Exploiting the implicit error correcting ability of networks that use random network coding

by

Suné von Solms

A dissertation submitted to the Faculty of Engineering

in partial fulfilment of the requirements for the degree

MAGISTER IN ENGINEERING

in

COMPUTER ENGINEERING

at the

NORTH-WEST UNIVERSITY – POTCHEFSTROOM CAMPUS

Supervisor: Prof ASJ Helberg

November 2009

Abstract

In this dissertation, we developed a method that uses the redundant information implicitly generated inside a random network coding network to apply error correction to the transmitted message. The obtained results show that the developed implicit error correcting method can reduce the effect of errors in a random network coding network without the addition of redundant information at the source node. This method presents numerous advantages compared to the documented concatenated error correction methods.

We found that various error correction schemes can be implemented without adding redundancy at the source nodes. The decoding ability of this method is dependent on the network characteristics. We found that large networks with a high level of interconnectivity yield more redundant information allowing more advanced error correction schemes to be implemented.

Network coding networks are prone to error propagation. We present the results of the effect of link error probability on our scheme and show that our scheme outperforms concatenated error correction schemes for low link error probability.

Keywords: Error correction; Network coding; Network error correction; Random network coding; Redundancy

Opsomming

In hierdie verhandeling word ‘n metode ontwikkel wat foutkorreksie op data in ‘n netwerk toepas. Oortollige inligting wat in ‘n netwerk gegenereer word, word gebruik om moontlike foute te korrigeer. Die resultate toon dat hierdie metode die invloed van foute in die betrokke netwerk kan verminder sonder dat oortollige inligting deur die versender saamgestuur word. Die metode bied vele voordele bo die reeds bestaande foutkorreksie metodes .

Daar is gevind dat verskillende foutkorreksieskemas deur die ontvanger-node geïmplementeer kan word sonder om oortolligheid by die versender by te voeg. Die dekoderingsvermoë van hierdie metode is afhanklik van verskeie netwerk eienskappe. Ons het gevind dat meer komplekse foutkorreksiekodes in ‘n netwerk geïmplementeer kan word indien die netwerk ‘n hoë vlak van interkonnektiwiteit besit.

Foutpropagasie is baie algemeen in netwerke wat netwerk kodering implementeer. Ons wys dat die voorgestelde metode die effek van foutpropagasie verminder in vergelyking met bestaande foutkorreksie skemas.

Sleutelwoorde: Foutkorreksie; Netwerk Kodering; Netwerk Foutkorreksie; Willekeurige Netwerk Kodering; Oortolligheid

Acknowledgements

I dedicate this dissertation to all those who made this possible:

My **BEST FRIEND**, Heavenly Father and *Guide* whose endless mercies follow me and whose goodness leads me home. I am not capable of achieving this – it is all because of YOU.

Prof. *Albert SJ Helberg* and Leenta Grobler who encouraged and helped me throughout these past two years. Thank you for the countless hours of listening, reading, planning and support. I cannot simply call you my supervisors, for you have become my friends.

Warren Linden for the endless prayers, support, love and patience. You give me the courage to pursue my dreams.

My parents, *Rossouw* and *Hester*, for the unconditional love, patience, understanding and support that got me to this point. Thank you for the endless phone calls and SMS's to check if I am still healthy, happy and okay. I do not know what I will do without you.

My sister, Woudi, for keeping me sane and reminding me that there is life outside studying.

My sisters in Christ, *Elzaan* and *Therese* for the prayers, visits and gym sessions that kept me going.

TeleNet Research group, especially *Melvin* and *Joubert*, whose support, advice and tons of coffee breaks got me through.

Telkom Centre of Excellence for the financial support. This research would not have been possible without you.

Declaration

I, ***Suné von Solms***, declare herewith that this dissertation entitled "***Exploiting the implicit error correcting ability of networks that use random network coding***", which I herewith submit to the North-West University in partial fulfilment of the requirements for the ***Master of Engineering*** degree, is my own work and has not already been submitted to any other university.

I understand and accept that the copies submitted for examination are the property of the North-West University.

_____ University number: 12987611

Signed at: ***Potchefstroom*** on this ***20th*** day of ***November 2009***.

Table of contents

ABSTRACT i

ACKNOWLEDGEMENTS iii

DECLARATION v

TABLE OF CONTENTS vi

LIST OF FIGURES xi

LIST OF TABLES xiii

LIST OF ADDREVIATIONS xiv

LIST OF SYMBOLS xv

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND 1-2

1.2 RESEARCH QUESTION 1-6

1.3 OBJECTIVES 1-7

1.4 METHODOLOGY OVERVIEW 1-7

1.4.1 Clearly define the problem 1-7

1.4.2 Postulate the important factors influencing the problem 1-7

1.4.3 Design conceptual and working model 1-9

1.4.4 Conduct experiment 1-9

1.4.5 Evaluate model and determine important factors 1-10

1.4.6 Revise working model 1-10

1.4.7 Conduct confirmatory experiment (collect additional data) 1-10

1.4.8 Verification and validation of model 1-10

1.4.9 Determine network requirements 1-11

1.5 ORGANISATION OF DISSERTATION 1-12

CHAPTER 2: GRAPH THEORY AND NETWORK CODING

2.1 GRAPH THEORY 2-2
2.1.1 Introduction 2-2
2.1.2 Network connectivity 2-2
2.1.3 Network diameter and average distance 2-4
2.1.4 Network capacity 2-4
2.1.5 Minimum cut and maximum flow 2-5
2.1.6 Symbol rate 2-7
2.2 DETERMINISTIC NETWORK CODING 2-8
2.2.1 A network coding example 2-8
2.2.2 Encoding 2-9
2.2.3 Decoding 2-10
2.3 RANDOM NETWORK CODING 2-11
2.3.1 Encoding 2-11
2.3.2 Decoding 2-13
2.4 ADVANTAGES OF NETWORK CODING 2-14
2.1.1 Throughput 2-14
2.1.2 Energy efficiency 2-15
2.1.3 Robustness 2-16
2.5 DISADVANTAGES OF NETWORK CODING 2-16
2.6 CONCLUSION 2-16

CHAPTER 3: NETWORK ERROR CORRECTION

3.1 INTRODUCTION 3-1
3.2 ERROR CORRECTION CODES 3-2
3.2.1 Forward error correction codes 3-3
3.2.2 Block codes 3-4
3.2.3 Linear block codes 3-5
3.3 NETWORK ERROR CORRECTION 3-8
3.3.1 Redundant packets 3-9
3.3.2 Redundant symbols 3-13
3.4 CONCLUSION 3-13

CHAPTER 4: IMPLICIT ERROR CORRECTION

4.1 INTRODUCTION 4-2

4.2 IMPLICIT ERROR CORRECTION METHOD 4-4

4.2.1 Implicit encoding 4-4

4.2.2 Error correction and decoding 4-6

4.2.3 Example 2-1(chapter 2) revisited 4-9

4.3 MODEL CONSTRUCTION 4-10

4.3.1 Network construction 4-11

4.3.2 Selection of network parameters 4-13

4.4 PROBABILITY OF DECODING 4-15

4.4.1 Simulation setup 4-15

4.4.2 Simulation results 4-17

4.5 ERROR CORRECTION ABILITY 4-18

4.5.1 Simulation setup 4-18

4.5.2 Simulation results 4-19

4.6 DISCUSSION 4-20

4.6.1 Advantages 4-20

4.6.2 Disadvantages 4-21

4.6.3 Observations 4-21

4.6.4 Validation and verification 4-22

4.7 CONCLUSION 4-22

CHAPTER 5: PARAMETER IMPROVEMENT

5.1 PROBABILITY OF DECODING 5-2

5.1.1 Selection of network parameters 5-2

5.1.2 Simulation setup 5-3

5.1.3 Simulation results 5-4

5.2 ERROR CORRECTION ABILITY 5-5

5.2.1 Simulation setup 5-5

5.2.2 Simulation results 5-5

5.3 DISCUSSION 5-6

5.3.1 Advantages 5-6

5.3.2 Disadvantages 5-6
5.3.3 Observations 5-7
5.4 CONCLUSION 5-7

CHAPTER 6: FINAL IMPROVEMENT AND ANALISIS

6.1 PROBABILITY OF DECODING 6-2
6.1.1 Selection of network parameters for improvement 6-2
6.1.2 Simulation setup 6-3
6.1.3 Simulation results 6-4
6.2 ERROR CORRECTION ABILITY 6-6
6.2.1 Simulation setup 6-6
6.2.2 Simulation results 6-8
6.3 DISCUSSION 6-9
6.3.1 Comparison of network error correction methods 6-10
6.3.2 Advantages 6-11
6.3.3 Disadvantages 6-12
6.4 CONCLUSION 6-13

CHAPTER 7: METHOD REQUIREMENTS, CHARACTERISTICS AND PERFORMANCE

7.1 INTRODUCTION 7-2
7.2 COMPUTATIONAL COMPLEXITY 7-3
7.2.1 Asymptotic Analysis 7-3
7.2.2 Computational characteristics 7-8
7.3 ERROR CORRECTING ABILITY 7-9
7.3.1 Simulation setup 7-9
7.3.2 Simulation results 7-12
7.3.3 Network *min-cut* requirement and characteristic 7-14
7.4 NETWORK REQUIREMENTS AND BER PERFORMANCE 7-14
7.4.1 Simulation setup 7-15
7.4.2 Selection of network parameters 7-18
7.4.3 Simulation results 7-19

7.4.4 Network requirements 7-23
7.4.5 Operational validation 7-24
7.5 CONCLUSION 7-24

CHAPTER 8: CONCLUSION

8.1 SUMMARY OF CONTRIBUTION 8-2
8.2 ACHIEVEMENT OF OBJECTIVES 8-2
8.2.1 Developing and constructing an error correction method that does not apply redundancy at the source 8-3
8.2.2 Verifying and validating the designed method 8-4
8.2.3 Determining the network requirements for the implimentation of the method 8-5
8.2.4 Commenting on the characteristics of this method 8-5
8.3 EVALUATION OF METHOD 8-6
8.4 FUTURE WORK 8-6
8.5 CONFERENCE CONTRIBUTIONS FROM THIS DISSERTATION 8-7
8.5.1 Engineering 8-7
8.5.2 Information Technology 8-8

REFERENCES R-1
APPENDIX A A-1

List of figures

Figure 1-1: Broad methodology overview 1-8

Figure 2-1: The butterfly network illustrating *min-cut* 2-6
Figure 2-2: Network with $min - cut = 3$ 2-7
Figure 2-3: Butterfly network 2-8
Figure 2-4: Information packets transmitted by source node 2-12
Figure 2-5: Network representation of min-cut requirements 2-12
Figure 2-6: Saving of energy in wireless network using network coding 2-15

Figure 3-1: Information packets with parity packet s transmitted by source node 3-10
Figure 3-2: Network representation of *min-cut* requirements 3-10
Figure 3-3: Representation of network error correction method 3-12
Figure 3-4: Information packet with redundancy symbols 3-13

Figure 4-1: Information packets without redundancy transmitted by source node 4-4
Figure 4-2: Network representation of *min-cut* requirements 4-5
Figure 4-3: Representation of proposed method 4-9
Figure 4-4: Network with $min - cut = 2$ 4-10
Figure 4-5: Probability of receiving valid parity packets 4-17
Figure 4-6: BER performance of the proposed method with Hamming decoding 4-20
Figure 4-7: Information packet in finite field $\mathbb{F}_2$ 4-22

Figure 5-1: Probability of receiving valid parity packets 5-4
Figure 5-2: BER performance of proposed method with LDPC decoding 5-6

Figure 6-1: Probability of receiving valid parity packets 6-5
Figure 6-2: BER performance of proposed method with RS decoding 6-8

Figure 7-1: Network error correction capacity network for specific *min-cut* 7-13
Figure 7-2: BER performance of network error correction schemes 7-20
Figure 7-3: BER performance in networks of size r 7-21
Figure 7-4: BER performance in networks with P_{le} 7-23

List of tables

Table 4-1: Probability of receiving valid parity packets 4-17
Table 4-2: Network parameters 4-23

Table 5-1: Probability of receiving valid parity packets 5-4
Table 5-1: Network parameters 5-8

Table 6-1: Probability of receiving valid parity packets 6-5
Table 6-2: Network size parameters 6-6
Table 6-3: Chosen network parameters 6-9
Table 6-4: Comparison of network error correction methods 6-10

Table 7-1: Summarisation of computational complexities 7-8
Table 7-2: BER performance in networks 4-22
Table 7-2: Requirements for implicit error correction method 7-25
Table 7-3: Characteristics of implicit error correction method 7-25

List of abbreviations

BER	Bit Error Rate
BSC	Binary Symmetrical Channel
FEC	Forward Error Correction
GF	Galois field
IEEE	Institute for Electrical and Electronics Engineers
LDPC	Low Density Parity Check
RS	Reed Solomon

List of symbols

$\boldsymbol{\alpha}$	global encoding vector
C	edge capacity
D_{ave}	average distance
D	network diameter
d_{ave}	average degree
d_{min}	minimum Hamming distance
$\delta_+(\ )$	in degree of node
$\delta_-(\ )$	out degree of node
E	permutation matrix
$\mathcal{E}$	set of edges
$\boldsymbol{e}$	error vector
$\mathbb{F}_q$	finite field
G	generator matrix
$\mathcal{G}$	directed, non cyclic synchronous network
$\Gamma_+(\ )$	input edge of node
$\Gamma_-(\ )$	output edge of node
$\mathcal{H}$	subset of network edges
$\mathcal{H}'$	subset of network edges connected to receiver node
h	network *min-cut*
k	dimension of linear block code
κ	node connectivity
λ	edge connectivity
$\boldsymbol{m}$	original message vector
n	length of block code
P_{le}	link error probability
p'	connectivity probability
R	symbol rate of network
r	network size

$\boldsymbol{r}$	transmitted codeword vector
$\boldsymbol{r}'$	received codeword vector
S	source node
$\boldsymbol{s}$	syndrome vector
T	overhead matrix
t	sink / receiver node
$\mathcal{V}$	set of nodes
$\mathcal{X}$	code alphabet
$\boldsymbol{x}$	transmitted packets
$\boldsymbol{y}$	received channel packets
$\mathcal{Z}$	source alphabet

Chapter 1:

Introduction

1.1 Background

The internet and other large scale communication networks are becoming a bigger part of our lives on a daily basis. New technologies are constantly being developed for the transmission of information over a wide variety of channels, varying from wireless channels to simple coaxial cables. The physical channels in these networks have different capacities and characteristics and new technology pursues to use each channel to its maximum capability.

Ahlswede et al. in [1] proposed that if intermediate nodes in a network are allowed to process the information they receive, the achievable rate of a multicast network can increase compared to straightforward routing. This approach, named network coding, requires the intermediate nodes to perform linear combinations on received information. In [1], the combinations formed by the nodes are based on a specific predefined topology.

The main advantages of network coding, compared to traditional routing techniques, are improvements in throughput, ease of management and a higher degree of robustness. These advantages are achieved by the better use of resources in the network where each node implemented with network coding receives the information from all the input nodes, encodes it and sends it out to the receiver nodes.

The concept of random network coding was introduced by Ho et al. in [2]. They present a randomized coding approach of information in networks that provides a series of advantages over traditional routing-based approaches. Their approach is to exploit the maximum capacity of the network by spreading the information over the available network capacity in order to keep the network flexible. By doing so, changes in the network's topology can be accommodated and therefore make the network more robust.

The improvement in robustness leads to the fact that the success of information reception does not depend on receiving packets that contain the specific transmitted information, but on receiving enough linear independent packets. Knowledge of the linear combinations of the information contained in each data packet is used to solve sets of simultaneous equations to obtain the transmitted data [3].

Ho et al. in [2] continued on the field of random network coding by describing it as follows: "*Network nodes independently and randomly select linear mappings from inputs onto output links over some field*." This means that all the nodes in the network, except the receiver node, perform independent random linear mappings of their inputs. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are formed from the inputs of the node. The outputs are chosen independently and randomly and must be non-zero.

The receiver node of the network obtains a series of independent linear combinations which it can use to decode the transmitted data. The receiver has to wait a certain amount of time in order to receive a set of equations that can be used to decode the transmitted message. The equations not used by the receiver are seen as redundant information and discarded. They prove that one of the benefits of this method is robustness to network changes and link failures.

