, with the output $M = [M_1, M_2, ..., M_n]$. The message, labelled M_i for the i-th bit node, declares that the bit values are known, or that some maximum number of decoder iterations has passed. This is demonstrated in Algorithm 3 with regard to the stopping criterion in equation 3.4.8.

$$H \times c^{*T} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.4.8}$$

An example of the each step is illustrated in figure 3.10, where a codeword $c = [0\ 0\ 1\ 0\ 1\ 1]$ is sent over an erasure channel and the message $c^* = [0\ 0\ 1\ \epsilon\ \epsilon\ \epsilon]$ is received. In terms of SensLAB implementation the 'ϵ' represents random erasures introduced by the weakened inner LT code in the Raptor code.

In the first step, the check node messages are calculated from the known values in the received message. If the check node has one incoming erasure 'ϵ', the new calculated value of the connected variable nodes will be the outgoing message, which will be passed back to the erased variable node in the next step. In step 2, each variable node that has an unknown value 'ϵ' uses its incoming messages to update its value, if possible. The procedure is summarised in the steps given below:

- Step 1: The check node messages are calculated. The 1st check node C_1 is joined to v_1, v_2 and v_4, and so has incoming messages '1', '0' and 'ϵ'. Since the check node has one incoming 'ϵ' message, from the v_4 variable node, its outgoing message on this edge, $E_{1,4}$, will be the value of the 4th codeword bit $E_{1,4} = M_1 \oplus M_2 = 0 \oplus 0 = 0$. The 2nd check C_2 includes v_2, v_3 and v_5, and so has incoming messages '0', '1' and 'ϵ'. Since the check node also has one incoming erasure from the v_5 node, its outgoing message on this edge, $E_{2,5}$, will be the value of the 5th codeword bit

[6] Note that the same values from the example in [35] were used.

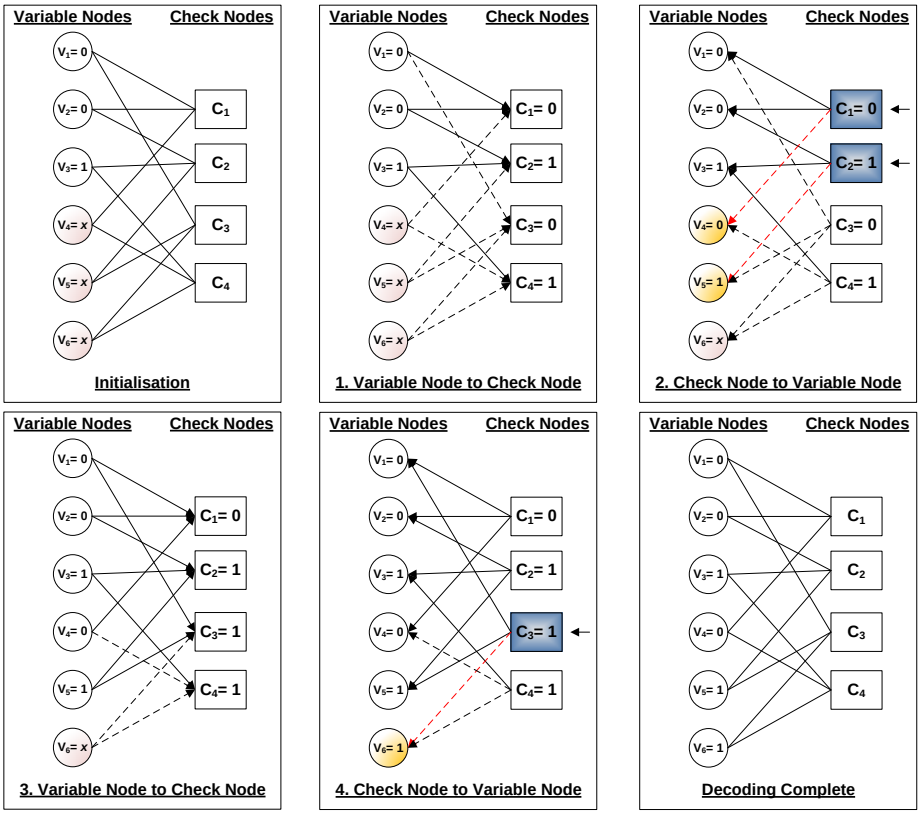

Figure 3.10: Belief propagation decoding of an irregular ($v_1 = \frac{1}{2}, v_2 = \frac{1}{3}, v_3 = \frac{1}{6}, h_3 = \frac{2}{3}, h_4 = \frac{1}{3}$) erasure-resilient LDPC code.

$E_{2,5} = M_2 \oplus M_3 = 0 \oplus 1 = 1$. The rest of the checks have more than one erasure and can, therefore, not be used to determine the values of any of the bits in this step.

- Step 2: Each bit node that has an unknown value uses its incoming messages to update its value. The 4th bit is unknown and has incoming messages, of '0' ($E_{1,4}$) and 'ϵ' ($E_{4,4}$) and so it changes its value to '0'. The 5th bit is also unknown and has incoming messages of '1' ($E_{2,5}$) and 'ϵ' ($E_{3,5}$) and so it changes its value to '1'. The 6th bit is also unknown. However, it has incoming messages of 'ϵ' ($E_{3,6}$) and 'ϵ' ($E_{4,6}$) and so cannot change its value. At the end of Step 2 we are left with $M = [0\ 0\ 1\ 0\ 1\ \epsilon]$.

- Step 3: Since there is one more erasure to correct, the Stopping criterion will not be satisfied. The algorithm will continue with Step 1 using the updated version of M and repeat until the maximum number of iterations is exceeded, or until the syndrome is equal to 0.

As previously mentioned the PEG algorithm, found in some of the work produced by MacKay, was used to create an irregular H-matrix. A few matri-

Algorithm 4 LDPC Erasure Decoding

repeat

  Variable node $\rightarrow$ Check node
  for $j = 1$ to m **do**
    for all $i \in B_j$ **do**
      if all messages into check j other than M_i are known **then**
        $E_{j,i} = \sum_{i' \in B_{j,i' \neq i}} (M_{i'} \bmod 2)$
      else
        $E_{j,i} = \epsilon$
      end if
    end for
  end for

  Check node $\rightarrow$ Variable node
  for $i = 1$ to n **do**
    if $M_i =$ 'unknown' **then**
      if there exists a $j \in A_i$ s.t. $E_{j,i} \neq x$ **then**
        $M_i = E_{j,i}$
      end if
    end if
  end for

  Stopping Criterion
  if all M_i known or maximum iteration reached **then**
    Syndrome $= M \cdot H^T$
    if Syndrome $== 0$ **then**
      Decoding Success
      break
    else
      Decoding Failure
      break
    end if
  end if

  Iteration++

until all erasures recovered or maximum iterations reached

ces were produced with this algorithm, using the proposed polynomial provided by Shokrohalli, and evaluated to determine if it is capable of correcting up to 10% of random erasures. Each matrix's erasure correcting performance was evaluated over 1000 independent tests with independent random erasures. The best-performing H-matrices are presented below in figure 3.11 to figure 3.13. For clarity, the figures on the right, labelled (b), are magnified illustrations of the roll-off area.

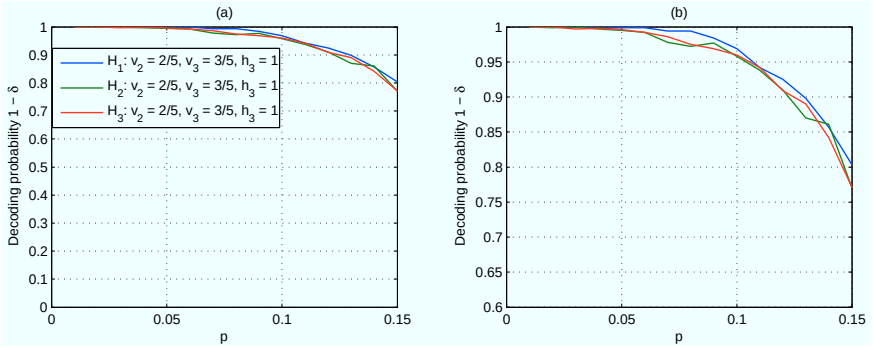

Figure 3.11: Performance evaluation of irregular parity-check matrices ($m = 24, n = 100$).

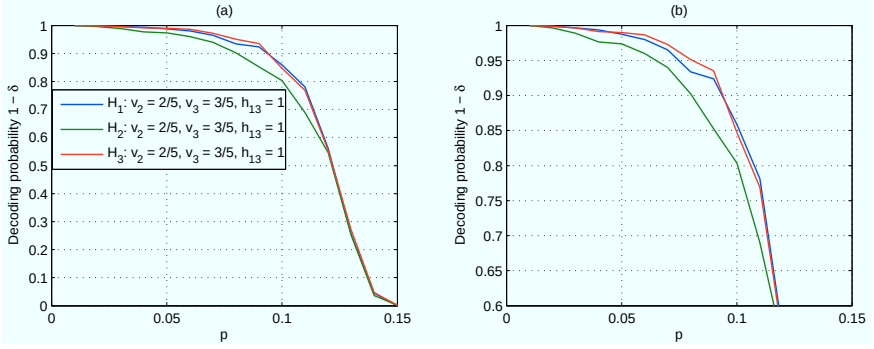

Figure 3.12: Performance evaluation of irregular parity-check matrices ($m = 80, n = 500$).

We can see that the selected parity-check matrices H_1 - H_3 performs almost identically. Matrix H_1 was selected in the LDPC pre-code for $n = 100$, since it performed smoothly throughout p, with a slightly higher roll-off peak. Matrix H_3 was chosen for message length $n = 500$, and H_2 for $n = 1000$ respectively.

The literature suggests an erasure correcting capability of up to $\varsigma = 0.05$ is excellent, and the selected matrices achieve this at reasonably high probability. The maximum number of decoding iterations were limited to 25. Assuming that the parity-check graphs are cycle free, an error correcting threshold can be estimated for a particular degree distribution, for an estimated number of

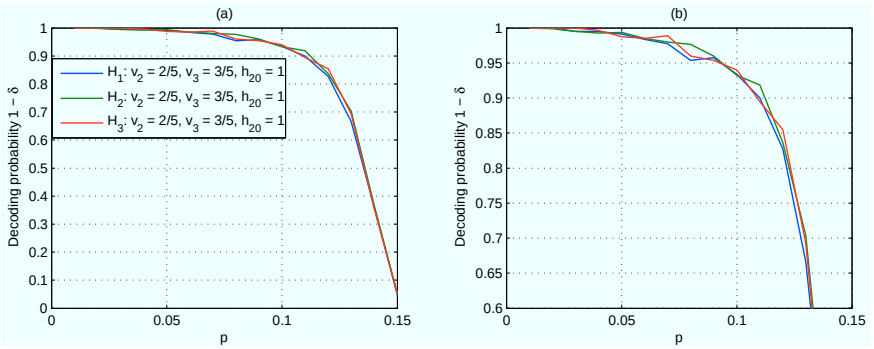

Figure 3.13: Performance evaluation of irregular parity-check matrices ($m = 180, n = 1000$).

iterations. The theory in [35] suggests that 25 iterations is enough for designs with these codeword lengths. The Raptor PDD now needs to be designed requiring that $(1-\varsigma)n$ is decodable (95% of n) with high probability. Basically, we require ς to be smaller than the LDPC error-rate, since smaller message sizes may produce larger deviations from the average ς. The next section will explain the design of an optimal degree distribution capable of meeting these requirements.

3.4.2 Raptor Degree Distribution

In this section, we will design the Raptor code degree distribution (the LT code's PDD used in the inner part of the Raptor code). The typical Robust soliton PDD used in a LT code is not sufficient for this implementation, since we require only a certain fraction of the symbols after decoding and not all. Moreover, the right interplay between the pre-code C and the LT code used to create the encoded symbols is crucial. In order to obtain this the PDD affords re-design and optimisation. Evidently, both the received encoded symbols and the constraint symbols are used for decoding the intermediate block c_n.

For Raptor codes the degree distribution design is much easier to optimise and implement, since it is capped at a value usually much smaller than the original message length k. We denote the distribution by its generator polynomial $\Omega(x) = \sum_{i=0}^{k} \Omega_i x^i$. In notation the expectation, or average degree, of this is given by $a = \Omega'(1)$.