In [4] Chou et al. commented that network coding environments are subjected to a variety of hostile factors like packet-losses, link failures and the occurrence of errors. According to [5] networks that implement network coding can be very sensitive to errors, defined by Cai and Yeung in [6] as an occurrence where the output symbol of a channel differs from the corresponding input symbol.

The need for reliability and the capability of networks to counter the effect of errors are therefore important characteristics of networks required today. These needs are widely addressed by implementing error correction in networks. Error correction for networks that implement network coding is of interest since 2002 when introduced by Cai and Yeung in [7].

An error correction code is able to correct and detect data packets corrupted due to additive errors and error propagation. Network error correction can improve the robustness of the network where the correct information can be obtained even when only partially correct information is received. Fragouli et al. in [8], translated the robustness of a network to the fact that successful information reception does not depend on receiving packets that contain specific information, but on receiving enough independent packets.

Yeung and Cai in [7] concluded by mentioning that the codes used for network coding are block codes. These codes are ideal types of codes to use, because they operate on packets of a fixed, predetermined size. Reed Solomon codes and Hamming codes are examples of block codes.

Lin and Costello [9] as well as Clark and Cain [10] define forward error correction as the addition of redundancy to the information sent. Linear block codes, a type of forward error correction code, are basically described in terms of a generator matrix and a parity-check matrix. They explain the different types of forward error correction codes and how each of them works, including the Hamming distance and Hamming weight of a codeword.

The broad spectrum of literature on classic error correction provides the fundamentals needed for the study of error correction in network coding. The field of network coding has progressed to the study of network error correction where this study aims to manage errors that occur in networks by detecting and correcting them. This will enable the receiver of the network to receive the correct information sent over the network.

An article on network coding and error correction [7] discusses classical error correction in point to point communication networks and looks at the correlation between that and network error correction. The authors show that network error correction is executed by using network coding, based on classical coding theory. They construct such a linear network code with error correction capabilities and show that in erroneous network channels, network error correction can be applied so that errors occurring in the network can be detected and corrected.

The different strategies and methods by different authors on error correction in random network coding provide insight into the different approaches for implementing error correction in random network coding. These insights on coding bounds, code constructions and error correction techniques developed by the authors supply the platform for further study in this field.

In 2007 Koetter et al. [11] presented error correction in random network coding where they introduced codes that are capable of correcting a series of errors and erasures by using forward error correction codes. They present a Reed Solomon code construction in random

network coding that provides codes that are capable of correcting different combinations of erasures and errors. This work formed a guideline for implementing forward error correction codes in random network coding. A “channel oblivious” random network coding network is considered where the transmitter and receiver are unaware of the method of the transferring of information.

In the above mentioned paper, the encoding of the error correction code is implemented at the transmitter of the network. This transmitter encodes the information to contain error correction capabilities, and sends it over the network. This code can then be corrected after receiving it at the end of the network.

Jaggi et al. [12] addresses the problem of error correction by extracting the source information from the received mixture of channel information and errors. They address this problem by adding redundancy to the source information that satisfies certain constraints and achieves optimal rates. Each packet contains a sequence of m symbols from the finite field $\mathbb{F}_q$. Out of the m symbols in the information packet; δm symbols are redundancy added by the source. The δm redundant symbols are chosen as parity symbols in order for the receiver to decode the channel packet

Yeung and Cai in [6] correct errors in network coding, not by adding redundancy to each information packet, but by adding redundant packets at the source to be sent over the network. This redundancy will enable the receiver node to correct network errors.

It can be seen, however, that the construction of the existing error correcting codes is in a concatenated form where error correcting encoding takes place prior to transmission.

The disadvantage of these schemes is that more than just information packets are injected into the network. The extra information needed for error correction must be generated at the source and this redundant information must be sent into the network. The encoding at the source and the injection of redundant packets also leads to greater energy consumption and is this not ideal for energy constraint networks, such as wireless sensor networks.

Looking at random network coding more closely, it can be seen that more equations are formed in the network than necessary. According to [11], the receiver node would normally

collect as many channel packets as possible in order to decode the source message. Because of network properties, such as the *min-cut* between source and receiver node, more than the needed linear independent equations are redundant information.

In this dissertation, we focus on using this redundant information collected by the receiver node to apply error correction to the transmitted message. This means that redundant information transmitted by the source node will no longer be necessary, because the network will transmit sufficient redundancy to the receiver for error correction.

1.2 Research Question

Silva et al. in [13] state that the main reason for interest in network error correction is the problem of error propagation in networks. Error propagation is an occurrence present in networks due to the inherent recombination characteristic of random network coding. A single packet corrupted by an error may continue to be transmitted through the network and contaminate other legitimate packets.

The network error correction methods described in the previous section address this problem of error correction in networks that use random network coding, but both these methods add redundancy to the source message transmitted over the network.

Random network coding environments have a tendency to generate redundancy inside the network which is normally discarded by the receivers. Redundancy is the foundation of network error correction and we aim to eliminate the need for the source node to encode data by exploiting this redundancy formed by the network to apply error correction. We found that currently no previous work was performed relating to the exploitation of the implicit encoding abilities of a network that implements random network coding.

This leads to our research question:

How can we reduce the effect of errors in random network coding by using the redundancy generated by a random network coding network?

1.3 Objectives

In order to answer this research question, we set the following objectives:

- To develop and construct an error correction method that does not apply redundancy at the source.
- To verify and validate the proposed method.
- To determine the network requirements for the implementation of the method.
- To comment on the characteristics of this method.

1.4 Methodology overview

The scientific method that is followed in this dissertation can be viewed in Fig. 1-1. This method is adapted from the "scientific or engineering method" described in [14].

1.4.1 Clearly define the problem

By studying the fields of network coding and forward error correction, we determine that there exists no documented network error correction method that addresses the problem of error propagation in random network coding without applying redundancy at the source nodes. The research question for this dissertation is defined in Chapter 1.2.

1.4.2 Postulate the important factors influencing the problem

We study the fields of network coding, with respect to various aspects of network characterisation; random network coding and network error correction in order to determine the factors that have an influence on this problem. The literature then focuses on forward error correction codes, their specific characteristics and the implementation thereof in random network coding. The literature is used to assist in the design of the implicit error correction method as well of the implementation thereof.

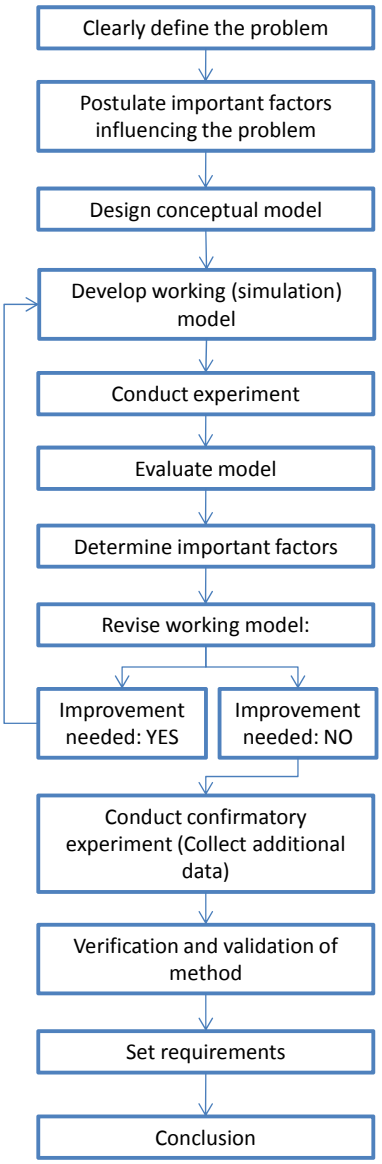


Figure 1-1: Broad methodology overview

The main network characteristics determined in the literature that may have an influence on our problem includes the following:

- The number of source and receiver nodes
- The size of the network
- The *min-cut* of the network
- The number of edge-disjoint paths in the network
- The characteristics and constraints of the source, receiver and intermediate nodes
- The size of the information packets that can be sent over each edge at a time

- Size and presence of packet headers
- The information rate of the network
- The probability of errors occurring in the network and on each link

The characteristics concerning forward error correction codes that may influence our problem include the following:

- Block size
- Finite field of data
- Linearity
- Minimum Hamming distance
- Information rate

1.4.3 Design conceptual and working model

The network characteristics and error correction algorithms are considered and a conceptual model is designed in compliance to those specifications. This conceptual model is based on the mathematical concepts in the literature in order to determine if the model is functional (if it can work) and to provide a basis for us to see how the model responds to each of the influential factors (size, connectivity etc.).

Next, a working model is developed from the conceptual model for simulation purposes. The implicit error correction method implemented in simulations is mapped from the conceptual (mathematical) design.

1.4.4 Conduct experiment

A basic parameter set is selected for the first iteration of this scientific process. A set of simulations is planned so that the influence of each of the factors in 1.5.2 can be determined. These experiments are designed so that the influential factors can be changed during the simulations in order to determine the exact influence of all the factors on the method.

Due to the non-deterministic nature of random network coding a large number of iterations have to be performed during the simulations. The results are obtained with respect to the mean value of each set.

1.4.5 Evaluate model and determine important factors

The experimental results obtained in the simulations regarding these factors (network information rate, connectivity, size, packet overhead, packet size etc.) are used to determine which factors are the most important and which have no significant influence on the performance.

1.4.6 Revise working model

The obtained results are used to revise and improve the working model so that better results can be obtained. The revised method is tested again on a simulated network and improved once more. Finally, a network parameter set is determined where the proposed method renders satisfactory results.

1.4.7 Conduct confirmatory experiment (collect additional data)

The proposed method is compared with existing network error correction methods in terms of computational complexity, network error correction capacity and Bit Error Rate (BER) performance at specific link error probabilities (P_{le}) and network size (r) to determine where this method can outperform existing concatenated schemes.

1.4.8 Verification and validation of model [15-19]

The Institute for Electrical and Electronics Engineers (IEEE) defines *verification* as the act of inspecting, testing, checking, or otherwise establishing whether or not a process conforms to specified requirements imposed at the start [15, 16]. Sargent in [17] defines *computerised model verification* as the act of assuring that the computer programming and implementation of the conceptual model is correct.

We will adopt this definition and verify the computerised model by determining if the model has been programmed correctly in the simulation language. To do so, we will use the static-testing procedure described in [17] where we use a structured walk-through to determine if the conceptual (mathematic) model is solved correctly by the implemented computer code.

The Institute for Electrical and Electronic Engineers (IEEE) defines *validation* as the evaluation of a model at the end of the development process to establish the compliance with the specified requirements [15, 16].

We see *compliance with the specified requirements* as the answering of the research question posed in Chapter 1.2. Thus we will validate the proposed method by checking whether the method **can reduce the effect of errors in random network coding without adding redundancy at the source.**

This will be done in two steps:

1. conceptual model validation and
2. operational validation.

Sargent in [17] defines *conceptual model validation* as the act of determining that the theory and assumptions underlying the conceptual model is correct; and that the model representation of the problem is reasonable for the intended purpose of the model.

We will validate the conceptual model by checking whether the conceptual design correctly implements the theory of random network coding, error correction and network error correction presented in the literature; and that it correctly addresses the problem stated in the research question.

Operational validation is defined in [17] as the act of determining that the model's output has sufficient accuracy for the model's intended purpose.

From the various sets of simulations done in this dissertation, we will be able to compare the results obtained from the implicit error correction method with that of the concatenated method present in the literature. These results will enable us to validate the model by determining if the model's output is sufficient for its intended purpose.

1.4.9 Determine network requirements

Finally, the specific network requirements for and characteristics of this method are determined. These requirements will assist us in determining when the proposed method can be successfully implemented in a network and be used as an alternative to existing concatenated network error correction schemes to render better results.

Once these network requirements are determined, we will be able us to evaluate a specific network scenario and determine if the implicit error correction method can be successfully implemented in the network to outperform existing concatenated network error correction schemes. We will also able to determine certain network characteristics associated with the implementation.

1.5 Organisation of dissertation

Chapter 2 and Chapter 3 represent the literature review of this dissertation.

In Chapter 2 we discuss graph theory, introduce the network model and the different network properties associated with it. This chapter further discusses deterministic and random network coding along with the advantages and disadvantages they present.

Chapter 3 contains an overview of the literature concerning network error correction in network coding. Forward error correction techniques are discussed, as well as the different implementation methods of these error correction codes in networks.

Chapter 4, Chapter 5 and Chapter 6 covers the design and improvement of the implicit error correction method and the network characterisation.

Chapter 4 presents the conceptual implicit error correction method developed in this dissertation. This method uses the redundant information obtained by the receiver node to apply error correction. This method is designed, a set of network parameters is selected and the obtained results are evaluated in order to learn if the proposed method can be implemented successfully and if it is an accurate representation of the conceptual model.

In Chapter 5, we improve the implicit error correction method discussed in Chapter 4. A new set of network parameters is selected and implemented in the network in order to obtain more favourable results. The obtained results are compared with the results obtained in Chapter 4 and evaluated.

In Chapter 6, this implicit error correction method is improved further. The characteristics of the method are summarised and compared with that of existing concatenated network error correction schemes. The chapter concludes by determining the advantages and disadvantages of the implicit error correction scheme.

In Chapter 7, we determine the network requirements needed for the improved method to be successfully implemented. The proposed method is evaluated along with the existing network error correction scheme and no error correction scheme by simulating it in the same network. We assess the computational complexity, error correction capacity and BER performance to determine the network requirements for the successful implementation of the implicit method.

This dissertation concludes in Chapter 8. The addressed research problem is analysed and presented, along with relevant literature on the problem. We discuss the methodology followed to achieve the main objectives and give an overview of the literature. We summarise the results obtained in this dissertation which include network requirements, method characteristics and performance. Finally, we discuss further work that may exist in the continuation of this field and publications written while completing this study.

Chapter 2:

Graph theory and network coding

In this chapter we discuss graph theory, introduce the network model and the different network properties associated with it. This chapter further discusses deterministic and random network coding along with the advantages and disadvantages they present.

2.1 Graph theory

2.1.1. Introduction

A graph can be considered as a mathematical structure that we use to model the relationships between different objects in a collection [20]. The graphs used in this chapter are collections of nodes and edges connecting the nodes. The graph theory discussed is modelled onto a network of the same characteristics, i.e. a network is a specific implementation of a graph.

We adopt the notation used in [2, 3], [21] of an acyclic network model. The network is represented by a directed, non-cyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in the network and $\mathcal{E}$ the set of edges in $\mathcal{G}$ which represents the communication channels. An edge from node a to b is indicated by $(a,b) \in \mathcal{E}$. Node a is called the input node of edge (a,b) and edge (a,b) is called the input edge of node b; while node b is called the output node of edge (a,b) and edge (a,b) is called the output edge of node a. For an edge $e = (a, b) \in \mathcal{E}$, the head and tail of an edge is denoted by $a = head(e)$ and $b = tail(e)$ respectively.

A non-cyclic network can be described as a network where no directed cycles are present and the sequence of edges $(v_0, v_1), (v_1, v_2), \dots, (v_{n-1}, v_n)$ exists in $\mathcal{G}$.

The set $\mathcal{Z}$ is the source alphabet and $\mathcal{X}$ is the finite set that serves as the code alphabet for the network where $\mathcal{X} \in \mathcal{Z}$ in a finite field $\mathbb{F}_q$. The source node $S \in \mathcal{V}$ transmits the source message into the network, where it must be received by the sink nodes $t = (t_1, \dots, t_n) \in \mathcal{U}$, where $\mathcal{U} \subset \mathcal{V}$, where $\mathcal{V}$ is the set of receiver nodes in the network.

2.1.2. Network connectivity [21]

The set of input and output edges of node $a \in \mathcal{V}$ can be denoted by $\Gamma_+(a) = \{(c, a): (c, a) \in \mathcal{E}\}$ and $\Gamma_-(a) = \{(a, b): (a, b) \in \mathcal{E}\}$, respectively. The in-degree of node a is defined as $\delta_+(a) = |\Gamma_+(a)|$, while the out-degree is defined by $\delta_-(a) = |\Gamma_+(a)|$.

If, for example, node a has d outgoing links, node a has a degree of d, or $\delta\_(a) = d$. The minimum degree, d_{min}, maximum degree, d_{max}, and average degree, d_{ave}, of the network is the smallest, largest and average degree of all the nodes in the network respectively.

In order to model graph robustness, two definitions of connectivity must be considered.

Definition 2-1:

- *Node connectivity* κ: the smallest number of nodes that must be removed in order for the graph, $\mathcal{G}$, to become disconnected.
- *Edge connectivity* λ: the smallest number of egdes that must be disconnected in order for the graph, $\mathcal{G}$, to become disconnected.

In this dissertation we only consider networks where connections are made completely at random. This means that the construction of networks is not dependent on physical distances between nodes. Using this model will enable us to construct optimally connected networks that have several advantages.

The Erdös-Rényi model [21, 22] states that when:

- a large enough graph is constructed
- where the connections are made at random
- with a fixed probability p' of an edge between any pair of nodes

an optimally connected network is created.

Definition 2-2 [21]: When $\kappa = \lambda = d_{min}$ in a graph, the graph is *optimally connected*, because the node and edge connectivities are as high as possible and the network is as robust as it can be, for a specific value of d_{min}.

Theorem 2-1 [22]: For any randomly generated graph, $\mathcal{G}$, of size r, the probability that $\kappa = \lambda = d_{min}$ approaches 1 as $n \to \infty$.

This theorem is tested in [21] with 200 000 random graphs of size $7 \leq r \leq 30$ and $d_{ave} = \sqrt{r}$. They found that the percentage of optimally connected graphs was:

94.8% for $r = 7$, to

99.98% for $r = 30$.

2.1.3. Network diameter and average distance [21]

A very important concept in graph theory is that of network diameter, D, and average distance, D_{ave}. The diameter, D, of a network is the longest of all the shortest paths between pairs of nodes, where the average distance, D_{ave}, is the average of all the shortest paths between pairs of nodes.

Normally, one would like the average distance of a network to be as small as possible, because long paths between source and receiver nodes lead to longer transmission time and higher probability of error propagation. For a network that must implement random network coding the average distance cannot be too small, because sufficient network coding will not take place. This will result in the receiver not obtaining enough linearly independent channel packets to decode successfully.

We must construct networks where information can be successfully encoded, but where error propagation is minimized. The next theorem will allow us to construct networks where the average distance, D_{ave}, between source and receiver node is satisfactory.

Theorem 2-2 [21, 22]: For a randomly generated graph, $\mathcal{G}$, of size n, the diameter will be approximately $D \geq 2\frac{\log(r-1)}{\log d}$, where the degree $d \geq 3$. The average distance is bounded by the diameter, where

$$\frac{rD}{2(r-1)} \leq D_{ave} \leq D. \tag{2-1}$$

2.1.4. Network capacity [22]

A directed graph with edge capacities is called a network. Assume $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a finite, directed graph where each edge $(a, b) \in \mathcal{E}$ has a positive, real valued capacity $C_{ab}, (a, b) \in \mathcal{E}$. The network then has the following properties:

- **Capacity constraint:** $f = (a, b) \leq C_{ab}\ \forall\ (a, b) \in \mathcal{E}$. The flow, f, along each edge in the network cannot exceed the capacity, C_{ab}, of the edge.

- **Skew Symmetry:** $f = (a, b) = -f(a, b)$. The total flow from node a to b, must be the exact opposite of the total flow from node b to node a.

- **Flow conservation:** $\sum_{a \in V} f(a, b) = 0$, unless $a = S$ or $a = t$. The content of information sent by an intermediate node must be derived from the collected information received by the node. This is known as the *law of information conservation*. The network must satisfy this restriction so that the amount of flow into a node must equal the amount of flow out of the node; except for source node, S, that only has outgoing flow and a receiver node, t, that only has incoming flow.

In this dissertation, we accept that all the edges of the network have unit capacity, therefore we assume that the capacity

$$C_{ab} = \begin{cases} 1 \text{ if } (a, b) \in \mathcal{E} \\ 0 \text{ otherwise} \end{cases}. \tag{2-2}$$

If edge $(a, b) \in \mathcal{E}$ has a capacity, $C_{ab} = z, z \in \mathbb{Z}$, we simply replace it with z links with unit capacity between node a and b.

2.1.5. Minimum cut and maximum flow [23]

One of the most important algebraic concepts used throughout this dissertation, is the *min-cut max-flow* theorem. Consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where source node $S \in \mathcal{V}$ transmits information over the network to receiver node $t \in \mathcal{V}$ over edges, $\mathcal{E}$, with unit capacity.

Definition 2-3: If $f = (a, b) = C_{ab}\ \forall\ (a, b) \in \mathcal{E}$ where the flow through each edge is as large as the capacity, the flow is called the *maximum flow* or *max-flow*.

Definition 2-4: A *cut* between node S and node t is a subset of graph edges, $\mathcal{H} \subset \mathcal{E}$, which removal will disconnect the two nodes. The *minimum cut* or *min-cut* of the graph is the smallest subset of edges, $\mathcal{H} \subset \mathcal{E}$, which removal will disconnect S from t. For a graph with unit capacity edges, the value (size) of a cut is equal to the number of edges in the subset, $\mathcal{H}$:

$$|\mathcal{H}| = min - cut = h. \tag{2-3}$$

A unique *min-cut* value, h, exists for each graph, as well as the possibility of multiple *min-cut* subsets. Consider the graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in Fig. 2-1. By inspection it can be seen that $min - cut = h = 2$, where the *min-cut* subsets are $(\{S, a\}, \{S, b\}), (\{a, t_1\}, \{d, t_1\})$ and $(\{b, t_2\}, \{d, t_2\})$. This network is traditionally known as the butterfly network.

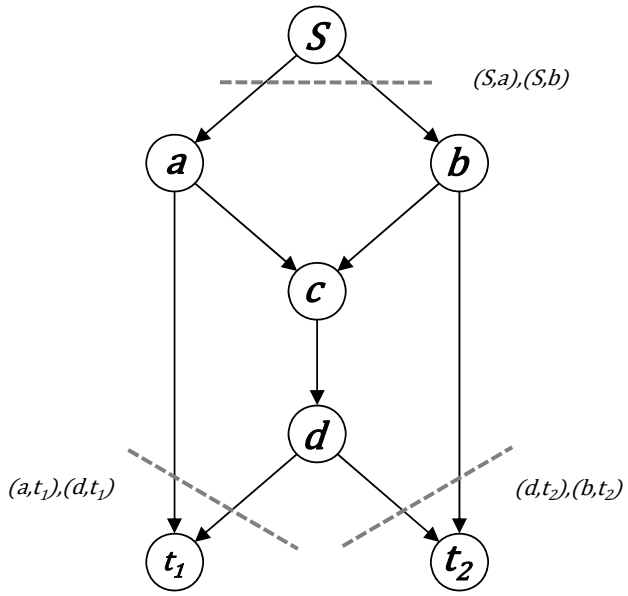


Figure 2-1: The butterfly network illustrating *min-cut*

Theorem 2-3: Min-cut max-flow [23]: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a single source, S, and receiver, T, is given. If the minimum cut between nodes S and t is $min - cut = h$, then information can be sent from source node to receiver node at a maximum rate equal to h.

Also, there exist exactly h edge-disjoint paths between S and t when the *min-cut* between them is h. This can be seen easily, because if more than h edge-disjoint paths exist, the removal of the h edges will not disconnect S from t.

This theorem is proved in [23].

Example 2-1 [23]: This example illustrates the concept of *min-cut* and edge-disjoint paths. Fig. 2-2 shows a network with unit capacity edges. Between the source node, S, and receiver node, T, $min - cut = 3$. This means that there exist three edge-disjoint paths (that can clearly be seen) between S and T where the receiver obtains symbols x_1, x_2 and x_3.

If this network did not have a *min-cut* of at least 3, it would have been unable to support the transmission of the 3 symbols.

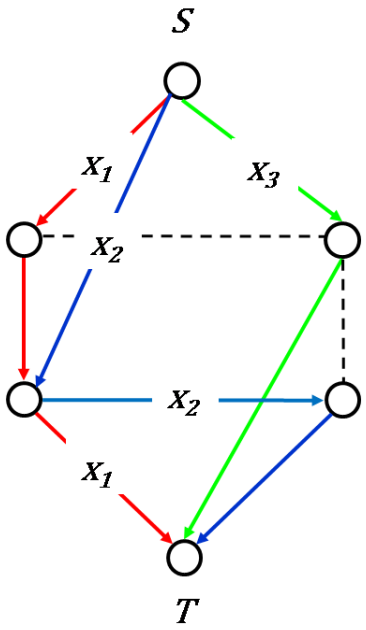


Figure 2-2: Network with $min - cut = 3$

2.1.6. Symbol rate

A very important concept used throughout this dissertation is the relationship between information symbols sent from the source node into the network and channel symbols received by the receiver from the network. We define this relationship as the *symbol rate*, R, of the network which indicates the relationship between the number of transmitted packets and received packets, where

$$R = \frac{transmitted\ symbols}{received\ symbols}. \tag{2-4}$$

Again, consider Example 2-1 where symbols x_1, x_2 and x_3 are generated at the source node, S and collected by receiver, T. The symbol rate of this network is:

$$R = \frac{transmitted\ symbols}{received\ symbols} = \frac{3}{3} = 1. \tag{2-5}$$

2.2 Deterministic network coding

Network coding is a technique introduced in [1] that aims to improve network throughput and performance. Yeung et al. in [1] realized that if the nodes in a network process the information it receives and not just forward it, a wide range of benefits can be achieved.

In network coding, the intermediate nodes in the network form linear combinations of the received information. This operation is not a concatenation of information, but means that if packets of size k are linearly combined, the resulting coded packet still has the size k.

2.2.1. A network coding example [7]

Fig. 2-3 represents a communication network, known in the network coding literature as the butterfly network, as an example of deterministic network coding. Assume that time slots have been provided for this network and that each channel can send one bit of data per time slot.

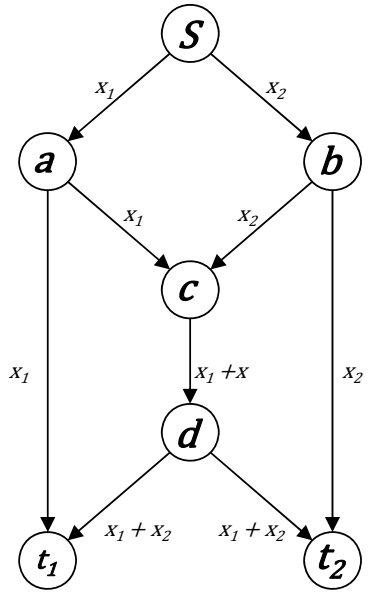


Figure 2-3: Butterfly network

The network consists of a single source node, S, and two receivers, t_1 and t_2. The source node transmits two bits, x_1 and x_2, into the network.

For t_1 to receive both source bits, bit x_1 can be sent from S, along edge $\{at_1\}$ to t_1. Bit x_2 can be sent from S, through edges $\{bc, cd, dt_1\}$ to t_1. The receiver t_1 has received both source bits, but used the network for itself.

Similarly, receiver t_2 can also receive both messages when using all network resources by itself: routing x_2 through edge $\{bt_2\}$ and x_1 through $\{ac, cd, dt_2\}$.

Assume that both receivers want to receive x_1 and x_2 simultaneously. Edge $\{cd\}$ forms a bottleneck in the network, because only one bit can be sent per time slot. By applying traditional routing, a decision has to be made at node c: either send bit x_1 or bit x_2. If bit x_1 is sent, then receiver t_1 will only receive x_1; while receiver t_2 will receive both bits and vice versa.

By implementing network coding as described in [1], node c can take the received bits, x_1 and x_2, and perform a bitwise x-or (i.e. linear combination) with them to create a third bit $x_3 = x_1 \oplus x_2$ (where the $\oplus$ sign is addition in $\mathbb{F}_2$ or x-or). Bit x_3 is then sent over edge $\{cd\}$. Receiver node t_1 receives $\{x_1, x_1 \oplus x_2\}$ that can be solved to retrieve x_1 and x_2. Similarly, t_2 retrieves x_1 and x_2 by solving $\{x_2, x_1 \oplus x_2\}$.

2.2.2. Encoding [2, 3] [23]

Although not shown in the butterfly network example, encoding can be performed recursively. This means that already encoded packets can be linearly combined with other encoded packets.

Assume that a packet contains a sequence of m symbols from the finite field $\mathbb{F}_q$. The source node, S, transmits k information packets, $x_1, x_2, \dots, x_k$ into the network.

Deterministic network coding (based on the topology of the network) is implemented in the network nodes where predetermined coefficients from a finite field $\mathbb{F}_q$ are selected. Each channel packet, y_i, formed in the network is a linear combination of all or selected packets $x_1, x_2, \dots, x_k, x_i \in \mathbb{F}_q$,

$$y_i = \sum_{\kappa=1}^{k} \alpha_{i\kappa} \boldsymbol{x}_\kappa \ , \mathrm{i} = 1, 2, \dots, k \tag{2-6}$$

where α is called the global encoding vector of $\boldsymbol{x}$.

2.2.3. Decoding [2, 3] [23]

Assume that k' channel packets, $y_1, y_2, \dots, y_{k'}$, are received by the receiver node, where $k' = k$:

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_{11}\boldsymbol{x_1} + \alpha_{12}\boldsymbol{x_2} + \cdots + \alpha_{1k}\boldsymbol{x_k} \\ \boldsymbol{y_2} &= \alpha_{21}\boldsymbol{x_1} + \alpha_{22}\boldsymbol{x_2} + \cdots + \alpha_{2k}\boldsymbol{x_k} \\ &\vdots \\ \boldsymbol{y_{k'}} &= \alpha_{k'1}\boldsymbol{x_1} + \alpha_{k'2}\boldsymbol{x_2} + \cdots + \alpha_{k'k}\boldsymbol{x_k} \end{aligned} \tag{2-7}$$

and $\boldsymbol{x_i}$, $i = 1, 2, \dots, k$ are the k information packets sent from the source and $\alpha_i, i = 1, 2, \dots, k$ are the predetermined coefficients. Equation (2-7) can also be written as

$$\boldsymbol{y} = \boldsymbol{\alpha x} \tag{2-8}$$

where $x = [x_{ij}]$ is the $k \times m$ transmitted array formed by stacking the information packets $x_1, x_2, \dots, x_k$, as the rows of $\boldsymbol{x}$, where the subscript of x_{ij} indicates the j'th entry of packet x_i , $i = 1, 2, \dots, k$. Also, $y = [y_{ij}]$ is a $k' \times m$ received array formed by stacking the received channel packets $y_1, y_2, \dots, y_{k'}$ as the rows of $\boldsymbol{y}$ where the subscript of y_{ij} indicates the j'th entry of packet y_i , $i = 1, 2, \dots, k'$. $\boldsymbol{\alpha}$ is a $k' \times k$ matrix corresponding to the overall transfer function of the network from the source to the receiver.

To retrieve the original information, the receiver must be able to successfully decode the received information. In order to do so, the linear system (2-7), with k' equations and k unknowns (where $k' = k$), must be solved.

In deterministic networks, such as in Fig. 2-4, each node uses fixed linear coefficients to form the linear combinations. This means that the packet does not have to include the global encoding vector, because the receiver node knows the topology of the network. The disadvantage of this type of network coding is that the network is not very robust because of its inability to change its functionality in event of an error, erasure or link failure.

In deterministic network coding it can be seen that the receiver nodes must have complete knowledge of the network topology, the management information on where network coding was implemented, as well as the coefficients used for decoding at the receiver nodes.

The topology of networks is not always known. Networks with an unknown topology can still implement network coding, but in a non-deterministic manner called random network coding.

2.3 Random network coding

In practical networks, the topology of a network is not always known. Random network coding, introduced in [2], is developed in order to exploit more of the capacity of the network by spreading the information over the available network capacity to keep the network flexible. Random network coding has the ability to achieve the same transmission rates as deterministic coding, without the need for network control and planning. It is practical and efficient without adding very much computational complexity to the intermediate nodes.

Random network coding is a localized scheme where each intermediate node only needs the information sent to it by its neighbour. The receiver carries no knowledge of the topology of the network, or how the channel packets are encoded.

Random network coding is useful in scenarios where the topology of a network is unknown, constantly changing or when the maintenance of coordination is expensive or impractical [24].