The same conditions for the LT code degree distribution apply for the Raptor degree distribution, since it is utilising the BP decoder. However, the distribution needs to be modified to ensure enough symbols of degree-one, to permit the initiation of the decoding process. Additionally, the distribution should be capped at some maximum degree d_{max}, giving it an appropriate average weight for each degree and allowing constant complexity.

This can be achieved by selecting a proper overhead parameter ϵ and calculating the maximum degree at which the distribution will be capped, $d_{max} :=$

$\left\lceil \frac{4(1+\epsilon)}{\epsilon} \right\rceil$. The polynomial is defined by equation 3.4.9, where $\mu := \frac{\epsilon}{2} + \frac{\epsilon^2}{4}$. Recall that the analysis of the BP decoding process in section 2.3.2 was used to define this equation.

$$\Omega(x) = \frac{1}{(\mu+1)}(\mu x + \frac{x^2}{1\cdot 2} + \frac{x^3}{2\cdot 3} + ... + \frac{x^{d_{max}}}{d_{max}\cdot(d_{max}-1)} + \frac{x^{d_{max}+1}}{d_{max}}) \quad (3.4.9)$$

From Lemma 1 in section 2.3.3 Shokrohalli argued that any aggregate of $m = (1+\frac{\epsilon}{2})n + 1$ received symbols from the LT code using $\Omega(x)$ as its degree distribution, is sufficient to recover at least $(1-\varsigma)n$ input symbols via BP decoding, where $\varsigma = \frac{\epsilon}{4(1+\epsilon)}$.

One way to optimise the PDD for our requirements is to consider using a linear programming (LP) optimisation technique shown in 3.4.10. This technique was suggested by Shokrohalli in [19, 22], and extended in [55] by Sejdinovi. It was shown that it can be used to find optimal weights w_d for each degree in the distribution function. As such, the work done in [55] was followed as an guideline to proceed with the design.

$$\begin{aligned} \min \quad & f^T x \\ \text{s.t.} \quad & A \cdot x \leq b \\ & x \geq 0. \end{aligned} \quad (3.4.10)$$

In order to use LP we need to determine the objective function and the constraints. Equation 3.4.9 can be set as the cost vector f in equation 3.4.10, which we want to minimise. The constraints matrix A is obtained by first calculating the matrix dimensions, which is set to a p-by-d_{max} matrix, since the interval $[\varsigma, 1]$ can be discretised with p equidistant points. The inequality column vector b was simplified in [55] to be $-ln(x_i)$, where it should also contain $i = p$ rows to satisfy the system of equations. The lower bounds are set to $w_d \geq 0$. These variables and their respective conditions reduces the LP optimisation problem to the following:

$$\text{LP: min } \alpha \textstyle\sum_d^{d_{max}} \frac{w_d}{d} \text{ (objective function)}$$

$$\begin{aligned} \alpha \sum_{d=1}^{d_{max}} w_d(1-x_i)^{d-1} \geq -ln(x_i), i \in N_p, \\ w_d \geq 0, d \in N_{d_{max}} \text{ and} \\ \sum_{d=1}^{d_{max}} w_d = 1, \end{aligned} \quad (3.4.11)$$

where $d \in N_{d_{max}}$. When a solution is determined, each individual weight of the degree distribution can then be calculated from $w(x)$ as $\Omega(x) = \frac{\int_0^x w(z)dz}{\int_0^1 w(z)dz}$.

The duality property of LP allows us to solve the inequality by switching the signs of the constraint matrix A, and of the inequality column vector, since typical LP[7] solve a problem described in equation 3.4.10 with an opposite inequality sign. The solution obtained by the LP technique is converted back to the node-perspective by multiplying each entry by the cost vector and followed by re-normalising. Very small values of w_d can be discarded (set to zero) and re-normalised to obtain the final PDD function.

The LP optimisation technique was used to develop three Raptor PDDs. These PDDs are shown in figure 3.14. Each distribution is capped at some discrete entry less than the value of the codeword length n, as we expected. The evaluation of each PDD is crucial and must comply to the conditions stated previously. The values of each of the weights for the different message lengths are indicated in table 3.2, along with the expected average degree a.

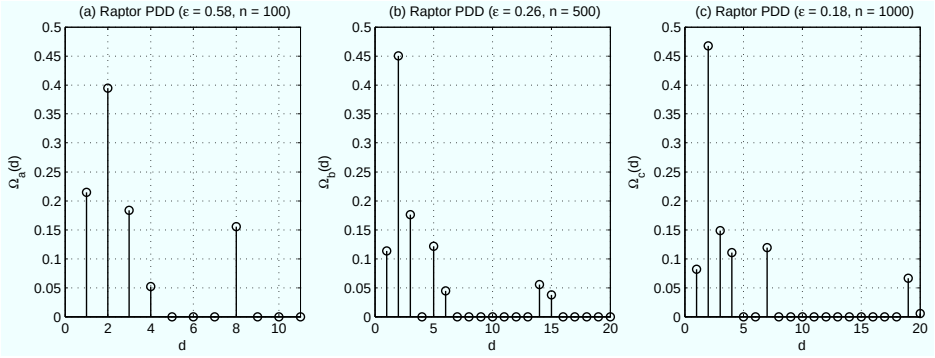

Figure 3.14: Designed Raptor degree distributions.

3.4.3 Density Evolution

In the previous section we designed an optimised PDD that constitutes an integral part of the Raptor code design. The optimisation of this PDD involved some heuristics and a few mathematical assumptions. In section 2.3.3 the correct interplay between the pre-code and the LT code was emphasised as it is crucial for the Raptor code. For these reasons it is important to test and evaluate the optimised PDD independently before considering it as part of the final Raptor code.

The new PDD design should weaken the inner LT code, inherently allowing the Raptor code to achieve an optimal encoding and decoding complexity within a constant degree. As such, it should not fail decoding the symbols prematurely until a certain fraction is decoded successfully. A method collectively referred to as *Density evolution* (DE) can be used to realise such a test [8, 56].

[7] In Matlab the function linprog($f, A, b, [], [], lb$) solves the linear programming problem. Example code was provided by Sejdinovi and is used with permission of the author.

Table 3.2: Raptor degree distributions for various design values of n and ϵ.

n	100	500	1000
Ω_1	0.2144	0.1135	0.0820
Ω_2	0.3947	0.4505	0.4674
Ω_3	0.1834	0.1762	0.1487
Ω_4	0.0519	0	0.1106
Ω_5	0	0.1217	0
Ω_6	0	0.0446	0
Ω_7	0	0	0.1192
Ω_8	0.1554	0	0
Ω_{14}	0	0.0554	0
Ω_{15}	0	0.0378	0
Ω_{19}	0	0	0.0663
Ω_{20}	0	0	0.0056
ϵ	0.58	0.26	0.18
a	3.0055	3.7634	4.1136

The evaluation of typical PDDs used for a larger Raptor code is provided in appendix A.1. This was used as a guideline to evaluate the PDDs that was designed in the previous section.

The DE method is based on an evaluation of the asymptotic finite length behaviour of the expected BP decoding process. Parameters of the pre-code and the PDD can be altered until an optimal result in terms of decoding success and overhead is obtained, by examining the asymptotic behaviour. As such DE can be used to test or anticipate the performance of a predetermined PDD.

More formally, the BP decoder should ensure that the decoding of $(1-\varsigma)n$ symbols is possible with an overhead of $n+\epsilon \cdot n$, at consecutive instances, and at high probability. From [8, 56] the expected asymptotic fraction of intermediate symbols connected to encoded symbols of reduced degree-one can be written as:

$$1 - x - e^{(1+\epsilon)\Omega'(x)}, \tag{3.4.12}$$

where x is the fraction of recovered intermediate symbols. The analysis reveals that if x_0 is the smallest root of this equation in the interval $[0, 1)$, then asymptotically the expected fraction of intermediate symbols still un-decoded at the end of the LT decoding process is $1 - x_0$. The root of this equation, therefore, indicates where the decoding process will fail during the decoding process, and what fraction of symbols is expected to be left un-decoded.

For a Raptor code it follows that, asymptotically, the degree distribution $\Omega(x)$ has to be designed in such a way as to ensure that all but the frac-

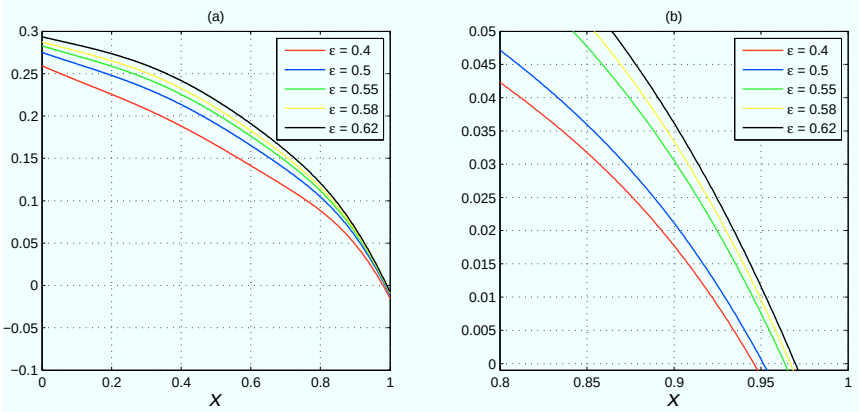

Figure 3.15: Density evolution of the asymptotic finite length behaviour for $\Omega_a(x)$.

tion ς of encoded symbols are decoded by the LT decoder. By substituting the derivative of the designed degree distribution $\Omega(x)$ in equation 3.4.12, the asymptotic analysis can reveal how the degree distribution is expected to perform. For equation 3.4.12, ϵ can be altered to evaluate the tolerable overhead performance of the PDD. Asymptotically, the PDD has to be designed as to ensure that:

$$\sup\{x \in [0,1) : 1 - x - e^{(1+\epsilon)\Omega'(x)} > 0\} \tag{3.4.13}$$

is maximised.

The evaluation process can be realised by back substituting the designed PDD $\Omega(x)$ into the function $1 - x - e^{(1+\epsilon)\Omega'(x)}$. Different values of ϵ can be tested, which must satisfy the asymptotic condition stated in equation 3.4.13. In addition, c can be altered in the function $1 - x - e^{(1+\epsilon)\Omega'(x)} - c\sqrt{\frac{1-x}{k}}$, to achieve a desirable interplay between the decoding error probability and ς.

The DE evaluation technique was performed on the Raptor code with message length $n = 100$ in an effort to validate the asymptotic trend in comparison to the larger Raptor code's asymptotic trend in appendix A.1. In figure 3.15 the asymptotic behaviour of the designed PDD, presented in figure 3.14(a) (for $n = 100$), is indicated with different values of ϵ.

Now we need to reconsider the erasure correcting capability of the LDPC pre-code that was designed in section 2.2.3, and re-evaluate its performance according to the DE trends given by the different ϵ values. The selected parity-check matrix (H_1), from figure 3.11, can correct erasures of 5% at $\approx$ 99% probability. A closer look at figure 3.15(b) reveals that many of the selected ϵ values will suffice. However, since these are estimates a conservative choice for ϵ is selected instead. In this case, the PDD $\Omega_a(x)$ was designed by selecting $\epsilon = 0.58$.

The methods used to develop the Raptor code PDD is based on mathematical assumptions and heuristics. Therefore, it seems reasonable to test

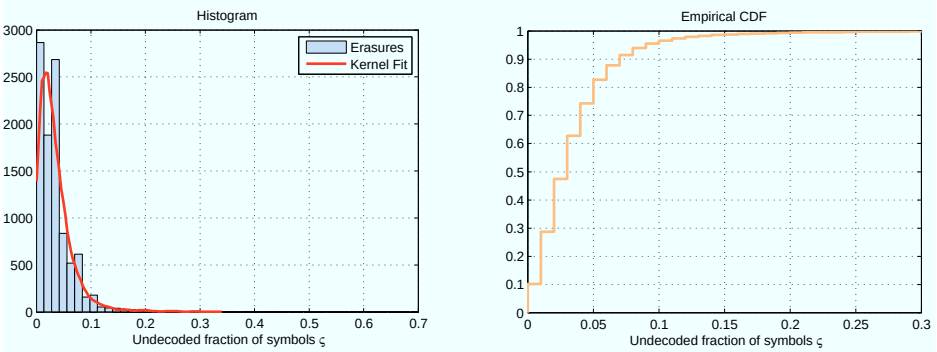

Figure 3.16: Belief propagation decoding test for $\Omega_a(x)$ with $\epsilon = 0.58$ (mean $= 0.0347$, std $= 0.0359$).

the PDD independently and evaluate its performance based on the intended design criterion.

The PDD is tested with the actual BP decoder, and a histogram is generated that indicates the frequency of the un-decoded fraction of symbols. The expected value, skewness of the distribution, outliers, median, quartiles, and deviations can be examined to determine how the PDD operates within the design specifications. The histogram, fitted with a kernel smoothing function, and the empirical cumulative distribution is shown in figure 3.16. These figures indicate that the majority ($\approx 80\%$)of the un-decoded symbols are between $0 - 5\%$. Further inspection in comparison to the other two Raptor codes will reveal the importance of the pre-code for this specific small message length Raptor code.

Up to this point the tests and evaluations of the parity check matrix H_1 and the optimised PDD $\Omega_a(x)$ reveal that: $\approx 95\%$ of the n symbols are decodable by the BP decoder, and the other 5% may be decoded by the LDPC decoder. For the other message lengths ($n = 500$ and $n = 1000$) the same procedure was followed and the results for each are shown in the graphs below. Figure 3.17 and figure 3.18 indicate test results for a $n = 500$, and figures 3.19 and figure 3.20 for $n = 1000$.

The mean values and standard deviations of each of the PDD evaluations provided in the presented figures are not robust statistics as it is not resistant to extreme observations. In the event of such observations the LDPC code-rate should be adapted to ensure tighter bounds around the expected un-decoded fraction of symbols produced by the BP decoder. Therefore, a boxplot is provided in figure 3.21 to illustrate the distribution skewness and to reveal extreme outliers in each case. In this figure it is clear that: for the shorter message length ($n = 100$) the erasure percentage is distributed more (larger range between the quartiles) and not skewed (indicated by the median value). There also seems to be outliers distributed further away from the median, compared to the other two PDDs. The larger message sizes (n = 500 and 1000) has less distributed (smaller range between the quartiles) erasure percentages,

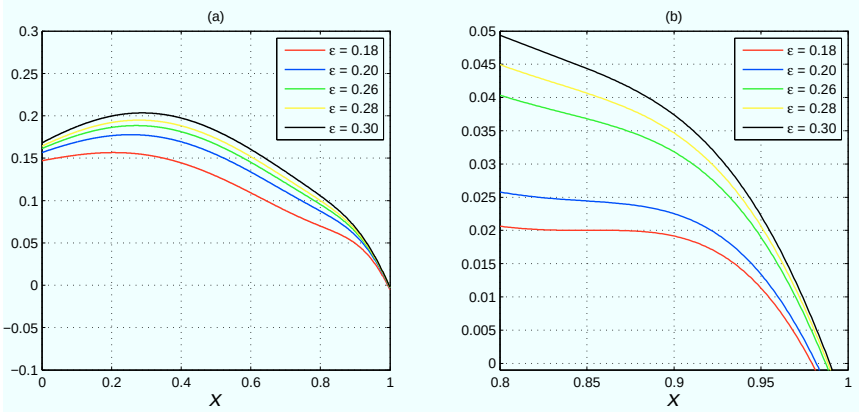

Figure 3.17: Density evolution of the asymptotic finite length behaviour for $\Omega_b(x)$.

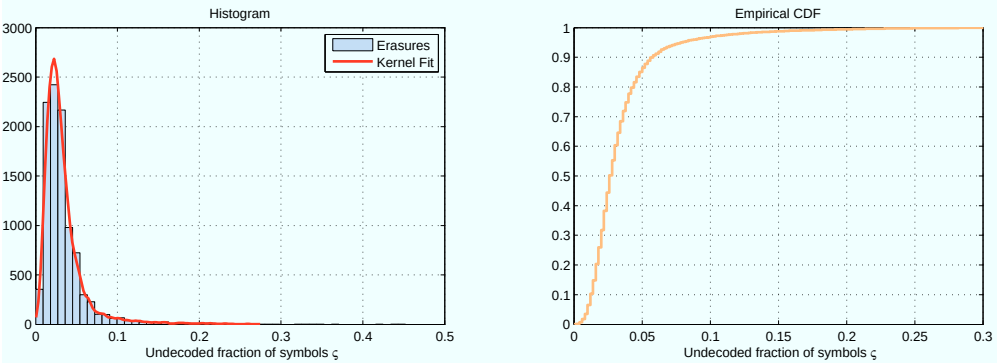

Figure 3.18: Belief propagation decoding test for $\Omega_b(x)$ with $\epsilon = 0.26$ (mean $= 0.0339$, std $= 0.0292$).

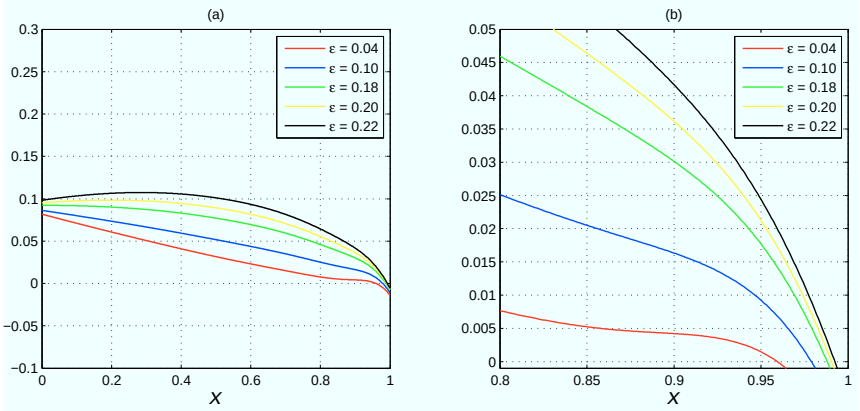

Figure 3.19: Density evolution of the asymptotic finite length behaviour for $\Omega_c(x)$.

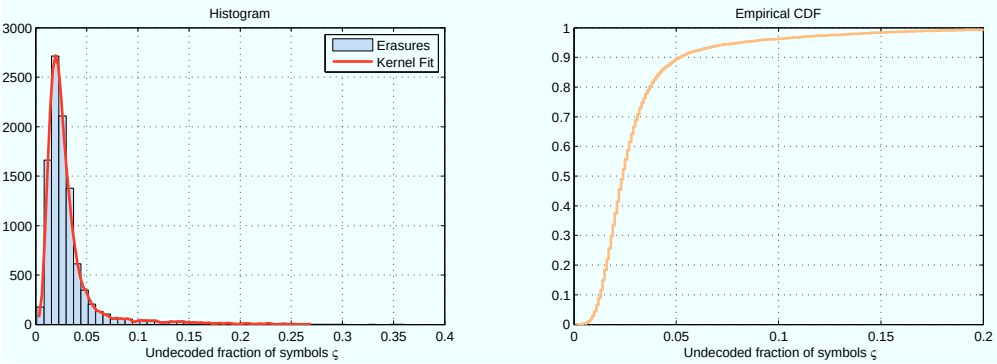

Figure 3.20: Belief propagation decoding test for $\Omega_c(x)$ with $\epsilon = 0.18$ (mean = 0.0317, std = 0.0301).

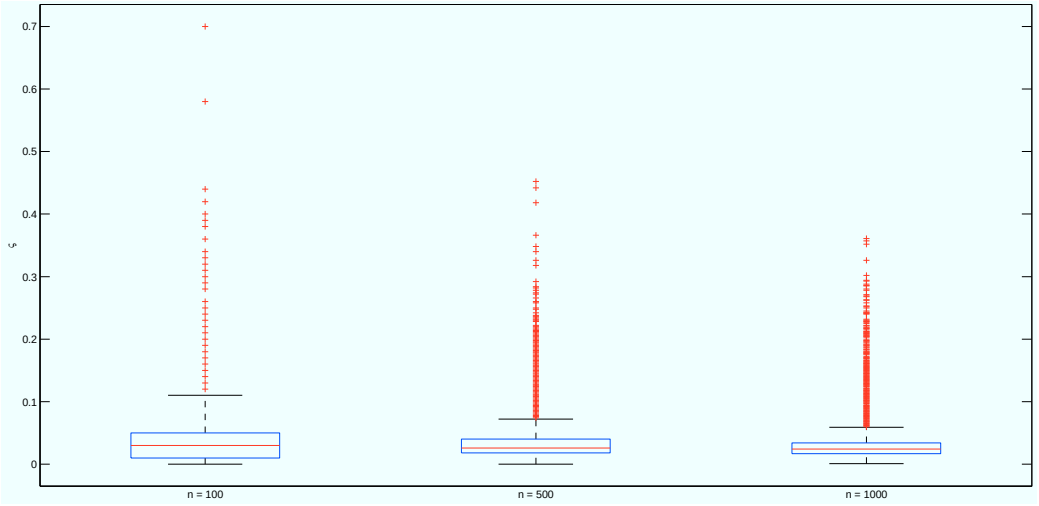

Figure 3.21: Boxplot comparison of the PDD performances.

is skewed to the left and has outliers closer to the median. These statistics were used to determine if the LDPC code (code-rate) needed to be adapted to accommodate the PDD's performance in order to correct a larger fraction of erasures. In this case, the LDPC code rate was adapted for $\Omega_a(x)$ $(n = 100)$ to ensure the correction of up to 10% erasures with slightly higher probability than the other two PDDs. The adaption entails a lower code rate compared to the other two codes. This allows the $n = 100$ LDPC code to correct slightly more erasures than the other two (recall from figure 3.11).

This section described the segregated designs of each component in the Raptor code. Each component was tested and evaluated independently and highlights the importance of the design parameters for each of the three Raptor codes. The erasure correcting capability of the LDPC code, as well as the decoding capability of the LT code, requires careful consideration in each design step. The presented evaluations can confer that the performance of the PDD is consistent with our anticipated LDPC code design criterion and with

Table 3.3: Performance summary of the designed PDDs.

n	100	500	1000
Outliers	267	586	780
max	0.70	0.45	0.36
Upper-adjacent	0.11	0.072	0.059
3de-Quartile	0.05	0.04	0.034
median	0.03	0.026	0.024
1st-Quartile	0.01	0.018	0.017
Lower-adjacent	0	0	0.001
min	0	0	0

high certainty. A complete performance comparison and in-depth discussion regarding these findings will be presented in chapter 4.

3.5 Design Cycle and Design Tools

In this section, we will describe the design cycle by referring to the research methodology in section 1.4. A description of all the tests and the evaluations of the candidate FEC codes for SensLAB will be explained. A summary of the design cycle in accordance with the main objectives described in section 1.4 is described below.

1. Investigate:
    a) the SensLAB platform, node hardware and software constraints;
    b) information theory fundamentals and channel model derivations;
    c) theory on channel capacity and its derivations;
    d) FEC theory and analyses;
    e) Fountain code theory and its analyses;
    f) the encoding and decoding schemes and its analyses.
2. Plan:
    a) identify suitable SensLAB hardware and software combinations for hosting the FEC strategy;
    b) identify an appropriate channel model to use as a reference for the FEC design;
    c) identify an appropriate error-correction strategy for SensLAB;
    d) identify the FEC strategy's corresponding encoding and decoding schemes;

e) identify possible design techniques to allow for a scalable FEC design in terms of transport frame size;

f) identify an appropriate evaluation framework for the candidate codes;

g) identify appropriate tools for the design and evaluation of the candidate FEC codes.

3. Create:

a) create a design framework and a simulator to evaluate the candidate codes;

b) design three candidate LT codes using two decoding methods and appropriate parameter selections;

c) design three candidate Raptor codes using the identified techniques.

4. Evaluate:

a) evaluate the two different LT code decoding schemes over the three different frame sizes, and in comparison to their expected performance;

b) evaluate the separate components in the Raptor code design;

c) evaluate the final LT codes and final Raptor codes design over the estimated SensLAB channel conditions by using the simulator.