2.3.1. Encoding [2, 3]

Encoding of information by means of random network coding works as follows:
The source node, S, transmits the k information packets, $x_1, x_2, \dots, x_k$, (vectors of length m over a finite field $\mathbb{F}_q$) into the network with $(min - cut \geq k)$. This means that there exists a subset of edges, $\mathcal{H} \subset \mathcal{E}$, with size $|\mathcal{H}| = min - cut \geq k$ that supports the transmission of k packets due to the existence of k edge-disjoint paths.

A representation of the information packets transmitted and the network construction can be seen in Fig. 2-4 and Fig. 2-5.

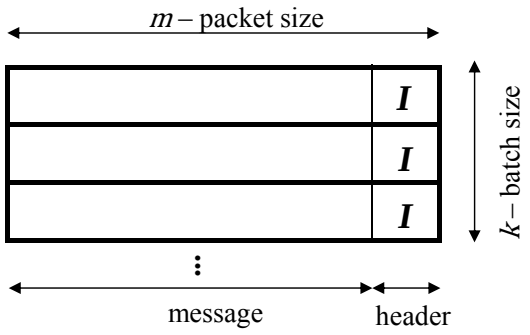


Figure 2-4: Information packets transmitted by source node

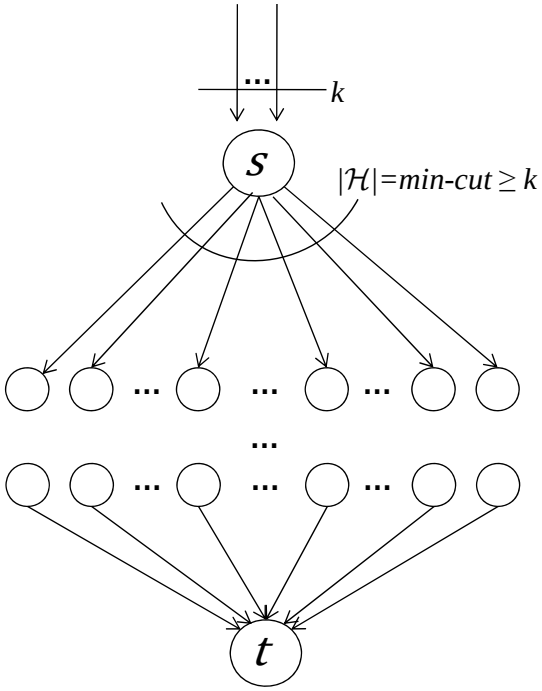


Figure 2-5: Network representation of min-cut requirements

Each network node, except the receiver node, randomly and independently selects coefficients from a finite field $\mathbb{F}_q$. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are made from all inputs to the node. Each channel packet, y_i, formed in the network is a random linear combination of $x_1, x_2, \dots, x_k, x_i \in \mathbb{F}_q$,

$$y_i = \sum_{\kappa=1}^{k} \alpha_{i\kappa} \boldsymbol{x}_{\boldsymbol{\kappa}}, i = 1,2, \dots, k \tag{2-9}$$

where α is the *global encoding vector* of $\boldsymbol{x}$. This global encoding vector makes it possible for the receiver to decode the information packets it receives. References [25] and [11] assume that this vector is sent along with $\boldsymbol{x}$ in its header. We will adopt the same assumption and send the global encoding vector inside the information packet as packet overhead.

2.3.2. Decoding [2, 3]

At the receiver end of the network, the receiver collects a large set of k' channel packets $y_1, y_2, \dots, y_{k'}$ where $k' \geq k$. This set of k' channel packets obtained by the receiver node contains a series of linearly independent combinations which is used to decode the information sent by the source node.

The source node transmitted k information packets into the network, therefore only k linearly independent channel packets are needed for decoding from the set of k' channel packets. The additional channel packets are seen as redundant information and discarded.

In a random network coding scenario, like with deterministic network coding, the receiver decodes the linear system containing k equations and k unknowns.

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_{11}\boldsymbol{x_1} + \alpha_{12}\boldsymbol{x_2} + \cdots + \alpha_{1k}\boldsymbol{x_k} \\ \boldsymbol{y_2} &= \alpha_{21}\boldsymbol{x_1} + \alpha_{22}\boldsymbol{x_2} + \cdots + \alpha_{2k}\boldsymbol{x_k} \\ &\vdots \\ \boldsymbol{y_k} &= \alpha_{k1}\boldsymbol{x_1} + \alpha_{k2}\boldsymbol{x_2} + \cdots + \alpha_{kk}\boldsymbol{x_k} \end{aligned} \tag{2-10}$$

where $\boldsymbol{x_i}$, $i = 1, 2, \dots, k$ are the k information packets sent from the source and α_i are random encoding coefficients. The linear system in (2-10) can also be written as

$$\boldsymbol{y} = \boldsymbol{\alpha x} \tag{2-11}$$

where $\boldsymbol{x} = [x_{ij}]$ is the $k \times n$ transmitted array formed by stacking the information packets $x_1, x_2, \dots, x_k$ as the rows of $\boldsymbol{x}$, where the subscript of x_{ij} indicates the j'th entry of packet x_i, $i = 1, 2, \dots, k$. Also, $\boldsymbol{y} = [y_{ij}]$ is the $k' \times n$ received array formed by stacking the received channel packets $y_1, y_2, \dots, y_{k'}$ as the rows of $\boldsymbol{y}$ where the subscript of y_{ij} indicates the j'th entry of packet y_i, $i = 1, 2, \dots, k'$. $\boldsymbol{\alpha}$ is a $k' \times k$ matrix over F_q corresponding to the overall transfer function of the network from the source to the receiver [3].

To retrieve the original information, the receiver must be able to successfully decode the received information by solving the linear system $\boldsymbol{y} = \boldsymbol{\alpha x}$. The receiver needs at least $k' \geq k$ equations for successful decoding.

Linearly dependent packets (packets with linearly dependent global encoding vectors) are useless for the decoding of the channel messages at the receiver. The receiver needs k linearly

independent equations in order to retrieve the sent information. When the receiver receives k channel packets with linearly independent global encoding vectors, it will be able to decode the k message packets [25].

Due to the non-deterministic nature of random network coding, the global encoding vectors received in the channel packets provide the information of each packet's linear combination in order for the receiver to decode the data. The lack of knowledge concerning the combinations of the channel packets makes it difficult for the receiver node to know how long it should wait to obtain sufficient channel packets for decoding.

2.4 Advantages of network coding

Network coding offers a wide range of benefits along a very diverse range of communication networks. These benefits include the following:

- throughput
- energy efficiency
- robustness

2.1.1. Throughput

The improvement of network throughput is the first advantage identified for network coding [26]. The better use of resources in the network improves the network throughput by letting the intermediate nodes in the network process the information it receives. Each node implemented with network coding receives the information from all the input nodes, encodes it, and sends it out to the receiver nodes. This process can increase the achievable rate of the network compared to straightforward routing [27, 28]

The benefit of throughput can be seen in the example of the butterfly network. The example shows that if intermediate nodes are allowed to combine information in the network and the information is then retrieved at the receivers, the throughput of the network can be increased.

2.1.2. Energy efficiency

Network coding can offer a wide range of advantages when used in a wireless environment. Conservation of energy is one of the largest benefits offered.

Consider Fig. 2-6, [29] a wireless network where devices A and C would like to exchange messages. Device A wants to send packet x_1 over B, which acts as a relay, while device C would like to send packet x_2 to A over the relay B.

We considered time as slotted and that only one packet can be sent over a channel in one time slot. This means that half duplex communication is used where only one packet can be sent or received during that time.

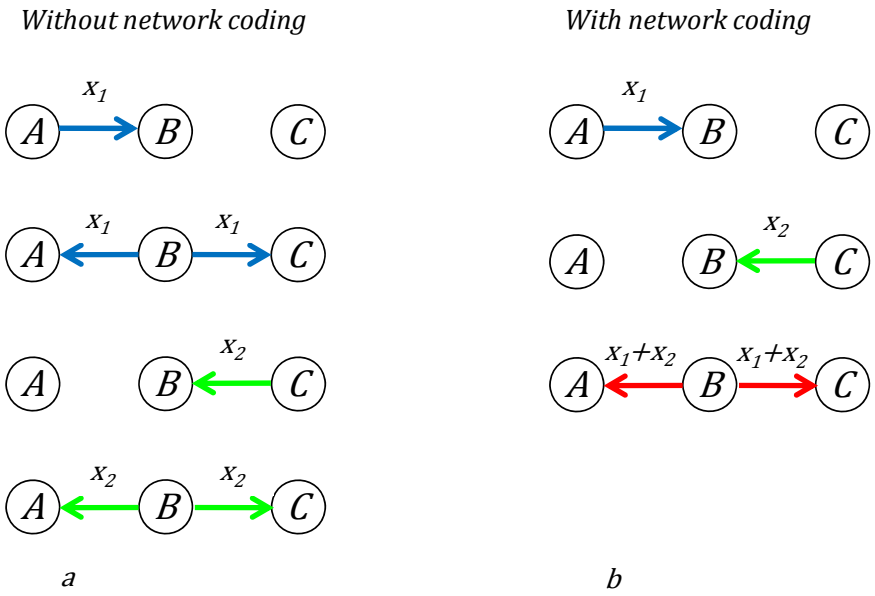


Figure 2-6: Saving of energy in wireless network using network coding

In Fig. 2-6 (a) the traditional approach can be seen. Node A and C forward their message to node B which forwards each message to both nodes A and C.

Fig. 2-6 (b) illustrates the network coding approach to broadcasting messages over wireless channels. Node B receives message x_1 from A and message x_2 from C and performs a bitwise x-or to create message $x_3 = x_1 \oplus x_2$. Node B then broadcasts this message to nodes A and C using a common transmission. Node A knows x_1 and therefore decodes x_2.

Similarly, node C knows x_2 and can decode x_1. This approach offers the advantage of energy efficiency, because node B only transmits once, instead of twice.

2.1.3. Robustness [27] [30]

Random network coding provides a higher level of network robustness. It utilizes the maximum capacity of the network; and by spreading out the information over the network, it becomes more robust. Network robustness is also improved, because each packet transmitted in the network contains a linear combination of information of several other packets. This aids in the recovery of the original data at the receiver.

2.5 Disadvantages of network coding [5]

The biggest disadvantage associated with network coding is the fact that the network can be very sensitive to errors. A single error packet has the potential to infect the whole network and corrupt all the packets used by the receiver for decoding.

When a corrupted packet is linearly combined with legitimate packets, it can corrupt all the information contained in those packets. Another problem that may occur is that an insufficient number of packets containing the correct information of a single source node may reach the receiver, therefore preventing the receiver to decode the correct messages sent from the source.

2.6 Conclusion

This chapter introduced graph theory which is used to model networks by showing the relationships of the edges and nodes. After discussing the different properties of networks, the *min-cut max-flow* theorem was presented and the impact of this theorem on networks was discussed.

After an overview of network coding and graph theory; deterministic network coding and random network coding were discussed along with the advantages and disadvantages they bring.

The impact of errors occurring in networks creates a need for some ability to counter the effects of errors in a network. A method is needed so that information transmitted over a network can be received successfully and correctly. Various error correction methods have been identified that can address these shortcomings if implemented in a network. An error correction code is able to correct and detect data packets corrupted due to additive errors occurring in a network. An error correction method can improve the robustness of the network by obtaining the correct information even when based only on partially correct information [8]. These error correction methods are discussed in Chapter 3.

Chapter 3:

Network error correction

Chapter 3 contains an overview of network error correction in network coding networks. Background on forward error correction codes, as well as the techniques and implementations of network error correction is given.

3.1. Introduction

Networks that implement network coding and random network coding are not necessarily error free and can be very sensitive to errors [4, 5]. An error is defined by Cai and Yeung in [6] as an occurrence where the output symbol of a channel differs from the corresponding input symbol.

The impact of errors present in networks creates a need for some ability to counter the effects of errors in the network. Methods are needed so that information transmitted over a network can be received successfully and correctly.

These needs were first addressed by implementing classical error correction methods in point-to-point networks. These error correction codes are based on classical coding theory and applied to networks on a link-by-link basis. Classical error correction codes can be implemented successfully in a network, because the redundancy is spread over time.

An example where classical error correction can be implemented, is a network where errors are caused by noise in the network channels. A classical error correction code can be successfully implemented on a link-by-link basis in such a network, because all the intermediate nodes in the network apply error correction to the information received before transmitting it on to its outgoing edge [31, 32].

One problem associated with this kind of error correction is that all the nodes in the network have to implement error correction which is computationally expensive. Another is that when errors are injected into the network by an adversary such as a malicious node, which is continually unreliable, this error correction code is insufficient. A classical error correction code spreads redundancy only over time and is unable to correct errors in the network constantly caused by adversaries [31] [33].

Cai and Yeung in [7] addressed these problems by introducing a more generalized approach to classical point-to-point error correction named *network error correction*. Network error correction is executed through the use of network coding in networks. The implementation of network error correction spreads the redundancy not only over time, but over space as well.

For linear network error correction, the intermediate nodes in the network only need to implement linear operations; and only the receiver node needs to implement error correction. This leads to a considerable computational advantage when compared to link-by-link error correction [31].

In [8] it is shown that in erroneous networks, network error correction can be used so that errors occurring in the network can be detected and corrected. Lower and upper bounds are also defined for the specific error-correcting capability of the code.

Fragouli et al. in [8] states that the implementation of a network error correction code can improve the robustness of the network by obtaining the correct information even when only partially correct information is received.

It is clear to see that network error correction is a generalisation of classical error correction and that it is able to improve network reliability and the network's ability to counter the effect of errors. To obtain a good understanding of network error correction, background information on classical error correction codes is first given.

3.2. Error correction codes [9, 10] [34, 35] [39]

3.2.1 Forward error correction codes

Forward error correction is achieved by adding redundancy to the information transmitted in to the network. This redundancy is added by a predetermined algorithm. These redundant symbols are a complex function of some/all of the original information symbols. The redundancy is then used by the decoder to determine if an error has occurred and if it can be corrected. The receiver does not send information back to the source node, hence the term *forward* error correction.

The two main categories of forward error correction codes are block codes and convolutional codes.

Block codes operate on blocks/packets with a fixed, predetermined size. Each block code depends only on the information contained by it and is independent of information contained in previous codewords. Therefore, encoders of block codes are memory-less.

Convolutional codes (continuous codes) work on a stream of bits or symbols of arbitrary length. Convolutional codes differ from block codes, because the encoder is not memory-less and the outputs given at a certain time depends on the *k* information symbols as well as the *m* previous input blocks of code.

In the following section of this literature survey, the focus falls only on block codes. This is due to the following considerations:

- In the subsequent chapters when networks are constructed for simulation purposes, the source node generates a random block of information symbols for each simulation iteration, independent from the information sent in the previous iteration. The links in these networks are also seen as memory-less Binary Symmetrical Channels.

- Yeung and Cai in [33] stated that block codes are the ideal type of codes to use for network coding, because network coding operates on packets of a fixed, predetermined size.

3.2.2 Block codes

A block code operates on a block of symbols, specified by (n, k). In the group of symbols, coded (redundant) bits are added to make a larger block by using a predetermined algorithm. A block code uses sequences of n symbols, where $n \geq 0$. Each code word/ block of length *n* contains *k* information symbols. The other $n - k$ symbols in the code word are redundant symbols and are called parity check symbols. These symbols do not contain additional information, but make it possible to detect and correct possible errors that may occur in the transmission of the code from source to receiver. These codes of length *n* and *k* information symbols are called a (n, k) block code.

Li et al. in [36] said that network coding is a linear process where a node encodes the information it receives through linear transformation. Therefore, linear block codes, a type of block code, will subsequently be studied.

3.2.3 Linear block codes

Codewords of a (n, k) linear block code, C, over field $\mathbb{F}_q$ has length n and contain k information symbols (dimension of code), where $k < n$. For linear block codes, the linear combination of codewords remains a codeword.

An important concept regarding valid codewords is that of the *Hamming distance*. The Hamming distance is developed for error correction and detection in digital communications.