In numerous occasions the design and evaluation cycle overlapped in order to ascertain whether or not appropriate results has been reached before proceeding with the rest of the design. The theory and concepts provided in the literature study were evaluated by using Matlab, including all the LT code and Raptor code designs and simulations. Most functions used for the design of the FEC codes comprised of powerful matrix and array manipulations. Furthermore, the random number generator allowed for easy sampling from uniquely developed distributions and was used to characterise the BEC channel. Optimisation tools such as LP, and statistical descriptive tools such as correlation functions and curve fitting estimators was also advantageous during the investigation and evaluation process.

Each component in the simulator corresponds to a separate Matlab function. All the functions and scripts used to design, analyse and test the candidate Fountain codes are listed below.

- $Ideal\_Soliton.m$: The theoretically ideal degree distribution used in the LT code process.
- $Random\_linear\_failure.m$: Used to evaluate theoretical bounds on a Gaussian decoder.

- $RS\_Degree.m$: This script is the Robust soliton degree distribution introduced by M. Luby.
- $Random\_matrix.m$: Used to evaluate the overhead-failure performance of a Gaussian decoder.
- $asymptotic\_finite\_length\_behaviour.m$: In this script, DE is performed to evaluate the asymptotic behaviour of the optimised PDD. The weight of each degree is set by the user as input.
- $[Omega, oh\_lb] = lin\_opt\_degdist(d\_max, delta, u)$: This function uses LP optimisation and computes the weights corresponding to each degree, and commutes them to a degree distribution for the Raptor code.
- $[H\_Matrix, final\_column\_weights] = H\_Generator(n, r, v, h)$: This function uses the PEG algorithm to determine the parity-check matrix.
- $[G\_Matrix] = G\_Generator(H\_Matrix)$: This function generates the generator matrix, from the parity-check matrix, used in the encoding process of the LDPC pre-code.
- $[Codeword] = LDPC\_Encoding(G\_Matrix, Message)$: Here the LDPC pre-code codeword is calculated.
- $[E] = LT\_Encoding(Index, n, Omega, Codeword)$: On the fly LT encoding is performed with this function.
- $[LT\_Output] = LT\_Decoding(n, LT\_Matrix)$: LT decoding is performed by this function using the BP algorithm.
- $[LDPC\_Output, Iterations] = LDPC\_Decode(LT\_Output, m, n, H\_Matrix, MAX\_ITER)$: Here the final LDPC erasure correction decoding is performed. The function also returns the number of iterations it used.

Scalability is realised by using the PEG algorithm in combination with the LP optimisation. All the corresponding simulator outputs (parameters, arrays, matrices, and distributions) from each of the candidate codes will be discussed in the following section.

3.6 Simulator

The simulator was written in Matlab to simulate loss in a BEC environment and to validate the operability of the design parameters, matrices and distributions. The BEC channel erases randomly received vector entries in the received LT code matrix, since this is the part of the Raptor code (the inner part) exposed to the channel. The simulator will be explained by referring to

the flow diagram in figure 3.22. This section will also provide an overview of the code design in terms of scalability and how it can be realised by using the presented design framework. The message length k, codeword length n, and overhead factor ϵ are given as input by the user. Options for this implementation include:

1. designing and evaluating a LT code using any PDD;
2. designing a Raptor PDD for the inner LT code;
3. designing a H-matrix for the outer LDPC code;
4. evaluating the optimised PDD using DE and actual BP decoding;
5. evaluating the LDPC code's H-matrix over a BEC channel;
6. evaluating the complete Raptor code over a BEC channel.

The parameter selections input by the user is evaluated and the following parameters are calculated before producing results of the designs:

- the maximum capped degree for the Raptor degree distribution d_{max};
- the variable μ used in the degree distribution design;
- the expected un-decoded fraction of message symbols ς;
- the recommended LDPC code-rate r;
- the number of encoded symbols m necessary to decode the $(1 - \varsigma)n$ fraction of message symbols.

The LDPC's code-rate can be selected and the simulator will then create a parity-check matrix H to be used in the LDPC pre-code. A H-matrix will be generate according to this specification. This H-matrix can be modified (in terms of r) to produce more robust LDPC codes — especially for smaller code lengths. A predetermined matrix can also be given as input. The maximum number of iterations can also be adjusted, although 30 is recommended for the specified message range. The generator matrix G is then determined, which concludes the pre-code design phase. The designed LDPC code is scalable within the message range $100 - 1024$.

The DE evaluation technique is added in the simulator to assist the designer in selecting a proper ϵ, for a given message size. The LP uses the selected parameters as inputs to optimise the PDD. Each PDD weight is normalised to obtain an appropriate degree distribution for the specific design. Using the asymptotic finite length analysis, a range of ϵ values can be to investigate to determine the decoding failure as a function of the fraction of un-decoded

symbols. The LP optimisation allows for scalability, since it is independent of the message length.

A random binary message of length k is generated and encoded by the LDPC pre-code, creating the codeword c of length n as explained in section 2.2.3 and section 2.3.3. This codeword is then passed on to the inner LT code and encoded as described in section 2.3.2. Random erasures are introduced to the transmitted packets to incorporate noise in a communication system. Note that no modulation or demodulation is performed, since we are not interested in quantised channel information. All algorithms used in this simulator are based on hard-decision computations.

The receiver gathers encoded packets until an estimated message E of length m has been received and the LT decoder starts decoding. BP is used in the Raptor code, which then decodes a fraction of the codeword, represented by c^*. When there are no more degree-one symbols available, the decoder stops and passes the codeword on to the LDPC decoder. This final stage in the decoding process will attempt to decode c^*. After the maximum number of iterations is reached, or the stopping criterion satisfied, the decoded c^* is compared to our initial message of length k, to determine whether decoding was successful.

If erasures still exist in the codeword, the simulator can be set up so that the receiver receives more encoded packets until the decoding is complete. Note that for such implementations, a feedback channel in the communications link is necessary in order to send a stop message. The simulator allows for scalable designs to assist the SensLAB community with multiple practically implementable solutions, which can accommodate other network coding, or protocol related strategies. The simulator outputs, required for a Raptor code implementation in SensLAB, are summarised below:

1. the message length — n,
2. the code-rate — r,
3. the parity check matrix — H,
4. the generator matrix — G,
5. the LP optimised PDD — $Omega$,
6. the estimated number of symbols necessary for decoding — m.

Each part of the design can be set up in the simulator to analyse different performance aspects. As shown in section 3.4.1 and section 3.4.3 it is good practice to perform a unit test on the LDPC and LT codes individually before final Raptor code integration. After desirable results have been obtained in each unit test, the codes can be integrated and evaluated. There are many different ways of testing Raptor codes. The most relevant however is to evaluate

the overhead-failure performance, and the universality property over a range of erasure probabilities. In other words, how many output symbols are necessary to decode the original message, and how does the introduction of noise influence those overhead requirements. These evaluations will be considered in the next chapter to investigate the performance of the presented designed codes.

3.7 Conclusions

In this chapter, the design steps for erasure-resilient LT codes and Raptor codes were provided. Initially, two decoding strategies were considered for the LT code: BP decoding and GE decoding. A design framework was developed to test the overhead and decoding failure rates of each decoding strategy. Each decoding strategy was evaluated by comparison to its theoretical bounds, as this provides insight into its expected performance and validates the algorithms. Various statistical tools were used to analyse the algorithms to confirm its functionality.

Pseudo code of the encoding and decoding strategies were presented. The design framework was setup as an evaluation platform to determine the overhead of each decoding strategy. Three different message lengths were identified as possible candidates for SensLAB implementation.

The design of a Raptor code was also presented. The Raptor code is a concatenated code that consists of a LDPC pre-code and a LT code. Typical decoding strategies for the LDPC code could not be used, since appropriate channel information is not available. An erasure-correcting message-passing algorithm was considered instead. In addition, typical parity-check matrices and typical degree distributions could not be used for the Raptor code design. Moreover, a parity-check matrix and an accompanying optimised degree distribution was required, which afforded a thorough treatment. As such, a preliminary evaluation of the result from each design technique needed to be carried out.

A high-rate irregular parity-check matrix was designed by using a polynomial recommended by Shokrollahi and the PEG algorithm introduced by MacKay. This matrix was evaluated over an erasure channel to determine if its error-correcting capability conforms to the design criteria. It was shown that the algorithm is capable of constructing parity-check matrices for small message lengths and performs well for the purpose of our implementation in SensLAB. In addition, the designed LDPC code can also serve as a FEC strategy on its own. The PDD was designed accordingly by using a LP optimisation technique and evaluated by using the DE method. Furthermore, the PDD was evaluated with the BP decoding algorithm to determine if it conforms to its expected theoretical performance.

The next chapter will focus on the erasure-correcting performance of the LT code and Raptor code over a spectrum of random erasures and discuss further implementation considerations.

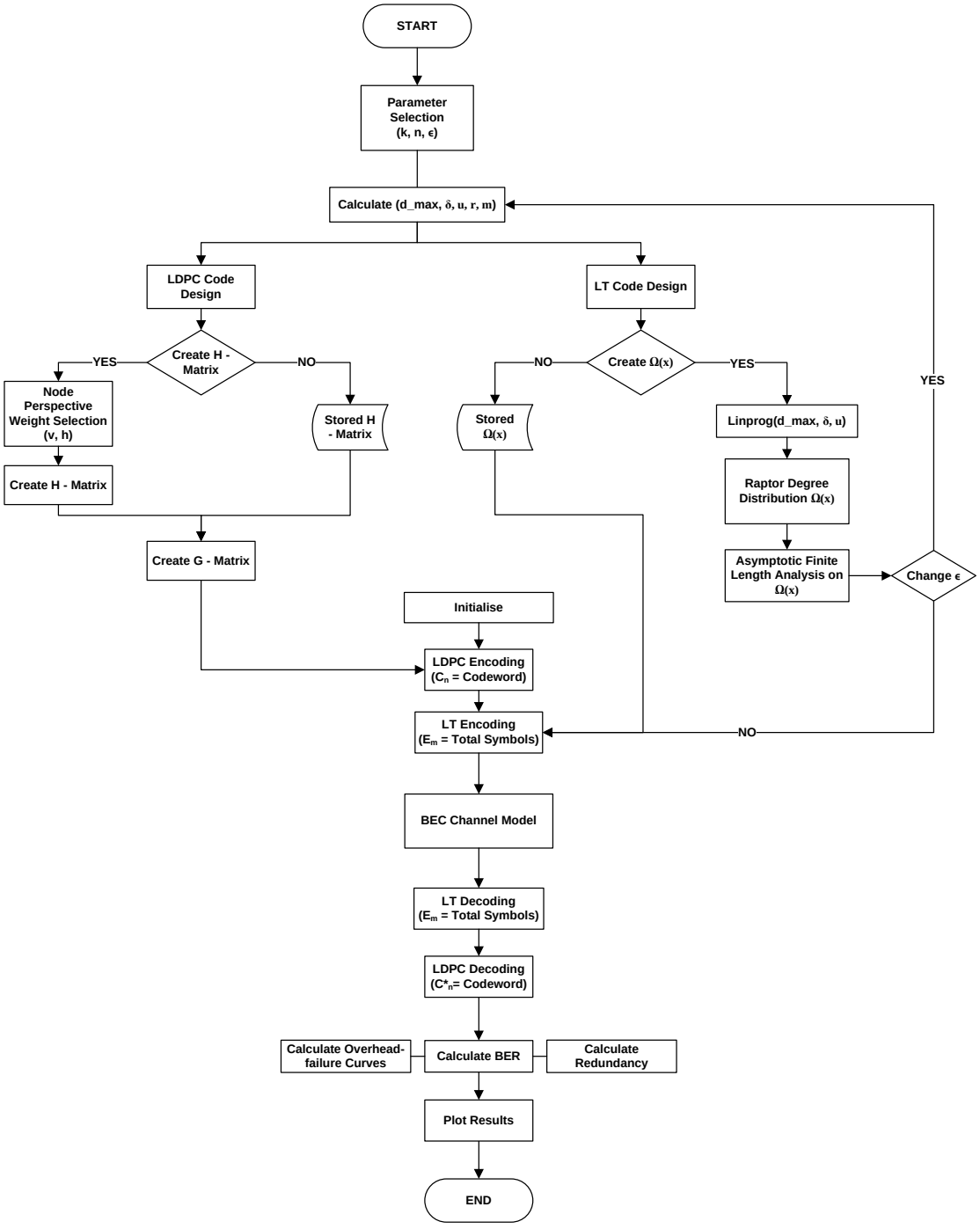

Figure 3.22: Flow diagram of simulator

Chapter 4

Results

4.1 Introduction

In this chapter, a summary of the main objectives, the collected results from simulations and a statistical analysis of the results is presented to evaluate and discuss the performance of the designed codes. The objectives, taken from chapter 1, will briefly be restated explaining the rationale behind each experiment. Most of the results will be presented by figures, followed by a table of descriptive statistics to avoid possible misinterpretation. Discussions and conclusions will be presented separately from the results.

In accordance to the motivation and the research methodology, presented in the beginning of this thesis, the aim of this study was to investigate a possible Fountain code forward error correction (FEC) strategy for the SensLAB platform. The SensLAB platform, the fundamental theoretical constructs of channel models, channel capacity theory, and Fountain code theory was investigated and analysed in an effort to identify a feasible candidate coding scheme. A channel model known as the binary erasure channel (BEC) was selected as a suitable channel model for the SensLAB platform, and served as a Fountain code design reference with regard to the associated decoding schemes. Ultimately, two candidate Fountain codes were identified as a possible FEC strategy: a LT code and a Raptor code.

A study of the SensLAB platform revealed that the sensor node resources are rather limited; and the communication layers can only accommodate short message frames. Furthermore, the communicated data between sensor nodes will most often consist of smaller message packages, constituted by measurements from the sensors. In contrast, the presented theory on Fountain codes is clear on the efficiency of the codes at very large message sizes. Therefore, special optimisation techniques and supplementary algorithms needed to be utilised to overcome possible inefficiencies, or ultimately FEC failure at shorter message sizes — particularly for the Raptor code design.

The results from the techniques and algorithms, used during the design of

the candidate codes, was presented and evaluated in chapter 3. The preliminary results presented in that chapter served as a design guide (with regard to sensible parameter selections for SensLAB), which indicated that the designs were in-line with the presented theory. The literature was meticulously adhered to considering each step of each of the FEC code developments.

Two decoding schemes were identified for the LT code: Belief propagation (BP) decoding and Gaussian elimination (GE) decoding. These two decoding schemes were evaluated in terms of shorter message sizes; in comparison to their respective expected theoretical behaviour. In both cases the decoding algorithms performed as expected.

A Raptor code was also identified as a possible candidate FEC strategy for SensLAB and designed according to the literature. However, two additional techniques were introduced to allow for a scalable high-rate irregular H-matrix design, and a supplementary optimised probabilistic degree distribution (PDD). Moreover, isolated evaluations of both these techniques were conducted and compared to their respective theoretical expectations. Linear programming (LP) optimisation and Density evolution (DE) was used to design and evaluate the expected performance of the PDD. The progressive edge growth (PEG) algorithm was used to generate the H-matrices. Conceptually, it was shown that these design techniques complemented each other, since it produced a functional interplay between the generation of un-decoded symbols and the decoding of such.

This chapter will evaluate the candidate code's performance over an estimated communication channel as a final verification of its operation. Each code will be evaluated by transmission over a modelled BEC channel. Furthermore, these results will indicate whether the designed codes comply with the rateless- and universality property, indicative of a true Fountain code. In addition, the evaluation of a reliable overhead estimate is required for implemetation consideration in SensLAB.

For each candidate code the results are provided for three different frame sizes to showcase the design framework's scalability across different frame sizes, and to evaluate the varying efficiency in terms of frame size and overhead.

4.2 LT Codes

In this section, results from the three designed LT codes are presented. Three candidate LT codes with different message sizes ($k = 100, 500$ and 1000) was designed for SensLAB. The overhead performance and the erasure correcting performance of these three candidate codes will be presented. Furthermore, it will be shown that the larger LT code ($k = 1000$) outperforms its smaller counterparts. Moreover, all three designed candidate LT codes conform to the rateless- and universality property of a true Fountain code.

According to the literature, presented in section 2.1.2 - 2.1.4, the LT code can be considered as maximum distance separable (MDS) over the BEC. Therefore, the overhead σ is a very important measure of its performance with regard to its potential for the resource limited SensLAB hardware. The overhead is defined as the number of encoded symbols that need to be collected by the receiver in order to successfully recover the source message symbols. In figure 4.1 the overhead performance of the three LT codes are shown. In (a) - (c), the overhead distributions are illustrated for each LT code ($k = 100$, $k = 500$ and $k = 1000$) — with an additional kernel fit to accentuate the trend of the overhead. For each simulation the Robust soliton PDD was used with parameter pair: $c = 0.1$ and $\delta = 0.99$. In (d) - (f), the cumulative distribution function (CDF) is plotted to indicate the probability at which the overhead distribution can be found under a certain value. Each experiment was performed 1000 times. These three tests demonstrate that the Robust Soliton PDD, coupled

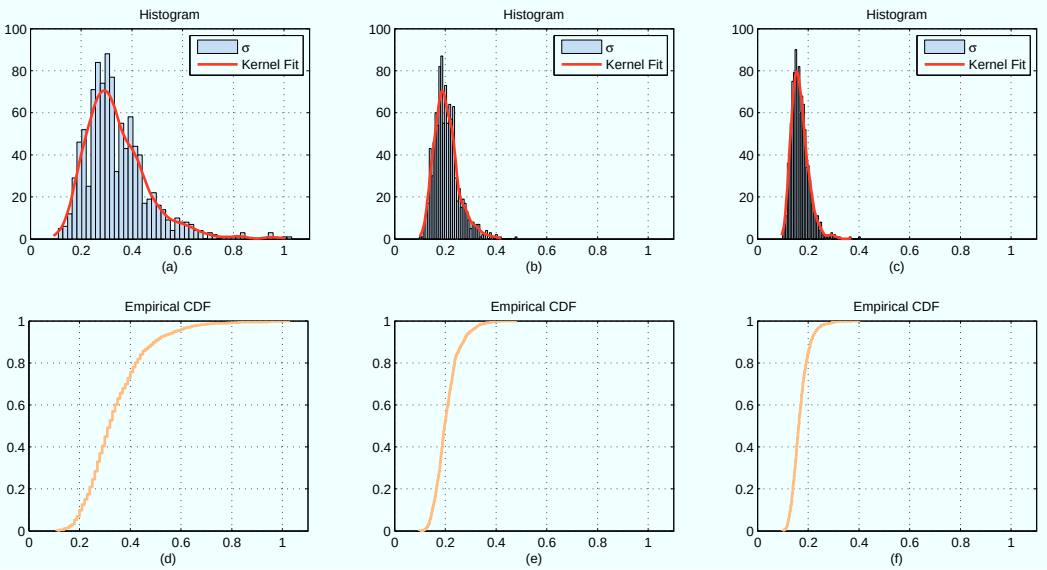

Figure 4.1: Overhead performance of the three candidate LT codes.

with the BP decoder, can provide some means of estimating the overhead to within reliable bounds. The overhead distribution, from the Robust Soliton PDD, is much more regular around the mean at higher frame sizes i.e., (b) and (c). It is also clear that the Robust Soliton PDD performs more efficiently at higher frame sizes, in terms of overhead. All three distributions are positively skewed, however, it becomes less pronounced at (b) and (c). When comparing these results to their theoretical expectations in figure 3.5 ($c = 0.1$ and $\delta = 0.99$), it is clear that the expected overhead becomes increasingly inaccurate at lower message frames (i.e., $k = 100$). This paradox will be explained in section 4.4.

In figure 4.2, a boxplot of the overhead performance of the three LT codes is presented. This figure illustrates the outliers, adjacent values, median, quartiles, and the maximum and minimum values of the overhead for each LT code. A summary of these results are given in table 4.1, which quantitatively describes the main features of the three LT codes. All the presented descriptive

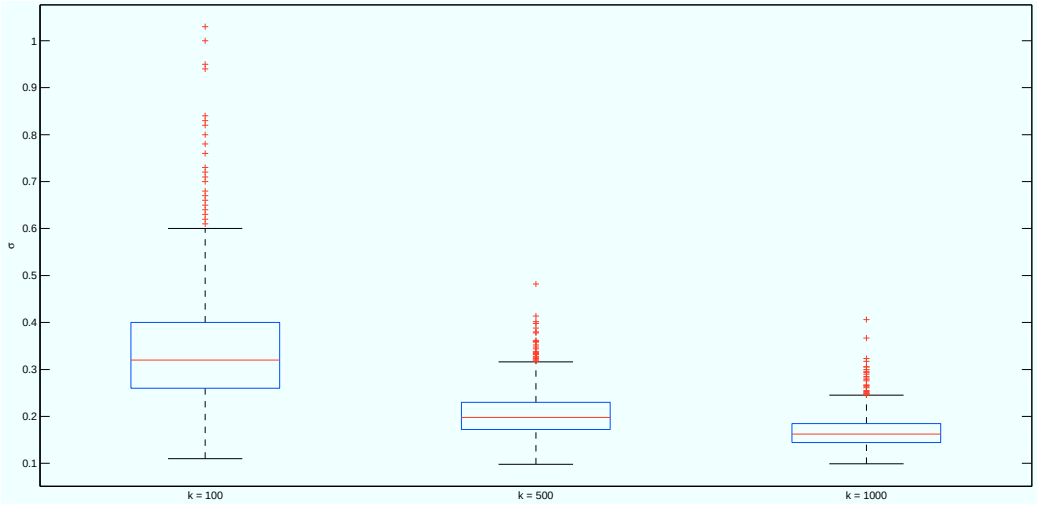

Figure 4.2: Overhead performance summary of the three candidate LT codes.

statistics, indicative of the overhead, shows a decreasing trend as k increases. Therefore, the efficiency and reliability of the LT code increases as k increases.

Table 4.1: Descriptive statistics summary of the three candidate LT codes.

k	100	500	1000
Outliers	40	34	30
max	1.03	0.48	0.40
Upper-adjacent	0.60	0.31	0.24
3de-Quartile	0.40	0.23	0.18
median	0.32	0.19	0.16
1st-Quartile	0.26	0.17	0.14
Lower-adjacent	0.11	0.09	0.09
min	0.11	0.09	0.09

A summary of the overhead performance between the two decoding schemes is indicated in table 4.2. Overhead results from the evaluation of the Gaussian decoder in section 3.3.3 is compared to the overhead results (from the CDFs)

from the BP decoder in figure 4.1. From these figures and the table it is clear that the Gaussian decoder outperforms the BP decoder by a large margin in terms of overhead.

Table 4.2: Summarised results for the three LT codes using Belief propagation and Gaussian elimination decoding.

Belief Propagation Decoding			
k	100	500	1000
N	180	680	1250
σ	0.8	0.36	0.26
Gaussian Elimination Decoding			
k	100	500	1000
N	110	507	1005
σ	0.100	0.014	0.005

The erasure correcting capability of the three designed LT codes was also tested. The graphs in figure 4.3 present the normalised results of the LT codes over a modulated BEC channel model as explained in definition 2.2 in section 2.1.2. All three candidate LT codes were tested to determine whether they remain rateless over a spectrum of erasure probabilities p. In this test, the encoder sends a stream of the encoded symbols. Random erasures are introduced based on the erasure probability while the decoder successively runs the BP decoding algorithm until decoding is completed. The decoded message is compared to the original message to validate successful decoding. The number of encoded symbols are then recorded for each erasure probability p. The result of this experiment is indicated by the average number of symbols required for successful recovery, and the upper and lower standard deviation bars to evaluate the fluctuations. A summary of the erasure channel performance of the three LT codes is presented in table 4.3. The average symbols and standard deviation is provided for the erasure probabilities ranging from $0 - 0.5$.

From the simulations shown in figure 4.3, it is clear that: as the erasure probability increases, the decoder will require more encoded symbols to recover the original message, since the number of erased symbols (dropped packets) also increases. This relationship should indicate a linear response over the entire erasure probability range in order for the universality property to apply. The presented results do indicate a linear response trend over the entire erasure probability range. However, at $k = 100$ the universality property becomes more sporadic.