Definition 3-1 [9, 10, 38]: Consider two code vectors, $\boldsymbol{x} = x_1, x_2, \dots, x_n$ and $\boldsymbol{y} = y_1, y_2, \dots, y_n \in \mathbb{F}_q$. The *Hamming distance*, $d(x, y)$, between these two vectors are equal to the number of symbols by which they differ.

The specification of a linear block code and the definition of $d(x, y)$, lead to a very important relationship in coding theory named *minimum distance*.

Definition 3-2 [9]: The minimum distance, d_{min}, of code C is defined as

$$d_{min} \triangleq \min \{d(x, y): x, y \in C, x \neq y\} \tag{3-1}$$

When the minimum distance of linear code C is d_{min} any two codewords of C differs in at least d_{min} places. For this specific code C, an occurrence of $d_{min} - 1$ errors cannot turn one codeword into another. This means that when fewer than d_{min} errors occur, the received codeword will not be a valid codeword in C and the receiver is able to detect the occurring errors.

Definition 3-3 [9]: A linear code, C, guarantees the detection up to $d_{min} - 1$ errors and the correction of $t = \frac{d_{min} - 1}{2}$ or fewer errors, where

$$t = \frac{d_{min} - 1}{2} \tag{3-2}$$

is called the *random-error-correcting capability* of code C.

Definition 3-4 [37]: A linear code, C, is *t-error-correcting* if it can correct all τ-errors for $\tau \leq t = \frac{d_{min}-1}{2}$, i.e., if the total number of errors in the network is at most t, then the source message can be recovered by the sink node $r \in \mathcal{V}$.

Encoding

Each block code that is used for error correcting purposes has a specific mapping between the codeword, $\boldsymbol{r}$, of length n and the message, $\boldsymbol{m}$, of length k.

All the codewords of a linear block code, with a dimension of k, can be represented as a linear combination of k linearly independent vectors, $\boldsymbol{g_1}, \boldsymbol{g_2}, \dots, \boldsymbol{g_k}$, so that

$$\boldsymbol{r} = m_1\boldsymbol{g_1} + m_2\boldsymbol{g_2} + \cdots + m_k\boldsymbol{g_k} \tag{3-3}$$

where $m_1, m_2, \dots, m_k$, $m_i \in \mathbb{F}_q$ are message symbols.

These linearly independent vectors can be represented as a matrix to form G.

$$G = \begin{pmatrix} \boldsymbol{g_1} \\ \boldsymbol{g_2} \\ \vdots \\ \boldsymbol{g_k} \end{pmatrix}. \tag{3-4}$$

To encode, the message symbols $\boldsymbol{m} = [m_1, m_2, \dots, m_k]$ need to be multiplied with G, so that

$$\boldsymbol{r} = \boldsymbol{m}G. \tag{3-5}$$

It is clear to see that the rows of matrix G are used to generate the codeword from the information message; therefore matrix G is called the *generator matrix*. The generator matrix can be used to form any one of $c = q^k$ valid codewords consisting of information and parity symbols in the form $\boldsymbol{r} = \{m_1 m_2 \dots m_k\, p_1\, p_2 \dots p_{n-k}\}$, where $p_i \in \mathbb{F}_q$, $i \in 1, \dots, n-k$.

Error detection and correction

The first step in error detection and correction is to determine if the received vector, $\boldsymbol{r}'$, is a valid codeword, $\boldsymbol{r}$. In order to do so, a parity check matrix, H, is generated. Every (n, k) linear block code has an associated $(n-k) \times n$ parity check matrix H. This parity matrix is computed from the generator matrix and then used throughout the decoding process. The linear equations used for calculating the parity symbols provide the information needed to generate H.

For a specific linear block code

$$GH^T = \mathbf{0}\,. \tag{3-6}$$

The parity check matrix has the important property of

$$\boldsymbol{r}H^T = \mathbf{0}\,. \tag{3-7}$$

where $\boldsymbol{r} \in C$.

To determine if an error occurred, the receiver must multiply H and the received codeword, $\boldsymbol{r}'$, to obtain a syndrome vector $\boldsymbol{s}$:

$$\boldsymbol{s} = \boldsymbol{r}'H^T\,. \tag{3-8}$$

This syndrome vector indicates if the received codeword, $\boldsymbol{r}'$, is a valid codeword or not. From (3-8) it can be seen that when $\boldsymbol{s} = \mathbf{0}$, the received codeword, $\boldsymbol{r}'$, is error free and $\boldsymbol{r}' = \boldsymbol{r}$. When $\boldsymbol{s} \neq \mathbf{0}$, $\boldsymbol{r}'$ is not a valid codeword and indicates that one or more errors have occurred.

Suppose codeword, $\boldsymbol{r} \in C$, is transmitted over an erroneous channel and vector $\boldsymbol{r}'$ is received. We can write

$$\boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{e}\,. \tag{3-9}$$

where $\boldsymbol{e}$ is the error vector. The elements of $\boldsymbol{e}$ is zero, except in positions e_i, $i \in \{1,2,\ldots,n\}$, where $e_i \in \mathbb{F}_q$, and indicate the indices where errors occurred.

We calculate the syndrome vector, $\boldsymbol{s}$

$$\boldsymbol{s} = \boldsymbol{r}'H^T = (\boldsymbol{r} + \boldsymbol{e})H^T = \boldsymbol{r}H^T + \boldsymbol{e}H^T = \boldsymbol{0} + \boldsymbol{e}H^T . \quad (3\text{-}10)$$

A t-error correcting block code can detect up to $n-k$ errors and correct $\frac{d_{min}-1}{2}$ errors, where $d_{min} - 1 \leq n - k$. When $\tau \leq t$ errors occur in the codeword (where the block code is t-error correcting) we are able to correct all the errors as well.

Decoding

When the codeword, $\boldsymbol{r}$, is successfully corrected, the original message, $\boldsymbol{m}$, can be obtained by simply multiplying $\boldsymbol{r}$ and G:

$$\boldsymbol{m} = \boldsymbol{r}G^T . \quad (3\text{-}11)$$

3.3. Network error correction [3] [6] [12] [37]

In this section we move on to implement a classical error correction code in a network that uses random network coding to perform network error correction. In order to do so, we must adhere to two important characteristics of network error correction discussed in Section 3.1:

- Network error correction is executed through the use of network coding in a network.
- In a network where network error correction is applied, the intermediate nodes in the network only need to implement linear operations and only the receiver node implements error correction.

These attributes are considered in [37] and we will base our choice of specific network characteristics on it. We consider a network that uses random network coding where:

- Only the source and receiver nodes apply error correction techniques and have no knowledge of the topology of the network.

- The intermediate nodes are unaware of the outer error correction code and simply perform random network coding by creating random linear combinations of their inputs and forwarding it to the next node.

3.3.1. Redundant packets

In Chapter 2, we considered the encoding and decoding operations of a random network coding network. We continue to look at these processes, but in a scenario where network error correction is needed. Yeung and Cai [6] proposed a network error correction scheme where redundant packets are added at the source and sent over the network.

Consider a network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where random network coding is applied. The source node, S, must transmit k information packets, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, (vectors of length m over finite field $\mathbb{F}_q$) over the network. Due to possible errors that may occur in the network, the node applies a network error correction code to the information packets.

The source node uses a (n, k) block code to encode these k information packets into n outgoing coded packets, denoted as $\boldsymbol{x'}_1, \boldsymbol{x'}_2, \ldots, \boldsymbol{x'}_n$ where $n > k$, basically adding packets as parity. The redundant packets are generated by using randomly generated coefficients g_{ij} from a finite field $\mathbb{F}_{2^q}$ so that

$$\boldsymbol{x'}_i = \sum_{j=1}^{k} g_{ij}\boldsymbol{x_j}\ , i = 1, 2, \ldots, n \tag{3-12}$$

The number of redundant packets generated for transmission is equal to at least $(n - k)$ packets. The set of coefficients $g_{i1}, g_{i2}, \ldots, g_{ik}$ can be referred to as the *encoding vector* for $\boldsymbol{x}_i$ [8] and are sent in the coded packet as the overhead. This overhead, however, is negligible because the information packets are sufficiently large.

These n coded packets are then transmitted by the source node, S , into the network with a $min - cut \geq n$. This means that there exists a subset of edges, $\mathcal{H} \subset \mathcal{E}$, in the network with size $|\mathcal{H}| = min - cut \geq n$. The network must have a *min-cut* equal to at least n in order to support the transmission of the n coded packets through n edge-disjoint paths. A

representation of the coded packets transmitted and the network structure can be seen in Fig. 3-1 and Fig. 3-2.

In the network, all the intermediate nodes randomly and independently select coefficients from a finite field $\mathbb{F}_q$. They create random linear combinations that are then forwarded to the next node.

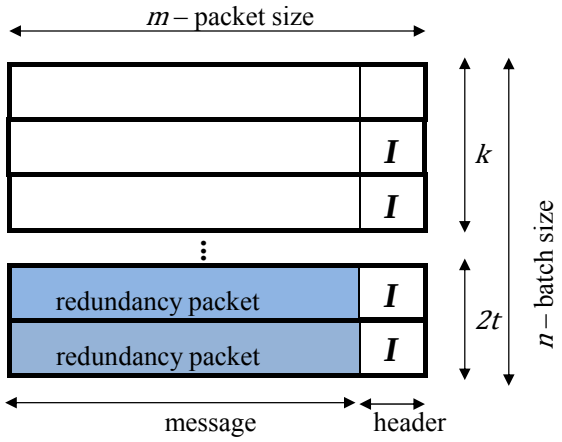


Figure 3-1: Information packets with parity packet s transmitted by source node

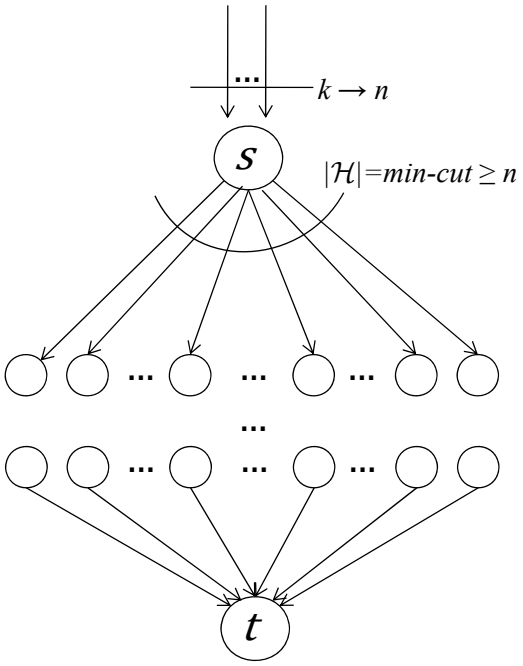


Figure 3-2: Network representation of *min-cut* requirements

When an error occurs on edge (a, b) in network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the symbol sent by node $a \in \mathcal{V}$ differs from the symbol received by node $b \in \mathcal{V}$. Jaggi et al. in [12] describe that errors occurring in the network can be seen as packets inserted by a second source and that the information received at the receiver are linear combinations of the source information as well as that of the error information. They implement network error correction to extract the source information from the received mixture of channel information and errors.

For the n coded packets transmitted into the network, let e_η denote error packets applied by the additional source to the information packet $x_\eta \in \mathbb{F}_q, \eta \in \{1, \dots, n\}$. Each channel

packet, y_i, formed in the network contains random linear combinations of the coded packets as well as the error packets:

$$y_i = \sum_{\eta=1}^{n} \alpha_{i\eta} \boldsymbol{x'}_\eta + \sum_{\eta=1}^{n} \beta_{i\eta} \boldsymbol{e}_\eta \, , i = 1,2, \dots, n \tag{3-13}$$

where α is the *global encoding vector* of $\boldsymbol{x'}$, $\boldsymbol{e} = [\boldsymbol{e}_1^T, \boldsymbol{e}_2^T, \dots, \boldsymbol{e}_n^T]$ is an array consisting of all the erroneous packets introduced in the network and β is the overall transfer matrix of these packets from the source to destination. If $e_\eta = 0$, no errors were applied to information packet $\boldsymbol{x'}_\eta \in \mathbb{F}_q, \eta \in \{1, \dots, n\}$.

The receiver collects a set of n' channel packets $y_1, y_2, \dots, y_{n'}$ where $n' \geq n$. This set of n' channel packets obtained by the receiver node now contains linear combinations of:

- the coded packets sent by the source node
- the error packets applied by an additional source.

The receiver selects a set n channel packets and applies error correction and detection techniques to the received linear system (3-15) containing n equations and n unknowns.

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_{11}\boldsymbol{x'_1} + \alpha_{12}\boldsymbol{x'_2} + \cdots + \alpha_{1k}\boldsymbol{x'_k} \\ \boldsymbol{y_2} &= \alpha_{21}\boldsymbol{x'_1} + \alpha_{22}\boldsymbol{x'_2} + \cdots + \alpha_{2k}\boldsymbol{x'_k} \\ &\vdots \\ \boldsymbol{y_n} &= \alpha_{n1}\boldsymbol{x'_1} + \alpha_{n2}\boldsymbol{x'_2} + \cdots + \alpha_{nk}\boldsymbol{x'_k} \end{aligned} \tag{3-14}$$

or

$$\boldsymbol{y} = \boldsymbol{\alpha x'} + \boldsymbol{\beta e}, \tag{3-15}$$

where $x = [x_{ij}]$ is the $k \times m$ transmitted array formed by stacking the information packets $x_1, x_2, \dots, x_k$, as the rows of $\boldsymbol{x}$, where the subscript of x_{ij} indicates the j'th entry of packet $x_i \, , i = 1, 2, \dots, k$. Also, $y = [y_{ij}]$ is a $k' \times m$ received array formed by stacking the received channel packets $y_1, y_2, \dots, y_{k'}$ as the rows of $\boldsymbol{y}$ where the subscript of y_{ij} indicates the j'th entry of packet $y_i \, , i = 1, 2, \dots, k'$. $\boldsymbol{\alpha}$ is a $k' \times k$ matrix corresponding to the overall transfer function of the network from the source to the receiver.

The receiver can then decode the original message, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, by decoding the (n, k) linear block code.

The symbol rate (defined in Section 2.1.6) of this network is:

$$R = \frac{transmitted\ symbols}{received\ symbols} = \frac{n}{n} = 1. \tag{3-16}$$

where n coded symbols are sent into the network and n channel symbols are received.

The process of network error correction in a random network coding network is summarised by Fig. 3-3.

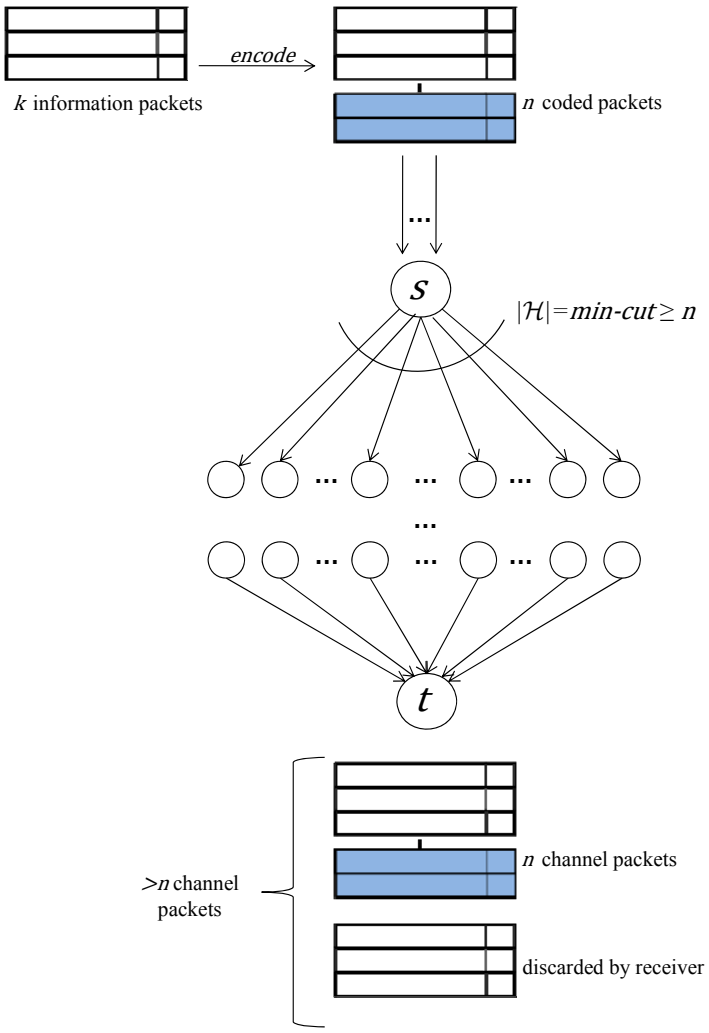


Figure 3-3: Representation of network error correction method

Another implementation of network error correction is not the addition of parity packets, but the addition of parity symbols to information packets.

Redundant symbols [12]

Jaggi et al. in [12] implement network error correction by adding redundancy to the source information that satisfies certain constraints. This information packet is constructed as in Fig. 3-4:

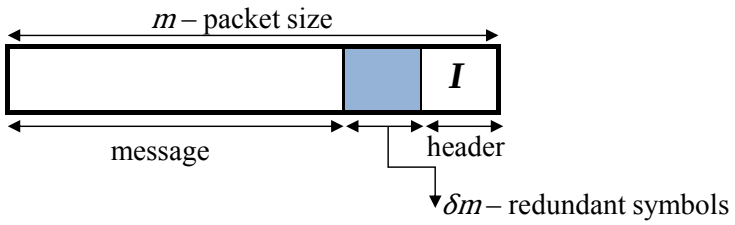


Figure 3-4: Information packet with redundancy symbols

Each packet contains a sequence of m symbols from the finite field $\mathbb{F}_q$. Out of the m symbols in the information packet; δm symbols are redundancy added by the source. The δm redundant symbols are chosen as parity symbols in order for the receiver to decode the channel packet. Also included in the packet is the global encoding vector, reflecting the linear combinations formed on the channel packet.

3.4. Conclusion

In this chapter we discussed the basics of forward error correction codes, as well as linear block codes and their encoding and decoding processes. The choice of using linear block codes as the basis for network error correction was given.

Network error correction can be executed by adding redundancy symbols at the source of the network and transmitting it over the network either inside the existing packets or in additional packets. These network error correction techniques improve network robustness by

allowing the receiver node to obtain the correct information even in the presence of channel errors.

The disadvantage associated with these error correcting schemes is that more than just information packets are injected into the network. The redundancy needed for error correction is generated at the source and must be transmitted in to the network as well. The encoding process at the source and the injection of redundant packets leads to greater energy consumption, as well as additional bandwidth use that cause more packet losses in a crowded network.

In Chapter 4 we propose our method that does not add redundancy to the network. It uses the redundant information implicitly generated inside the network to correct errors.

Chapter 4:

Implicit error correction

Chapter 4 presents the conceptual implicit network error correction method developed in this dissertation. This method uses the redundant information obtained by the receiver node to apply network error correction. This method is designed, a set of network parameters are selected and the obtained results are evaluated in order to learn if the proposed method can be implemented successfully and if it is an accurate representation of the conceptual model.

4.1 Introduction

In this chapter a conceptual method is designed that implements network error correction in a random network coding network without adding redundancy at the source. The design of this method is based on the concepts of random network coding, error correction and network error correction seen in the literature.

In the reviewed literature on network error correction, discussed in Chapter 3, it is evident that the construction of the existing network error correction schemes for random network coding networks is in a concatenated form. This means that redundant information is added at the source node and transmitted over the network in order for the receiver to apply error correction.

From the study of random network coding in Chapter 2, we can see that these environments have a tendency to generate redundancy inside the network which is normally discarded by the receivers.

Redundancy is the foundation of network error correction and we aim to eliminate the need for the source node to encode data by exploiting this redundancy formed by the network to apply error correction. We found that currently no previous work was performed relating to the exploitation of the implicit encoding abilities of a network that implements random network coding.

To achieve this, we propose a method that operates as follows:

1. A block of k linearly independent information packets is generated at the source $S \in \mathcal{V}$ of network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
2. This block of information is transmitted into the random network coding network ($min - cut \geq k$) which contains a subset of edges, $\mathcal{H} \subset \mathcal{E}$ with size $|\mathcal{H}| = min - cut \geq k$.
3. Intermediate nodes perform linear combinations on their inputs and channel packets are formed that propagate through the random network coding network.

4. The receiver node is connected to another subset of edges, $\mathcal{H}' \subset \mathcal{E}$ with size $|\mathcal{H}'| \geq n > k$. The receiver selects a set of n channel packets that consist of two sets of linearly independent equations of sizes k and $(n-k)$ respectively.
5. The receiver decodes the transmitted information and the possible errors are detected and corrected.

By waiting for more channel packets and utilizing the redundant channel packets received, the receiver obtains additional information for decoding that may be used for network error correction. The coded channel packets obtained by the receiver provide the redundancy required for error correction.

This method differs from the existing concatenated network error correction methods because:

- The network acts as a network error correction encoder: The network encodes the transmitted information implicitly by adding redundancy needed for error correction so that no encoding is needed at the source node.

- The receiver implements independent decoding: The receiver node does not have to communicate with the source node. It can apply any error correction scheme to the received data without informing the source. Its choice of error correction algorithm can be based solely on the channel packets it collects.

This method will be evaluated to determine if it renders an improvement in information rate and error correction ability compared to the existing concatenated network error correction methods. The improvement of the method is done in three iterations. The first iteration, discussed in this chapter, is to determine if the method can be implemented and if it works properly. Chapter 5 and Chapter 6 describe further improvement iterations.

The natural randomness of random network coding makes this analysis non-deterministic. For this reason a large number of iterations had to be performed in order to obtain an accurate result. We decided to run between 1 000 and 1 000 000 simulation iterations for each parameter set, which rendered relatively constant results in respect to the mean value of each set.

4.2 Implicit error correction method

4.2.1 Implicit encoding

This method utilizes a network that uses random network coding where the *min-cut* between the source and the receiver nodes must be large enough to support the transmission of k linearly independent information packets ($min - cut \geq k$) , although n channel packets will be used by the receiver node.

As stated in Chapter 2, the receiver node would normally select k linearly independent packets from the collected channel packets in order to decode the source message. But in a network where $min - cut \geq k$, more than k linear equations are considered as redundant information. Our method proposes to use this redundant information received by the receiver node to apply error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because the network will transmit sufficient redundancy to the receiver for error correction.

In this method, k independent information packets (vectors of length m bits over finite field $\mathbb{F}_q$) are sent from the source node, as seen in Fig. 4-1, where the packets propagate through the network.

According to [25], the overhead (header) sent with the information packet has a size of

$$m - p = k \log_2 q \tag{4-1}$$

symbols, which is negligibly small if the packet size, m, is sufficiently large.

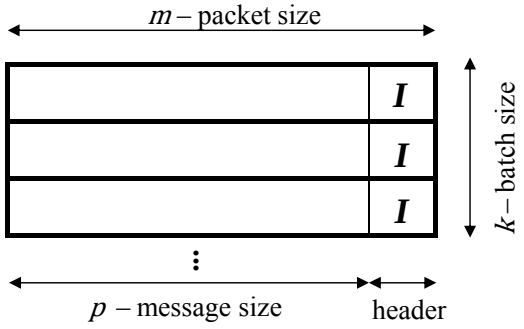


Figure 4-1: Information packets without redundancy transmitted by source node

We assume that the global encoding vector is sent along with information packet in its header, but consider it negligible because the packets we use are sufficiently large. Therefore we assume that, $m \approx p$.

Let $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ (vectors with the length of m symbols over finite field $\mathbb{F}_q$) be the k information packets that must be transmitted over the network. The source node *does not* apply a (n, k) a linear block code to the k information packets. The k information packets are transmitted by the source node, S, into the network with a $min - cut \geq k$. This means that there exists a subset of edges, $\mathcal{H} \subset \mathcal{E}$, in the network with size $|\mathcal{H}| = min - cut \geq k$. Since *min-cut* of the network is equal to at least k, it supports the transmission of the k independent information packets.

The k information packets propagate through the network; linear combinations are formed from them by intermediate nodes and the receiver waits until it receives n or more channel packets. For the receiver to obtain at least n channel packets, the receiver must be connected to another subset of edges, $\mathcal{H}' \subset \mathcal{E}$, in the network with size $|\mathcal{H}'| \geq n > k$. The additional $(n - k)$ edge-disjoint paths connected to the receiver make the reception of valid parity packets possible. The representation of this type of network can be seen in Fig. 4-2.

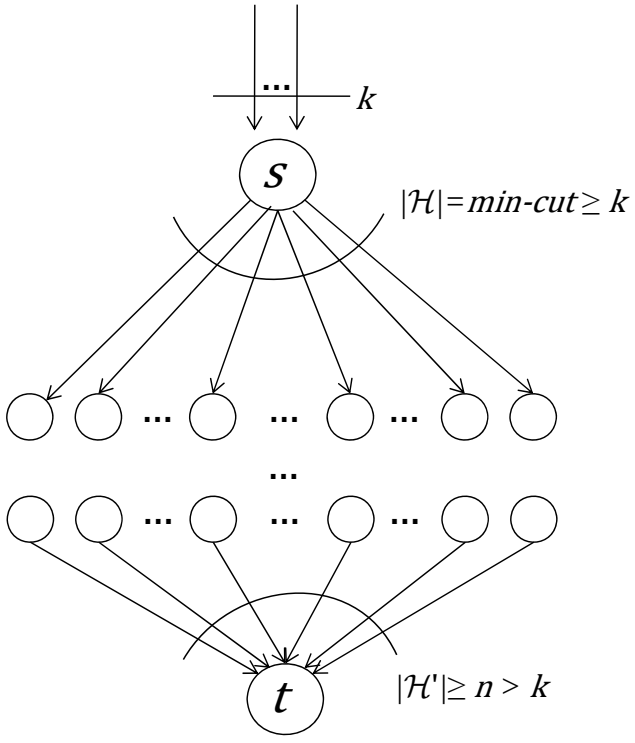


Figure 4-2: Network representation of *min-cut* requirements

4.2.2 Error correction and decoding [40 – 42]

The receiver waits until it can collect at least n channel packets, $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n$, where

$$\boldsymbol{y}_i = \sum_{\eta=1}^{n} \alpha_{i\eta} \boldsymbol{x}_\eta \ , i = 1,2, \dots, n \ . \tag{4-2}$$

The values of n and k are determined by the specific (n, k) linear block code selected. These n channel packets must consist of two sets of linearly independent packets, of size k and $(n - k)$, respectively. The extra $(n - k)$ received packets are what we intend to use in order to correct any possible errors. They will be used as parity packets.

In order to find the needed two sets of linearly independent channel packets, sets of simultaneous equations have to be solved. The suggested numerical method to solve simultaneous equations is to create an upper triangle matrix from the coefficient matrix in our case the overhead matrix, T, of the channel packets.

The `[Q,R,E] = QR(T)` MATLAB function performs orthogonal-triangular decomposition on the received coefficient matrix and enables us to determine the sets of linearly independent columns of the overhead matrix, T. This function produces a unitary matrix, Q, upper triangular matrix, R and permutation matrix, E, where $T \times E = Q \times R$. The columns of the permutation matrix allows us to determine which columns of T is linearly independent. When the independent columns of matrix T are determined, it is straightforward to produce two linearly independent channel packet sets to form the codeword $\boldsymbol{r'}$.

The first set of linearly independent equations that must be decoded is that of the traditional k packets:

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_{11}\boldsymbol{x_1} + \alpha_{12}\boldsymbol{x_2} + \cdots + \alpha_{1k}\boldsymbol{x_k} \\ \boldsymbol{y_2} &= \alpha_{21}\boldsymbol{x_1} + \alpha_{22}\boldsymbol{x_2} + \cdots + \alpha_{2k}\boldsymbol{x_k} \\ &\vdots \\ \boldsymbol{y_k} &= \alpha_{k1}\boldsymbol{x_1} + \alpha_{k2}\boldsymbol{x_2} + \cdots + \alpha_{kk}\boldsymbol{x_k} \end{aligned} \tag{4-3}$$

or

$$\boldsymbol{y_{data}} = \sum_{\kappa=1}^{k} \alpha_{i\kappa} \boldsymbol{x}_\kappa \tag{4-4}$$

where $\boldsymbol{y_{data}} = [y_{ij}]$ is a $k \times m$ matrix formed by stacking the received message packets $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$ as the rows of y_{data}, where the subscript of y_{ij} indicates the *j*'th entry of packet $\boldsymbol{y_i}$, $i = 1, 2, \ldots, k$.

Once decoded, the matrix representation of these packets is:

$$\begin{array}{c|cccc} \boldsymbol{x_1} & 1 & 0 & \ldots & 0 \\ \boldsymbol{x_2} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \boldsymbol{x_k} & 0 & 0 & 0 & 1 \end{array} \tag{4-5}$$

The other set of linearly independent packets must be of size $(n-k)$ and will be used for error correction:

$$\begin{aligned} \boldsymbol{y_{1+k}} &= \alpha_{(1+k)1}\boldsymbol{x'}_{\boldsymbol{1+k}} + \alpha_{(1+k)2}\boldsymbol{x'}_{\boldsymbol{2+k}} + \cdots + \alpha_{(1+k)n}\boldsymbol{x'}_{\boldsymbol{n}} \\ \boldsymbol{y_{2+k}} &= \alpha_{(2+k)1}\boldsymbol{x'}_{\boldsymbol{1+k}} + \alpha_{(2+k)2}\boldsymbol{x'}_{\boldsymbol{2+k}} + \cdots + \alpha_{(2+k)n}\boldsymbol{x'}_{\boldsymbol{n}} \\ &\vdots \\ \boldsymbol{y_n} &= \alpha_{n1}\boldsymbol{x'}_{\boldsymbol{1+k}} + \alpha_{n2}\boldsymbol{x'}_{\boldsymbol{2+k}} + \cdots + \alpha_{nn}\boldsymbol{x'}_{\boldsymbol{n}} \end{aligned} \tag{4-6}$$

or

$$\boldsymbol{y_{parity}} = \sum_{\eta=1+k}^{n} \alpha_{i\eta}\boldsymbol{x}_{\eta} \tag{4-7}$$

where $\boldsymbol{y_{parity}} = [y_{ij}]$ is a $(n-k) \times m$ matrix formed by stacking the received message packets $\boldsymbol{y}_{1+k}, \boldsymbol{y}_{2+k}, \ldots, \boldsymbol{y}_n$ as the rows of y_{parity} where the subscript of y_{ij} indicates the *j*'th entry of packet $\boldsymbol{y_i}$, $i = 1+k, 2+k, \ldots, n$; and packets $\boldsymbol{x'_i}$, $i = 1+k, 2+k, \ldots, n$ are linearly dependant on packets $\boldsymbol{x_i}$, $i = 1, 2, \ldots, k$.

These redundant $(n-k)$ symbols, formed during the linear random network coding action in the network, are linear functions of the original k message packets and will act as the parity symbols for error detection and correction.

These parity packets, along with the decoded data packets are used to form the Generator matrix, G, where

$$G = \begin{bmatrix} 1 & 0 & \dots & 0 & \alpha_{(1+k),1} & \cdots & \alpha_{(1+k),n} \\ 0 & 1 & \dots & 0 & \alpha_{(2+k),1} & \cdots & \alpha_{(2+k),n} \\ \vdots & \vdots & \ddots & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix} \tag{4-8}$$

where $\alpha_{i,j} \in \mathbb{F}_2$.

The receiver also forms the codeword vector $\boldsymbol{r'} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k, \boldsymbol{y}_{1+k}, \boldsymbol{y}_{2+k}, \dots, \boldsymbol{y}_n\}$ in $\mathbb{F}_q$. The receiver uses G and $\boldsymbol{r'}$ to apply conventional linear error correction and detection methods, as discussed in the literature, in order to determine if an error has occurred.

The receiver can decode the message correctly only when a second set of $(n-k)$ linear independent channel packets are received. This reception, however, cannot be guaranteed and is a function of the random network coding network.

It can be seen that exactly the same error correction and decoding process is followed as in the concatenated network error correction method. The difference is that less information is injected into the network. The *min-cut* of the network is smaller than for the concatenated network error correction method and the symbol rate is:

$$R = \frac{transmitted\ symbols}{received\ symbols} = \frac{k}{n} < 1\,, k\ <\ n \tag{4-9}$$

A representation of this implicit error correction method can be seen in Fig. 4-3.