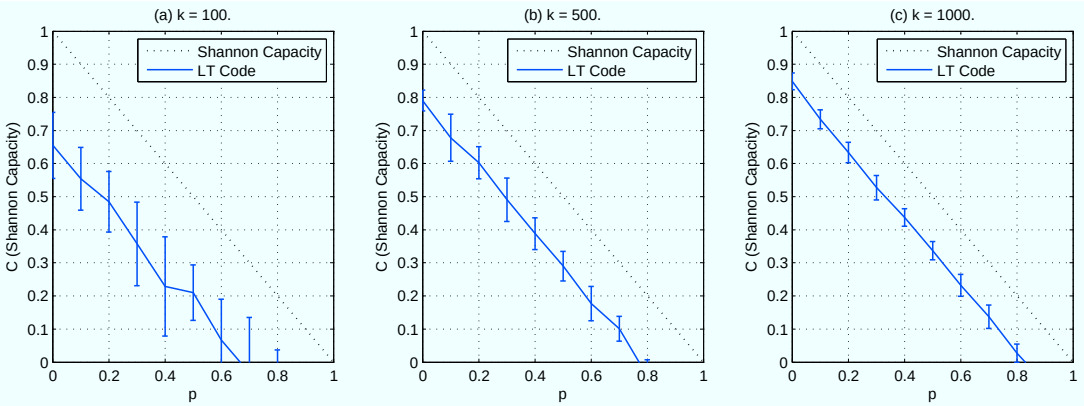

Figure 4.3: Erasure channel performance of the three LT codes.

Table 4.3: Summarised erasure channel performance of the three LT code.

$k = 100$						
p	0	0.1	0.2	0.3	0.4	0.5
$\mathrm{E}(N)$	132.9	142.2	151.6	167.7	171.3	182.5
$\mathrm{std}(N)$	10.3	14.8	7.8	9.5	9.9	9.0
$k = 500$						
p	0	0.1	0.2	0.3	0.4	0.5
$\mathrm{E}(N)$	598.5	652.6	701.6	744.8	807.8	852.1
$\mathrm{std}(N)$	27.3	27.1	26.1	27.1	25.1	24.3
$k = 1000$						
p	0	0.1	0.2	0.3	0.4	0.5
$\mathrm{E}(N)$	1165.8	1271.6	1387.8	1452.6	1574.4	1687.7
$\mathrm{std}(N)$	17.1	21.0	24.2	25.1	21.6	22.6

4.3 Raptor Codes

In this section, results from the three designed Raptor codes are presented. Three candidate Raptor codes with different message sizes ($n = 100, 500$ and 1000) was designed for the SensLAB. The overhead performance and the erasure correcting performance of these three candidate codes will be presented. Furthermore, it will be shown that the larger Raptor code ($n = 1000$) outperforms its smaller counterparts. Moreover, all three designed candidate Raptor codes conform to the rateless- and universality property of a true Fountain code.

According to the literature, presented in section 2.1.2 - 2.1.4 and section 2.3.3, the Raptor code can be considered as maximum distance separable (MDS) over the BEC. Therefore, the overhead σ is a very important measure

Table 4.4: Parameter selection summary of the three candidate Raptor codes.

n	100	500	1000
ϵ	0.58	0.26	0.18
v	$v_1 = \frac{2}{5}$, $v_2 = \frac{3}{5}$	$v_1 = \frac{2}{5}$, $v_2 = \frac{3}{5}$	$v_1 = \frac{2}{5}$, $v_2 = \frac{3}{5}$
h	$h_3 = 1$	$h_{13} = 1$	$h_{20} = 1$
r	0.76	0.84	0.82
m	130	566	1091
PDD	Ω_a	Ω_b	Ω_c
H-matrix	H_1	H_3	H_2
Estimated σ	0.3	0.13	0.09

of its performance with concern to a SensLAB implementation. The overhead is defined as the number of encoded symbols that need to be collected by the receiver in order to recover the source message symbols at high probability.

A summary of the design parameters selected for these three codes are indicated in table 4.4.

In figure 4.4 the overhead performance of the three Raptor codes are shown. In (a) - (c), the overhead distributions are illustrated for each Raptor code ($n = 100$, $n = 500$ and $n = 1000$) — with an additional kernel fit to accentuate the trend of the data. The designed H-matrices from section 3.4.1, and the optimised PDDs from table 3.2 in section 3.4.2 was used in the design of the three candidate Raptor codes. In (d) - (f), the CDF is plotted to indicate the probability at which the overhead distribution can be found under a certain value. Each experiment was performed 1000 times. These three tests demonstrate that the inner LT code, coupled with the BP decoder, and the outer LDPC code, coupled with the erasure-correcting BP decoder, can provide some means of estimating the overhead to within reliable bounds. The optimised PDD coupled with the PEG algorithm allows the Raptor code to exhibit much more regular overhead around the mean at higher frame sizes i.e., (b) and (c). It is also clear that the Raptor code performs more efficiently at these higher frame sizes, in terms of overhead. All three distributions are positively skewed, however, it becomes less pronounced at (b) and (c). When comparing these results to their theoretical expectations in table 4.4, it is clear that the expected overhead, m, are very close to the actual simulated overheads for all three designed Raptor codes.

In figure 4.5, a boxplot of the overhead performance of the three Raptor codes is presented. This figure illustrates the outliers, adjacent values, median, quartiles, and the maximum and minimum values of the overhead for

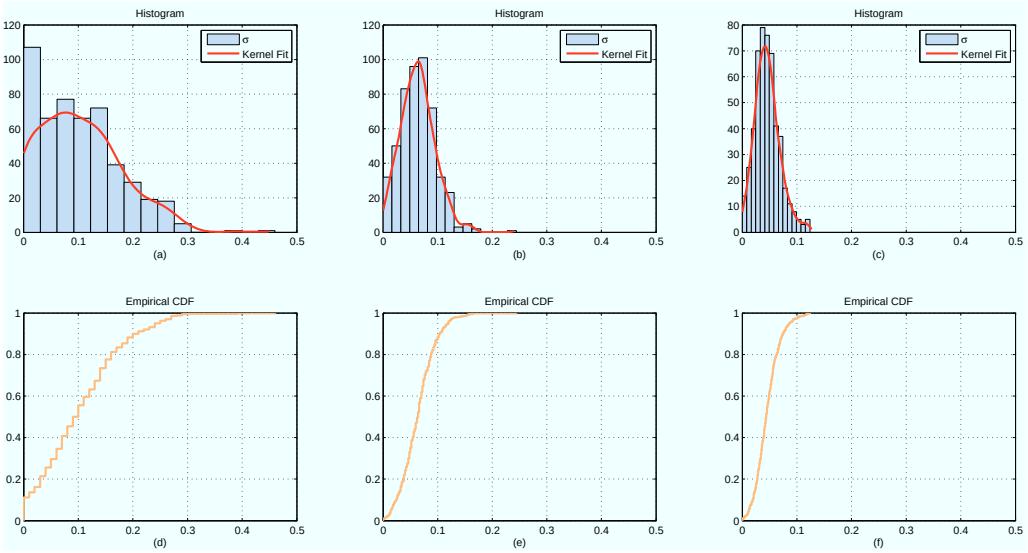

Figure 4.4: Overhead performance of the three candidate Raptor code.

each Raptor code. A summary of these results are given in table 4.5, which quantitatively describes the main features of the three Raptor codes.

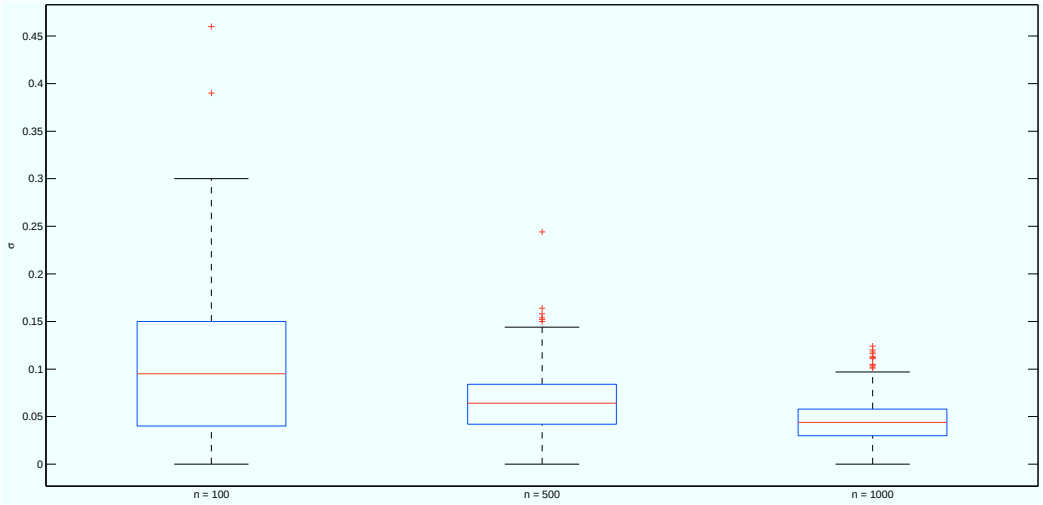

Figure 4.5: Overhead performance summary of the three candidate Raptor codes.

Most of the presented descriptive statistics, indicative of the overhead, shows a decreasing trend as n increases apart from a few outliers. Therefore, the efficiency and reliability of the Raptor code increases as n increases. The erasure correcting capability of the three designed Raptor codes was also tested. The graphs in figure 4.6 present the normalised results of the Raptor codes over a modulated BEC channel model as explained in definition 2.2 in section 2.1.2. All three candidate Raptor codes were tested to determine

Table 4.5: Descriptive statistics summary of the three candidate Raptor codes.

n	100	500	1000
Outliers	2	8	13
max	0.46	0.24	0.12
Upper-adjacent	0.3	0.14	0.09
3de-Quartile	0.15	0.08	0.05
median	0.09	0.06	0.04
1st-Quartile	0.04	0.04	0.03
Lower-adjacent	0	0	0
min	0	0	0

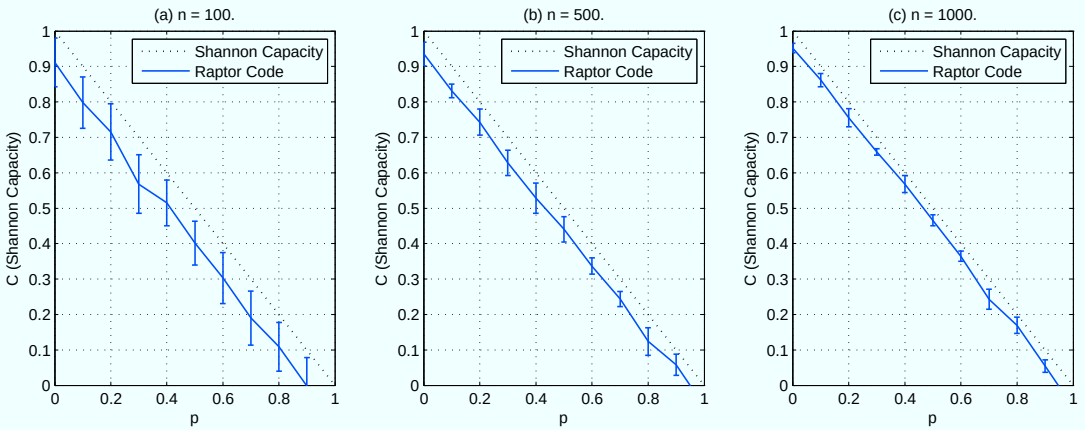

Figure 4.6: Erasure channel performance for the three candidate Raptor codes.

whether they remain rateless over a spectrum of erasure probabilities p. In this test the encoder sends a stream of the encoded symbols. Random erasures are introduced based on the erasure probability while the two decoders successively runs the BP decoding algorithms (LT code decoding and LDPC code decoding) until decoding is completed. The decoded message is compared to the original message to validate successful decoding. The number of encoded symbols are then recorded for each erasure probability p. The result of this experiment is indicated by the average number of symbols required for successful recovery, and the upper and lower standard deviation bars to evaluate the fluctuations.

A summary of the erasure channel performance of the three Raptor codes is presented in table 4.6. The average symbols and standard deviation is provided for the erasure probabilities ranging from $0 - 0.5$.

From the simulations shown in figure 4.6, it is clear that: as the erasure probability increases, the decoder will require more encoded symbols to recover

Table 4.6: Erasure channel performance of Raptor code.