This method offers benefits in terms of energy efficiency, because less information is injected into the network: the source node transmits k packets instead of n, into a network with a $min-cut\ \geq\ k$, instead of $min-cut\ \geq\ n$. Thus fewer packets are subject to the linear random network coding combinations. In effect we eliminate the redundant action of constructing parity using linear combinations of the information symbols at the source and then subjecting these same parity symbols to linear combinations again in the random network coding network.

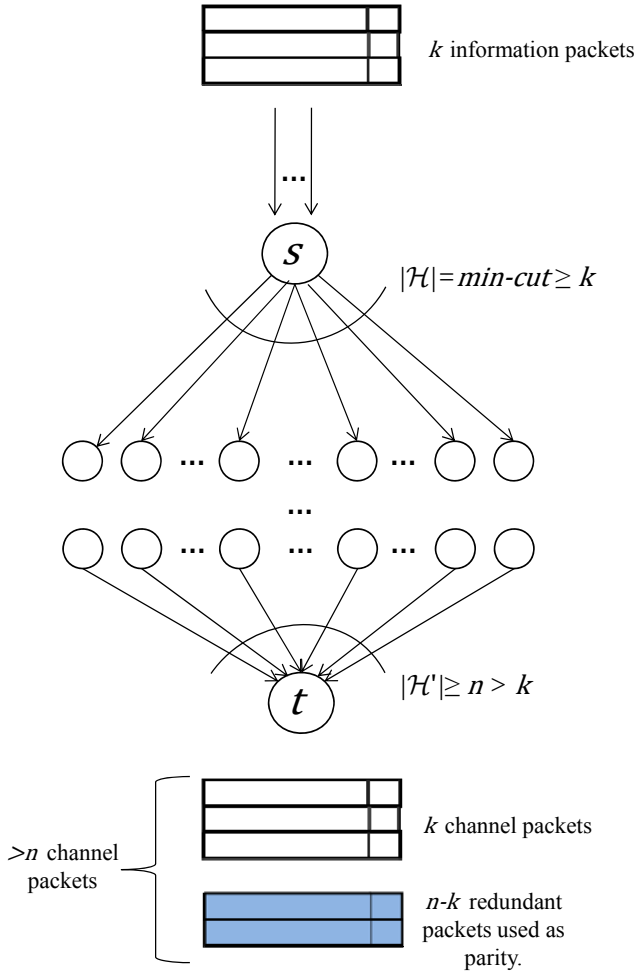


Figure 4-3: Representation of proposed method

4.2.3 Example 2-1(chapter 2) revisited [23]

Consider Fig. 4-4 where the network has a $min - cut = 2$ and unit capacity edges. This means that the network only has 2 edge-disjoint paths and can only support the transmission of 2 independent symbols. By implementing our proposed network error correction method, the source node only needs to send symbols x_1 and x_2 into the network. The third symbol, x_3, that acts as the parity symbol for error correction is generated in the network through network coding at the intermediate node.

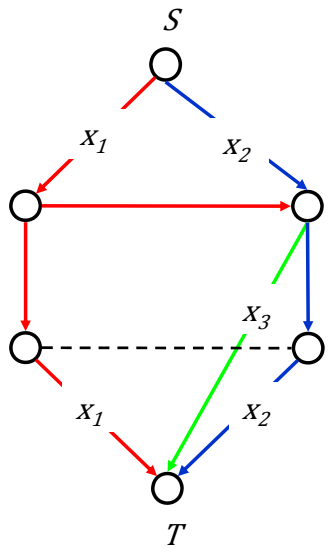


Figure 4-4: Network with $min - cut = 2$

The symbol rate of this network is:

$$R = \frac{transmitted\ symbols}{received\ symbols} = \frac{2}{3} < 1 \tag{4-10}$$

Fig. 4-4 can be compared to the network in Fig. 2-2 (Section 2.1.5). Assume that symbol x_3 transmitted over the network is the parity symbol where $x_3 = x_1 \oplus x_2$. The *min-cut* required for simultaneous transmission over the network in Fig. 2-2 then is $min - cut = 3$ with a symbol rate of 1.

The energy efficiency benefits of our proposed method can be seen, where less information is injected into the network, but the same results are obtained. The source node transmits 2 packets instead of 3, where the network size stays the same.

4.3 Model construction

The conceptual model designed in the previous section, shown in Fig. 4-3, as well as its network requirements is considered in the construction of this method for simulation

implementation. The network construction, choice of network parameters and setup of the simulation are discussed subsequently.

4.3.1. Network construction

We are able to evaluate the functionality of the implicit error correction method by simulating it in MATLAB. The random network coding network generated in MATLAB has the following structure and attributes:

The network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a non-cyclic, generic, random network where $\mathcal{V}$ is the set of nodes in the network and $\mathcal{E}$ the set of edges. This network contains a single source node $S \in \mathcal{V}$, and a single sink node, $t \in \mathcal{V}$. The source node transmits the k data packets into the network, which consists of $r \in \mathcal{V}$ intermediate nodes. The network also contains a subset of edges, $\mathcal{H} \subset \mathcal{E}$ with size $|\mathcal{H}| = min - cut \geq k$ which supports the transmission of the k independent information packets.

The receiver is connected to another subset of edges, $\mathcal{H}' \subset \mathcal{E}$, in the network with size $|\mathcal{H}'| \geq n > k$. The additional $(n - k)$ edge-disjoint paths connected to the receiver make it possible for the receiver to collect at least n channel packets. The receiver only receives a single encoded packet from each of the $|\mathcal{H}'| \geq n > k$ edges.

Each randomly generated network has the following properties:

1. All vectors and matrices have components from finite field, $\mathbb{F}_q$.
2. Number of intermediate nodes/ network size = r.
3. $Min - cut \geq k$.
4. Contains subset of edges, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| = min - cut \geq k$.
5. Contains subset of edges $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n$, connected to the receiver.
6. Average degree of node connectivity, $d_{ave=}\sqrt{r}$.
7. Link error probability, P_{le}.

Our network generator produces random networks, satisfying the above requirements, by using the following steps:

1. The user specifies the number of information packets, k, to be sent over the network.

2. The user specifies the number of intermediate nodes, r, in the network.
3. The user specifies the link error probability, P_{le}, of the network.
4. The generator generates a $z \times z$ zero matrix, where $z = (k + r + 1)$.
5. The $z \times z$ zero matrix is divided into two layers, consisting of k and $r + 1$ rows, respectively.
6. The matrix rows are filled with either a 1 or a 0, randomly generated by using the computer's clock, under the above specified conditions about the connections of the nodes in the layers.
7. Random 1's and 0's are only placed in the top triangle of the matrix. This is due to the fact that the network is non-cyclic.

Each node in the network randomly and independently generates a linear combination of its inputs. This random process works in the following steps:

1. Each (a, b) entry in the $z \times z$ network matrix is evaluated.
    a. If entry $(a, b) = 1$, the link between the two nodes exists and the process continues.
    b. If entry $(a, b) = 0$, there exists no link and the process is stopped.

2. Each existing link in the network is seen as a binary symmetrical channel (BSC), therefore noise on each link is generated specified by the link error probability, P_{le}.

3. The information obtained by each node is received from the existing noisy links.

4. For each outgoing link, the node randomly generates an output:
    a. The node randomly and independently generates coefficients from the finite field, $\mathbb{F}_q$.
    b. A channel packet is formed based on the chosen coefficients and the global encoding vector is updated.

The generation of these random network coding networks will provide a platform from where we will be able to determine whether our proposed method is functional and where improvements can be made in terms of different network parameters.

4.3.2. Selection of network parameters

In this Section, we aim to test the functionality of the method. For this reason the simplest set of parameters is chosen based on error correcting theory and mathematical calculation.

The parameters chosen for these simulations are those that will make a significant contribution toward obtaining the desired results. There exist more network parameters that are not discussed that may have a significant influence on the outcome of the simulations. These parameters do not fall in the scope of this dissertation and will remain constant throughout the simulations.

The following network parameters are either disregarded, or specifically constrained for the purpose of the study:

1. *Synchronisation*: We do not consider the effects of synchronisation in this simulation, therefore constraining the network to a non-cyclic, synchronous network. The reason for it is that we are more concerned with the functionality of the implicit encoding ability of the network, than the synchronisation thereof.

2. *Delay*: We assume that there is no processing delay at the intermediate nodes. We also assume that a single unit time delay is allowed for each packet transmission. This time delay approach is used in [36].

3. *Network infrastructure*: We assume that in a time span of a single iteration, no nodes enter or exit the network. This is justified since the network is designed and tested for its encoding ability and not its adaptability to changing infrastructure.

The following network parameters are considered important and will be evaluated and improved throughout the improvement process of this method.

Forward error correction code

In the light of the fact that for this simulation the functionality of the method is tested, and not its performance, the straightforward single error correcting Hamming code is chosen for

decoding. The Hamming (7, 4) code is a binary linear block code with a Hamming distance of $d_{min} = 3$ and an error correcting capability of $t = \frac{(d_{min} - 1)}{2} = 1$. The length of the code is $n = 7$ and the dimension $k = 4$ [9] [35].

Information packets

The selection of the binary linear Hamming (7, 4) code for decoding leads to the fact that the information packets sent over the network will only contain data from a finite field $\mathbb{F}_2$. This means that each information packet will only contain one bit of information.

Network min-cut

The theory of our proposed method described in Section 4.2, states that the random network coding network must be large enough to support the transmission of k linearly independent information packets. Therefore the network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, must contain a subset, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| \geq k$ to have a $min - cut \geq k$. For the implementation of this method with a (7, 4) Hamming code, the network must have a $min - cut \geq k = 4$, which renders at least four edge-disjoint paths between the source and receiver in the network. Also, for the implicit error correction method to be successfully implemented, network, $\mathcal{G}$, must also contain subset $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n$ where $n = 7$. This will ensure that the receiver node will receive sufficient channel packets for decoding.

Network connectivity

In Section 2.1.2 we determined that a random network is optimally robust when $\kappa = \lambda = d_{min}$ and when connections are made at random with an average degree of $d_{ave} = \sqrt{r}$, where r is the size of the network.

Each edge in the network will be randomly generated with a connectivity probability $p' = 0.5$, which ensures that all the edges are chosen with an equal probability.

Network diameter and average distance

In Section 2.1.3 we learned that the diameter of a random network of size r and $d \geq 3$ is approximately

$$D \geq 2\frac{\log(r-1)}{\log d}. \tag{4-11}$$

and the average distance

$$\frac{rD}{2(r-1)} \leq D_{ave} \leq D. \tag{4-12}$$

Connecting the network as described in the previous paragraph will ensure that the described lower bound for average distance is satisfied. This bound places a restriction on the smallest amount of random network coding opportunities for each information packet sent into the network. This bound is of value for our implementation because:

- the average distance is large enough to provide enough random network coding opportunities for each information packet, and
- the average distance is small enough to limit error propagation.

4.4 Probability of decoding

Due to the non-deterministic encoding character of the network, the possibility exists that the channel packets obtained by the receiver may not contain valid parity symbols. This will prevent the receiver from decoding the information successfully. In the first set of simulations, we investigate the probability of receiving a set of valid parity symbols from the network.

4.4.1 Simulation setup

A network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, implementing random network coding is generated for network sizes ranging from seven intermediate nodes, as described in Section 4.3. Although only four intermediate nodes are needed to obtain subset $\mathcal{H} \subset \mathcal{E}$ that provides a $min - cut \geq k = 4$ and four edge-disjoint paths; at least seven intermediate nodes, $\mathcal{H}' \subset \mathcal{E}$, are needed to create the additional three channel packets needed for error correction.

The *network size parameter* will be increased throughout the simulation until a valid set of parity symbols can be guaranteed. For each network generation, 1 000 simulation iterations for each parameter set are run. Simulations are repeated until a network size is determined that renders a guaranteed set of parity symbols.

The simulation runs in the following steps:

1. A predetermined block of input data, $\boldsymbol{x} = x_1, x_2, \dots, x_4$ (bits in finite field $\mathbb{F}_2$), is generated at the source.

2. The source node, $S \in \mathcal{V}$, transmits the four data packets in to the network.

3. The information symbols are implicitly encoded within the network.

4. The receiver collects channel packets and evaluates each global encoding vector.

5. The receiver tests if it is able to collect four channel packets, $y_{data} = y_1, y_2, \dots, y_4$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{k=1}^{4} \boldsymbol{\alpha_{ik}} x_k \ , i = 1,2, \dots ,4 \ . \tag{4-13}$$

   This test is done by performing orthogonal-triangular decomposition on the global encoding matrix of the received channel packets.

6. If possible: the receiver tests if it is able to collect an additional three channel packets, $y_{parity} = y_5, y_6, y_7$, with linearly independent global encoding vectors that will act as the parity packets:

$$y_{parity} = \sum_{k=5}^{7} \boldsymbol{\alpha_{ik}} x'_k \ , i = 5,6,7 \ . \tag{4-14}$$

   This test is done by performing orthogonal-triangular decomposition on the global encoding matrix of the remaining channel packets.

7. If possible, the network is marked as a network where the received combinations can be used to decode the original information as well as form valid sets of parity symbols for error correction.

4.4.2 Simulation results

The average results are represented in Fig. 4-5 in respect to the mean value of each set.

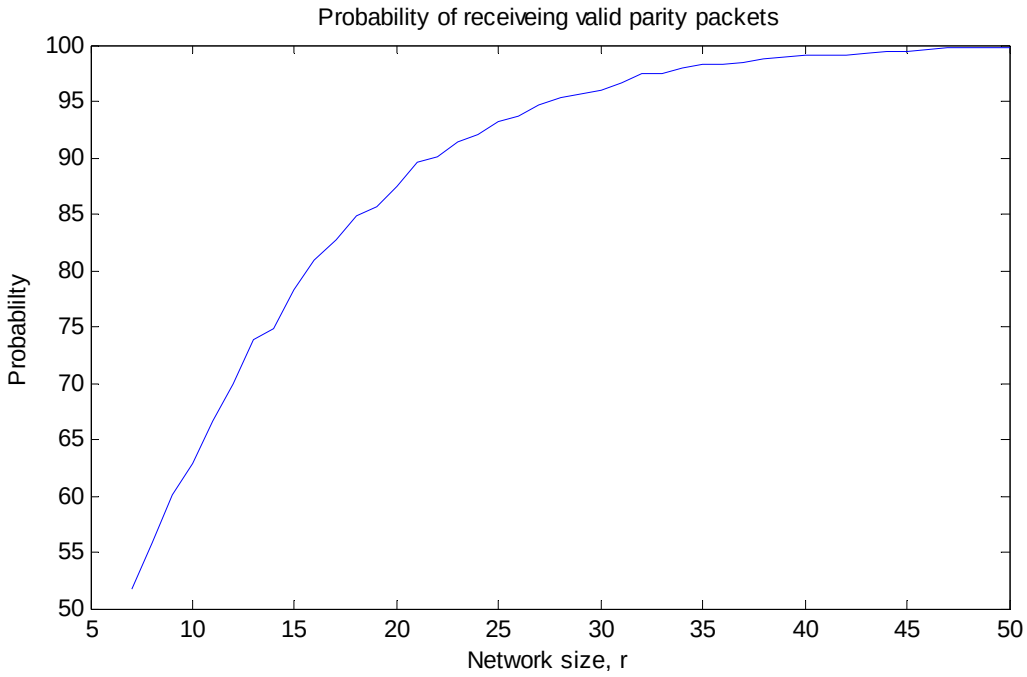


Figure 4-5: Probability of receiving valid parity packets

From the above graph, the probability of receiving valid parity packets can be deduced for the network size *r* and represented in Table 4-1.

Table 4-1: Probability of receiving valid parity packets

Network size *r*	**Probability of valid parity packets**
> 14	$> 75\ \%$
14 – 16	$70\ \% - 90\ \%$
17 – 39	$90\ \% - 98\ \%$
< 40	$> 99\ \%$

From the above simulation of a (7,4) Hamming code, one can expect to receive channel packets that can be used as parity packets for error correction less than 75 % of the time when the network has less than 14 intermediate nodes. When the network becomes bigger, the probability of receiving valid parity packets increases until about 99% for networks of sizes 40 and larger.