$n = 100$						
p	0	0.1	0.2	0.3	0.4	0.5
E(N)	108.8	120.2	128.5	143.2	148.5	159.9
std(N)	6.9	7.2	8.2	6.4	6.2	7.1
$n = 500$						
p	0	0.1	0.2	0.3	0.4	0.5
E(N)	628.5	686	735.8	779.7	831.6	878
std(N)	16.6	9.7	18.4	17.8	21.4	17.9
$n = 1000$						
p	0	0.1	0.2	0.3	0.4	0.5
E(N)	1048.0	1138.4	1244.4	1341.0	1432.0	1534.2
std(N)	13.0	18.5	25.8	8.8	24.1	15.5

the original message, since the number of erased symbols (dropped packets) also increases. This relationship should indicate a linear response over the entire erasure probability range in order for the universality property to apply. The presented results do indicate a linear response over the entire erasure probability range.

In accordance to the design in section 3.4.1 - 3.4.3, it turns out that the PEG algorithm works well at shorter message frames ($n = 100$), which is a favoured outcome, since the inner LT code struggles with consistent decoding at smaller lengths. In this case, the PEG algorithm was used to create a H-matrix of a slightly lower code-rate (see table 4.4). This allowed the outer LDPC code to fix more erasures introduced by the channel, and the LT code's un-decoded symbol irregularities.

4.4 Summary

In this chapter, the results of the two candidate group FEC codes for SensLAB were presented. These included results from an overhead performance test and an erasure channel performance test. The data produced from these tests were inspected by using descriptive statistical tools. Moreover, a kernel smoother function was added to the histograms to help determine whether any irregularities existed in the data distribution, and to help indicate possible skewness of the data distribution. An accompanying empirical distribution function (the CDF plots) was also provided. This function estimates the true underlying CDF of the data and was used to determine the overhead necessary for complete decoding (at a probability of 1).

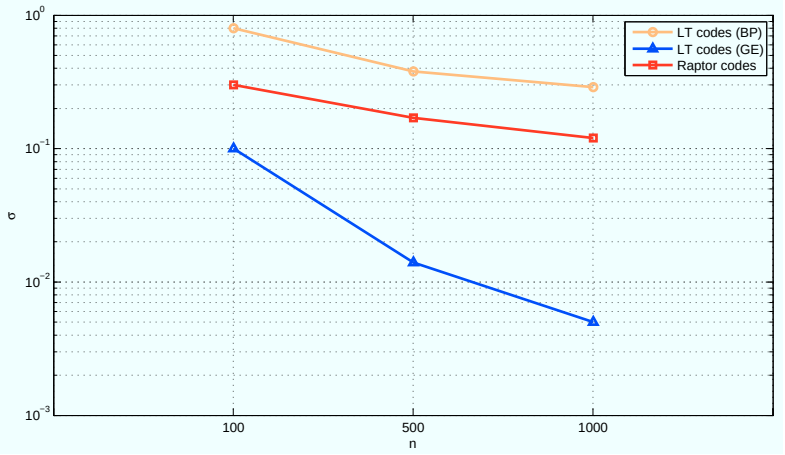

Figure 4.7: Overhead performance summary of all three candidate Fountain codes.

A summary of the expected overhead performance of the two LT code decoders and the Raptor code is given in figure 4.7. The LT code performance of the two decoding schemes (GE and BP), and the Raptor code performance is shown. The overhead values at the three different message sizes for each code is presented. These overhead values were taken from the CDF graphs in figure 3.6 - 3.8 and figure 4.1 and figure 4.4 at 1 convergence (where the highest likelihood is for successful decoding).

From this summarised overhead performance it is clear that the LT code using GE decoding outperforms the other two codes by a large margin in terms of σ. In addition, the overhead fraction for the GE decoder decreases significantly as the message size increases (from $n = 100$ to $n = 1000$), in comparison to the other two codes. The GE overhead performance is used here as an indication of near-optimal performance behaviour of a Fountain code. Furthermore, it is shown that the Raptor code outperforms the the LT code, that uses the BP decoding, over all three message sizes. However, it does not perform near optimal compared to the GE curve over these shorter message sizes.

The lower efficiency for smaller message frames may be explained by the law of large numbers (LLN). This is what complicates the transition from the theory to the actual practical and efficient constructs of Fountain codes that uses BP decoding. The LLN is a theorem that describes the result of performing the same experiment a large number of times. In accordance to the theorem, more trials will result in tighter bounds (less deviation) around the mean value of the result of the experiment. In other words, at smaller message frames less samples are taken from the PDDs, which makes it less likely to perform as initially designed (according to the theory). In general, this breaks down the theoretical estimations on Fountain codes using BP decoding alone. Concretely, for larger message frames more samples are taken from the PDDs

resulting in tighter bounds around the expected performance, and results in higher efficiency and reliability (see figure 3.4). This effect can be seen by examining figure 4.1 and figure 4.4, and table 4.1 and table 4.5. In the case of the Raptor code the outer LDPC code can be set up to remedy this issue as shown in this thesis.

Chapter 5

Conclusions, Contributions and Recommendations

5.1 Conclusions

The work described in this thesis, set out to identify a possible FEC strategy for the SensLAB platform. The candidate FEC code needed to conform to the SensLAB infrastructure constraints, and provide possible extended network related applications for the SensLAB community. An investigation of the SensLAB hardware, available embedded operating systems, and available software, revealed that these sensor nodes have limited energy consumption, low memory, modest computational resources, and limited transport layer frame sizes.

A literature study was presented, which considered two possible channel models for SensLAB, and their accompanying channel capacity performance bounds. The BEC approach (erasure-resilient) was selected as it afforded simplicity and low computational requirements with regard to a suitable decoding algorithm for SensLAB. Moreover, this channel model do not require quantified signal-reception information, which poses difficulty in obtaining that from the current sensor node radio modules.

Traditional coding schemes were considered, however, such strategies rely on ARQ feedback resources when a frame is corrupt beyond the correction capability of the FEC scheme. Furthermore, these kind of implementations exhibit high computational complexity with regard to the encoding and decoding algorithms, and do not accommodate network coding related extensions for the SensLAB community. As a result, these arguments motivated an investigation for alternative modern FEC schemes, such as LDPC codes and Fountain codes.

A literature study on LDPC codes and Fountain codes were presented. These codes are characterised by sparse graphical representations and predisposed to rigorous analysis, specifically with regard to their decoding mech-

anisms. Two FEC strategies were identified for SensLAB: a LT code and a Raptor code. It was shown that both these codes can be considered as MDS over the BEC channel, and as such, their overhead fraction is considered as an important performance indication. Concretely, it was shown that Fountain codes are considered as asymptotically optimal and require very large message sizes to realise their theoretical expectations. As a consequence the general consensus view on smaller message sized Fountain codes, in a practical setting, seem unappealing — as performance estimations for smaller messages are more complicated, and not really feasible in terms of efficiency. This fact was kept in mind throughout the design process of the candidate FEC codes for SensLAB.

Two possible decoding mechanisms were considered for the LT code: *belief propagation* (BP) decoding and *Gaussian elimination* (GE) decoding. It was shown that the latter can produce exceptionally low overheads, however, it is computationally expensive and not recommended for SensLAB at higher message sizes. The BP decoding mechanism produced higher overheads, however, it relies on a low complexity iterative decoding algorithm, which favours the SensLAB's hardware restrictions. The theoretical bounds on decoding overhead as a function of the decoding failure probability were compared to actual simulations to determine the accuracy of these estimates at smaller message frames.

The presented LT code, coupled with the BP decoder, required large overheads (introducing high encoding and decoding computation) and the results indicated a sporadic trend in the produced overhead distributions. This may have unacceptable consequences for SensLAB, since it can introduce redundant and unexpected data transmissions between nodes. This will in all probability cause problems in an energy and communications constrained environment. A Raptor code design was considered to alleviate this tendency, although a standard Raptor code design procedure could not be used for the SensLAB. Nonetheless, two methods were identified and used by generalising and adapting standard LT code and LDPC code analyses, which have proven useful in formulating a functional, and close to near optimal small message size Raptor code. In addition, these methods allowed the Raptor code to be scalable over a range of three different message sizes, which will benefit the SensLAB community in terms of application adaptability.

Results indicated that the Raptor code (in all three cases) ensures a more regular and more efficient decoding process than its counterpart, the LT code. The Raptor code was designed to have a more constant decoding overhead, which lowers its computational complexity, and increases its reliability in terms of symbol overhead. The increase in efficiency is accomplished by relaxing the requirement of the inner LT code (see section 3.4.2 and section 3.4.3), where only a fraction of the input symbols need to be recovered. The extended pre-coding (LDPC code) allows for the supplementary erasure-correction capability (see section 3.4.1). Subsequently, the outer LDPC code can be utilised

as a FEC strategy on its own (depending on the SensLAB application requirements), but will require a supplementary ARQ scheme. However, the key property making Fountain codes so attractive for SensLAB is that: for every erasure probability any set of received encoded symbols can be used to replicate the original message. This was a key result from the designed Raptor codes as both the rateless- and universality property were still validated, considering the supplementary design techniques. Conceptually, this can be a favoured outcome in terms of functionality in a *wireless sensor network* (WSN) environment.

When considering the SensLAB hardware it was seen that the transceiver modem is capable of providing error-free frames to the application layer, since it has built-in *cycle redundancy check* (CRC) detection. From the available operating systems and accompanying software, the *Contiki OS* coupled with the *uIP* stack seems eminently suitable to host the designed codes. These platforms were specifically designed for networking applications, and is capable of threading to host the decoding algorithm. The *protothread* library coupled with the uIP stack allows for UDP implementation over the application layer, or the *Rime stack* can allow for the codes to be implemented through individual modules such as *abc*, *broadcast*, *unicast*, or *stunicast*.

These available SensLAB options may allow for the integration of Fountain codes. Alternatively, each of the designed candidate codes can be modified by using the methods and techniques provided in this thesis. The presented design framework permits this kind of scalability within reasonable parameter settings.

5.2 Applications related contributions

This thesis has addressed a unique design framework that can be suitably configured to develop LT codes and Raptor codes of variable lengths; to accommodate applications that require specific transport frame sizes over the application layer of SensLAB.

The research area of Fountain codes is still relatively new and many untouched topics concerned with the utilisation of their properties, in a practical network setting (i.e., network coding), still require exploration. This thesis addressed the problem of a Fountain code design, particularly in the context of a WSN setting, applicable to the SensLAB platform, in an effort to support such attempts. Moreover, a contribution was made by creating a design framework capable of producing a close to near optimal short message size Raptor codes, and a simulator that can evaluate its performance. The underlying principles and conceptual statements of the codes were considered throughout the design process, and evaluated in isolation before final integration in the simulator. The presented parameters were considered critical, and evaluated to ensure high performing implementations for SensLAB. The work done under

this project did not necessarily set out to expand the very interesting theoretical base for Fountain codes, as described by key researchers. The simulations and tests produced in Matlab script during this project, can now be quite easily applied by other platform users, by simply translating the required scripts into appropriate node coding.

It was shown that the PEG algorithm coupled with a LP optimisation technique can produce high performing short message length Raptor codes, which can provide a more efficient and practical candidate Fountain code for SensLAB. This version still conforms to the rateless- and universality properties indicative of a true Fountain code.

Current focus by researchers using SensLAB, is on energy efficient routing techniques. Currently, fairly simple BSC related FEC coding is implemented in some cases to increase efficiencies in a typically constrained WSN environment. The work done under this project did not necessarily intend to expand on the very interesting Fountain coding theory developed by key researchers. However, not too much of this work has filtered through to the applications domain yet. The practical availability of a set of tested and optimised modern BEC codes as herein developed, will provide a new set of tools for use by WSN researchers utilising the SensLAB platform. It should certainly enhance and assist these general research efforts.

5.3 Recommendations for Future Work

The presented Fountain codes show many possibilities for SensLAB and other similar test environments, in terms of distributed and decentralised networks, including broadcasting and multicasting capabilities. The presented results concerned only a point-to-point transmission over a BEC channel environment and should be accompanied by the appropriate protocols available in the Rime stack. This will enable a point-to-multipoint transmission environment where a sender node can transmit data across the network to multiple node receivers. In such a setting the rateless- and universal properties of Fountain codes will come in handy; and will provide reliability in a scalable way. The same goes for a multipoint-to-multipoint transmission scenario, where a group of sender nodes, each possessing a piece of information, and a group of receiver nodes connected to a subset of those senders, receive the information (i.e., a peer-to-peer network environment).

In many cases, even though there is no back-channel in the transmission channel that is used for FEC, there are other back channels on the receiving devices, e.g., they can receive through broadcast but also communicate back through unicast. In other cases this is not possible and depends on the use case of the application. These schemes will depend on the protocol design. No feedback is simpler to manage and targeted for applications where feedback is not available, e.g., broadcast or multicast transmissions. The specific chan-

nel model applicable has, however, to be kept in mind. Here is an example of a proposed broadcasting network setup for SensLAB: Let n_1 denote the transmitting node, and n_i a set of receiver nodes. The nodes are configured with the *broadcasting*, *neighbour discovery* and *runicast* (the robust version of unicast) protocols from the provided *Rime* stack. When n_1 streams data to its n_i neighbouring nodes, each of the n_i nodes will be exposed to uniquely varying channel conditions that will evidently result in unique packet loss for each node. However, n_1 keeps broadcasting its encoded symbols until all of its known neighbours (using neighbour discovery) have replied with a stopping bit (using runicast), or until a maximum number of transmissions have been reached ($N \times z$, where $z \geq 1$).

Many other combinations of the Rime stack protocols can be set up to develop new network protocols. This motivates further research efforts to be directed to the implementation of the proposed candidate Fountain codes on SensLAB to extend the current network protocol to more feature-full application. From the presented Fountain codes, the larger message frame implementations for the LT- and Raptor code is recommended, due to their asymptotically optimal nature.

Appendices

Appendix A

A.1 Raptor Degree Distributions and Density Evolution

The popular degree distribution, proposed by Shokrollahi in [8, 22], serve as a good reference to explain the ideal decoding behaviour when using analytical tools such as Density evolution (DE), and optimisation tools such as linear programming (LP). The degree distribution below was developed by Shokrollahi by using similar LP optimisation techniques as described in the design section.

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

This degree distribution was derived to ensure that the decoding process will continue with high probability until it has recovered all but a fraction ς of the intermediate symbols. In this case, ς was chosen to be 0.01.

Efficient Raptor codes with good finite length behaviour usually have a message length in the order of 50000 or even much larger. This example will be concerned with a typical value for the message length, $k = 65536$. Figure A.1(a) shows the plots of the function $1 - x - e^{(1+\epsilon)\Omega'(x)}$ for multiple values of ϵ. From this figure it is clear that the decoding will fail prematurely if ϵ is chosen below 0.002. Asymptotically it is shown that this degree distribution can afford an overhead somewhere between $0.002k$ and $0.01k$.

Another parameter that becomes more pronounced with larger message lengths is the positive design parameter c. The larger c gets, the more likely it will be for the decoder to succeed in decoding all except the fraction ς of intermediate symbols. The plot of $1 - x - e^{(1+\epsilon)\Omega'(x)} - c\sqrt{\frac{1-x}{k}}$ for different c values are given in figure A.1(b). The value $c = 2.5$ intersect the x-axis fairly early, and will therefore not be a reliable choice. The inequality condition can be manipulated by using an appropriate c value, and adapting the degree

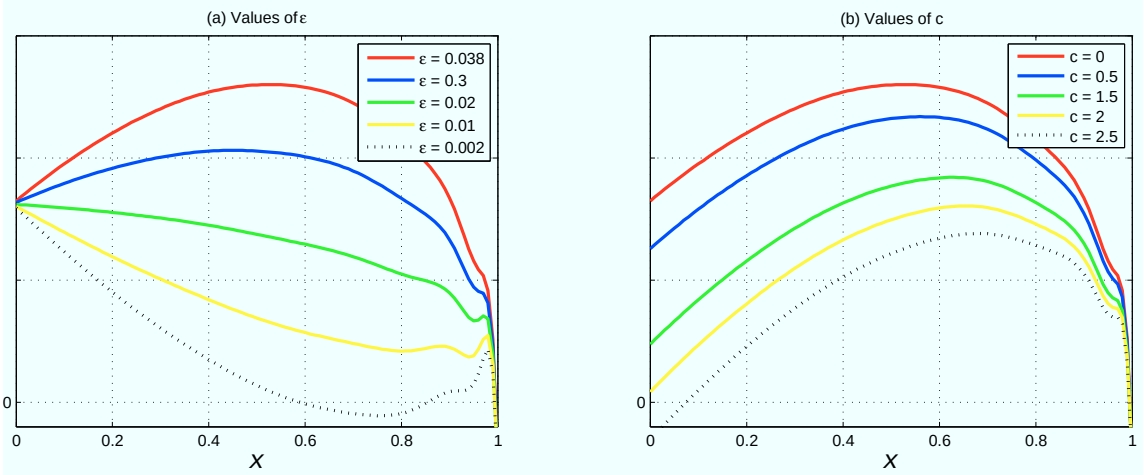

Figure A.1: Density evolution for $\Omega(x)$

Table A.1: Raptor distributions for various design values of k and ϵ.

k	65 536	80 000	100 000	120 000
Ω_1	0.007969	0.007544	0.006495	0.004807
Ω_2	0.493570	0.493610	0.495044	0.496472
Ω_3	0.166220	0.166458	0.168010	0.166912
Ω_4	0.072646	0.071243	0.067900	0.073374
Ω_5	0.082558	0.084913	0.089209	0.082206
Ω_8	0.056058	0.049633	0.041731	0.057471
Ω_9	0.037229	0.043365	0.050162	0.035951
Ω_{18}	0	0	0	0.001167
Ω_{19}	0.055590	0.045231	0.038837	0.054305
Ω_{20}	0	0.010157	0.015537	0
Ω_{65}	0.025023	0	0	0.018235
Ω_{66}	0.003135	0.010479	0.016298	0.009100
Ω_{67}	0	0.017365	0.010777	0
ϵ	0.038	0.035	0.028	0.02
a	5.87	5.91	5.85	5.83

distribution to enable optimal asymptotic behaviour. Table A.1 gives a good indication of several optimised degree distributions also found in [8, 22]. These distributions were designed by Shokrollahi using similar methods as presented in this thesis. The average degree is represented by the variable a.

List of References

[1] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[2] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 374–423, 1948.

[3] M. Luby, "LT codes," in *Annual IEEE symposium on foundations of computer science*, pp. 271–280, 2002.

[4] "SensLAB information on webpage." http://www.senslab.info/. Accessed: 2013-09-22.

[5] M. Luby, T. Stockhammer, and M. Watson, "IPTV Systems, Standards and Architectures: Part II-Application Layer FEC In IPTV Services," *Communications Magazine, IEEE*, vol. 46, no. 5, pp. 94–101, 2008.

[6] D. Minoli, *IP multicast with applications to IPTV and mobile DVB-H*. Wiley-IEEE Press, 2008.

[7] M. Luby, T. Stockhammer, and M. Watson, "Raptor FEC Schemes for FECFRAME," 2011.

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

[9] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, T. Gasiba, B. Furht, and S. Ahson, "Application layer forward error correction for mobile multimedia broadcasting," 2008.

[10] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[11] A. Bruen and M. Forcinito, *Cryptography, Information Theory, and Error-correction: A Handbook for the 21st Century*. Wiley-Interscience series in discrete mathematics and optimization, Wiley-Interscience, 2005.

[12] T. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Blackwell, 2005.

[13] B. Lathi, *Modern Digital and Analog Communication Systems*. The Oxford series in electrical and computer engineering, Oxford University Press, 1998.

[14] D. Pozar, *Microwave and RF Wireless Systems*. John Wiley, 2001.

[15] J. Andrews, A. Ghosh, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX: understanding broadband wireless networking*. Prentice Hall communications engineering and emerging technologies series, Prentice Hall, 2007.

[16] T. Tirronen, *Optimizing the Degree Distribution of LT Codes*. PhD thesis, Citeseer, 2006.

[17] P. Elias, "Coding for two noisy channels," in *Information Theory, Third London Symposium*, vol. 67, 1955.

[18] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *Information Theory, IEEE Transactions on*, vol. 42, no. 6, pp. 1732–1736, 1996.

[19] A. Shokrollahi, "Fountain codes," 2003.

[20] D. MacKay, "Fountain codes," 2005.

[21] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001.

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

[23] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *Information Theory, IEEE Transactions on*, vol. 24, no. 3, pp. 384–386, 1978.

[24] R. McEliece and H. van Tilborg, "On the Inherent Intractability of Finding Good Codes," *Deep Space Network Progress Report*, vol. 37, pp. 83–87, 1976.

[25] A. Tanenbaum, *Computer Networks*. Computer Science, Prentice Hall PTR, 2003.

[26] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[27] S. Wicker and V. Bhargava, *Reed-Solomon codes and their applications*. Wiley-IEEE Press, 1999.

[28] J. du Toit, O. Jumira, and W. R., "DTT Systems Comparison Study," 2010.

[29] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[30] D. MacKay, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 399–431, 1999.

[31] Y. Dai, Z. Yan, and N. Chen, "Optimal overlapped message passing decoding of quasi-cyclic LDPC codes," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 565–578, 2008.

[32] Y. Chen and K. Parhi, "Overlapped message passing for quasi-cyclic low-density parity check codes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 6, pp. 1106–1113, 2004.

[33] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619–637, 2001.

[34] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 249–258, ACM, 1998.

[35] S. Johnson, "Introducing Low-Density Parity-Check Codes," 2006.

[36] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 56–67, ACM, 1998.

[37] J. Byers, M. Luby, and M. Mitzenmacher, "A Digital Fountain approach to asynchronous reliable multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1528–1540, 2002.

[38] R. Tanner, "A recursive approach to low complexity codes," *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533–547, 1981.

[39] H. Tarus, J. Bush, J. Irvine, and J. Dunlop, "Exploiting redundancies to improve performance of LT decoding," in *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*, pp. 198–202, IEEE, 2008.

[40] H. Wang, "Hardware Designs for LT Coding," 2006.

[41] C. Harrelson, L. Ip, and W. Wang, "Limited randomness LT codes," 2003.

[42] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly Gaussian Elimination for LT codes," *Communications Letters, IEEE*, vol. 13, no. 12, pp. 953–955, 2009.

[43] F. Lu, C. Foh, J. Cai, and L. Chia, "LT codes decoding: Design and analysis," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pp. 2492–2496, IEEE, 2009.

[44] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 2576–2580, IEEE, 2006.

[45] C. Chen, Y. Chen, T. Shen, and J. Zao, "On the optimization of degree distributions in LT code with covariance matrix adaptation evolution strategy," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, IEEE, 2010.

[46] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. Harris, and M. Zorzi, "SYNAPSE: A network reprogramming protocol for wireless sensor networks using fountain codes," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*, pp. 188–196, IEEE, 2008.

[47] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimizing the degree distribution of LT codes with an importance sampling approach," in *RESIM 2006, 6th International Workshop on Rare Event Simulation*, Citeseer, 2006.

[48] "CC2420 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver." https://www.ti.com/lit/ds/symlink/cc2420.pdf. Accessed: 2013-09-22.

[49] "MSP430 Ultra-Low Power 16-bit MCU." http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/overview.page.

[50] "SensTools contiki." http://senstools.gforge.inria.fr/doku.php?id=os:contiki.

[51] P. Cataldi, M. Shatarski, M. Grangetto, and E. Magli, "Implementation and performance evaluation of LT and Raptor codes for multimedia applications," in *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on*, pp. 263–266, Ieee, 2006.

[52] C. Botha, "Designing Luby Transform Codes as an Application Layer Reliability Mechanism for Media Streaming over IP Networks," Master's thesis, University of Johannesburg, 2008.

[53] G. Wei, "Invertibility Probability of Binary Matrices."

[54] B. Rüffer and C. Kellett, "Implementing the Belief Propagation Algorithm in MATLAB," 2008.

[55] D. Sejdinovi, *Topics in Fountain Coding*. PhD thesis, University of Bristol, 2009.

[56] M. Luby, M. Mitzenmacher, and M. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pp. 364–373, Society for Industrial and Applied Mathematics, 1998.