Based on the result of this simulation, we choose a 50-node network for the following set of simulations. Under the predetermined network parameters and network size discussed, a set of parity symbols will be guaranteed, for a 50 node network renders valid parity packets 99,83% of the time.

4.5 Error correction ability

In the following set of simulations, we aim to determine the BER performance of this method in a network where a valid set of parity packets is guaranteed. The assured probability of decoding will enable us to test the error correction performance of this method when it is successfully implemented in a network.

4.5.1 Simulation setup

In order to test the implicit error correction ability of a network, a 50-node generic network is constructed as described in Section 4.4. This network has a $min - cut \geq k$, where each node in the network randomly and independently generates a linear combination of its inputs. Errors are introduced to the network where each link acts as a binary symmetrical channel (BSC) with a specified link error probability, P_{le}.

In the simulation a random network is generated, with a specified link error probability, P_{le}, ranging from 0.5 to 10^{-12}. For each parameter pair, 100 000 simulations are run.

The simulation runs in the following steps:

1. A block of input data, $\boldsymbol{x} = x_1, x_2, \dots, x_4$ (bits in finite field $\mathbb{F}_2$), is generated at the source.

2. The source node, $S \in \mathcal{V}$, transmits the four data packets in to the network.

3. The information bits are encoded within the network.

4. The receiver collects four channel packets, $y_{data} = y_1, y_2, \ldots, y_4$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{k=1}^{4} \boldsymbol{\alpha_{ik}} x_k \tag{4-15}$$

5. The receiver decodes the source packets and forms the following matrix:

$$\begin{matrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_k} \end{matrix} \left| \begin{matrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \right. \tag{4-16}$$

6. The BER is calculated.

7. The receiver collects an additional three channel packets, $y_{parity} = y_5, y_6, y_7,$ with global encoding vectors that that will act as the parity packets:

$$y_{parity} = \sum_{k=5}^{7} \boldsymbol{\alpha_{ik}} x'_k \tag{4-17}$$

8. These parity packets, along with the decoded data packets are used to form the Generator matrix, G, where

$$G = \begin{bmatrix} 1 & 0 & \ldots & 0 & \alpha_{(1+k),1} & \cdots & \alpha_{(1+k),n} \\ 0 & 1 & \ldots & 0 & \alpha_{(2+k),1} & \cdots & \alpha_{(2+k),n} \\ \vdots & \vdots & \ddots & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix} \tag{4-18}$$

where $\alpha_{i,j} \in \mathbb{F}_2$.

9. The Generator matrix, G, is used as parity symbols and the codeword vector, $\boldsymbol{r'}$, is formed where $\boldsymbol{r'} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{y}_5, \boldsymbol{y}_6, \boldsymbol{y}_7\}$.

10. The receiver applies a $(7, 4)$ error correction method to the codeword, $\boldsymbol{r}'$.

11. The receiver decodes the source information.

12. The BER is calculated.

4.5.2 Simulation results

The BER of the implicit error correction method and the standard network (without error correction) is represented in Fig. 4-6.

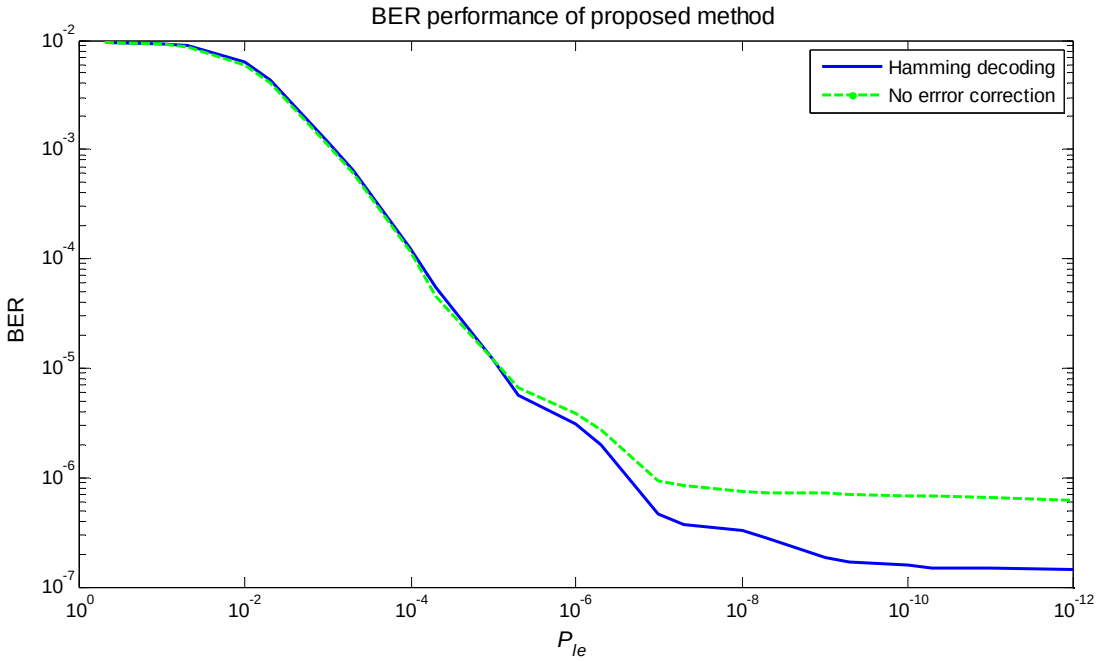


Figure 4-6: BER performance of the proposed method with Hamming decoding

In Fig. 4-6, a definite improvement in the BER can be seen for a low error probability, P_{le}. It becomes clear that an error that occurs in the network can be corrected by the receiver by waiting for additional packets and exploiting the redundant information they carry.

4.6 Discussion

4.6.1 Advantages

The literature, as discussed in Chapter 2, states that the receiver node normally collects as many channel packets as possible from a network with a $min - cut \geq k$, but discards the redundant packets and only uses k channel packets with linearly independent global encoding vectors. From the results obtained in Fig. 4-5, it becomes apparent that a network of certain size and connectivity that implements random network coding is also able to implicitly encode sufficient data that can be used for network error correction. The encoding of the source information and the addition of redundancy at the source node is therefore unnecessary.

By eliminating the redundant action of constructing parity at the source, we also eliminate the redundant action of subjecting these same parity symbols to linear combinations in the random network coding network. Therefore fewer packets are subject to the linear random network coding combinations which lead to energy conservation.

4.6.2 Disadvantages

Two disadvantages of the implicit encoding of the network can be identified at this stage:

1. The receiver has to wait longer in order to obtain the extra channel packets.
2. The decoding process has a higher computing complexity and may take longer to decode the source information.

4.6.3 Observations

The result obtained is sufficient to conclude that the method is functional, but by no means renders competitive results.

1. The network must be very large (50 nodes average) for the receiver to have an acceptable probability of decoding. This is due to the non-deterministic character of the network, where a valid set of parity symbols is not guaranteed.

2. The information packets only include a single bit of data, with an overhead four times larger, as can be seen in Fig. 4-7. This relationship is by no means ideal, so information must not be limited to the field, $\mathbb{F}_2$.

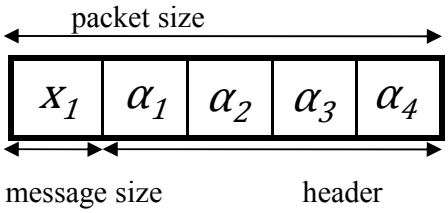


Figure 4-7: Information packet in finite field $\mathbb{F}_2$

3. The Hamming code used for decoding is only limited to binary codes, and is a very limited error correction scheme. This code must be replaced by a stronger code capable of correcting multiple errors in the finite field, $\mathbb{F}_q$.

4.6.4 Validation and verification

Conceptual model validation

In this chapter we designed the conceptual model of an error correction scheme that does not add redundancy at the source. The design is based on the theory and associated algorithms of error correction, random network coding and network error correction presented in the literature.

This enables us to validate the conceptual model of the implicit error correction method, because the theory presented in the literature is accurately implemented in the conceptual design.

Computerised model verification

By comparing the steps of the simulation setup with the process of the conceptual design, it can be seen that the steps followed in the implemented computer code are mapped directly from the conceptual design of the implicit error correction method.

Thus we can verify the implicit error correction method, because the method implemented in the simulations (computerised model) accurately represents the conceptual design constructed from the literature.

4.7 Conclusion

In this chapter we presented the implicit network error correction method. This method uses the redundant information obtained by the receiver node to apply network error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because suitable networks transmit sufficient redundancy to the receiver for error correction.

Effectively, the proposed method utilizes networks using random network coding where the network effectively acts as an error correction encoder. In these networks, the *min-cut* between the source and the receiver nodes is large enough to support the transmission of k linearly independent information packets in a network ($min - cut \geq k$), although n channel packets will be utilized by the receiver node.

In the work done in this chapter we found that this method can be successfully implemented in a network. The network parameters chosen were simply to test the method's functionality, therefore the simplest set of parameters was chosen. The network parameter used in this chapter is summarised in Table 4-2.

Table 4-2: Network parameters

Network parameter	
FEC code	Hamming (7, 4) code
Information packets	A single bit from finite field $\mathbb{F}_2$
Network *min-cut*	subset of edges, $\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut \geq k = 4$ subset of edges, $\mathcal{H}' \subset \mathcal{E}, \lvert\mathcal{H}\rvert \geq n = 7$ connected to $t \in \mathcal{V}$
Network size, r	$r \geq 50$

One aspect which is apparent in the information contained in the above table is that these parameters do not render satisfactory results. The improvement of these network parameters are presented in the following chapter.

Chapter 5:

Parameter improvement

In this chapter, we aim to improve the implicit encoding method discussed in Chapter 4. A new set of network parameters is selected and implemented in the network in order to obtain more favourable results. These results are compared to the results obtained in Chapter 4 and evaluated.

5.1 Probability of decoding

As in Chapter 4, we investigate the probability of receiving a set of valid parity symbols from a network.

The network structure and attributes described in Section 4.3 remain exactly the same for the following set of simulations. Only a few changes are made regarding the network parameters discussed below.

5.1.1. Selection of network parameters

The following changes in network parameters are made to the default simulation setup described in Chapter 4.

Forward error correction code

The Hamming code was replaced by a Low-Density-Parity-Check (LDPC) code.

LDPC codes are a well known forward error correction block code proposed by Robert Gallager in 1962 [35]. LDPC codes are normally used in large block lengths that lead to a large minimum distance, which makes these codes very effective for error detection and correction, since $t = (d_{min} - 1)/2$. LDPC codes, however, differ from classical block codes, such as the Hamming code, in the area of decoding. Classical block codes' decoding algorithms are short and algebraically designed for a less complex task, where LDPC codes are decoded by more complex iterative decoding algorithms [9]. The large block lengths used for LDPC codes result in high computational complexity which leads to the use of iterative decoding methods.

Although LDPC codes are normally used in very large block sizes, we are going to use a very short $(8, 4)$ LDPC code, so that $n = 8$ and $k = 4$. This choice of code is sufficient, because we want to improve the decoding scheme of the proposed method. The LDPC code implemented in the simulation will make use of the sum-product decoding algorithm, described in [9]. The iterative decoding process will ensure that the additional packets collected by the receiver will render more valid sets of parity symbols. This will increase the probability of decoding and decrease the necessary network size.

Information packets

LDPC codes can be defined over any finite field, $\mathbb{F}_q$, but the majority of literature focuses on the use of LDPC codes in the finite field, $\mathbb{F}_2$. The information packets used for this simulation will therefore still only contain data from a finite field $\mathbb{F}_2$.

Network connectivity

The connections in the network will be randomly generated with a connectivity probability $p' = 0.5$.

Network diameter and average distance

The network diameter will be approximately $D \geq 2\frac{\log(n-1)}{\log\sqrt{n}}$, and the average distance $\frac{rD}{2(r-1)} \leq D_{ave} \leq D$.

Network min-cut

For the (8, 4) LDPC code used, the network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, must contain a subset of edges, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| \geq k = 4$ to have a $min - cut \geq k = 4$. Also, for the implicit error correction method to be successfully implemented, network, $\mathcal{G}$, must also contain subset of edges $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n = 8$ connected to the receiver, $t \in \mathcal{V}$.

5.1.2. Simulation setup

A synchronous network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, implementing random network coding is generated for network sizes ranging from 7 intermediate nodes to a network size that guarantees valid parity sets. For each network generation 1 000 simulation iterations are run for each parameter set. Simulations are repeated until a network size is determined that renders a guaranteed set of parity symbols.

The same simulation setup is used as in Section 4.4.1.

5.1.3. Simulation results

The average results are represented in Fig. 5-1 in respect to the mean of each set.

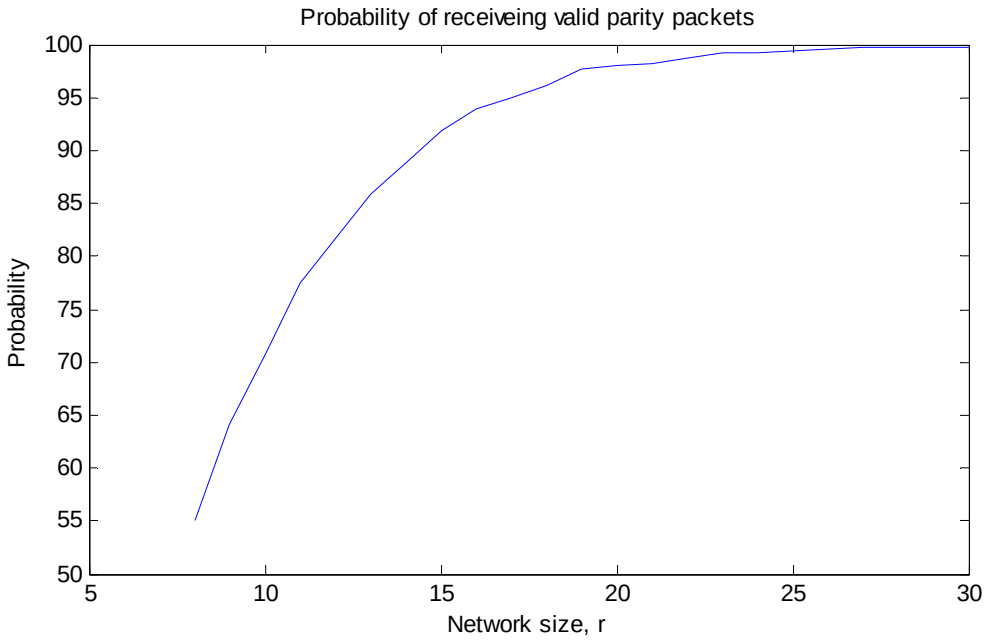


Figure 5-1: Probability of receiving valid parity packets

From the above graph, the probability of receiving valid parity packets can be deduced for the network size *r* and represented in Table 5-1.

Table 5-1: Probability of receiving valid parity packets

Network size *r*	**Probability of valid parity packets**
$>$ 10	$>$ 70 %
10 – 15	70 % – 90 %
16 – 23	90 % – 98 %
$<$ 23	$>$ 99 %

From the above simulation of a (7,4) LDPC code, one can expect to receive channel packets that can be used as parity packets for error correction less than 70 % of the time when the network has less than 10 intermediate nodes. When the network becomes bigger, the

probability of receiving valid parity packets increases until about 99% for networks of sizes 23 and larger.

5.2 Error correction ability

Based on the above obtained result, we choose a 30-node network for the following simulation. Just as before, this is done to guarantee a valid set of parity symbols under the predetermined network parameters and network size discussed, for a 30-node network renders valid parity packets 99,77% of the time. This will make it possible to evaluate the error correction ability of the proposed method if implemented successfully in a network.

5.2.1 Simulation setup

In order to test if the changes in network parameters lead to improved error correction, the same implicit network encoding simulation is run as in Section 4.6.

The network used for this simulation, however will contain only 30 nodes. Again, 100 000 simulation iterations are run for each parameter pair with a specified link error probability, P_{le}, ranging from 0.5 to 10^{-12}.

5.2.2 Simulation results

The BER results of using LDPC decoding are added to the results obtained in Section 4.6 and shown in Fig. 5-2.

In Fig. 5-2, a definite improvement in the BER can be seen for a low link error probability, P_{le} of the LDPC scheme. Clearly, when the link error probability is low, the LDPC scheme can correct a large range of single bit errors; much better than the case with the Hamming decoding scheme.

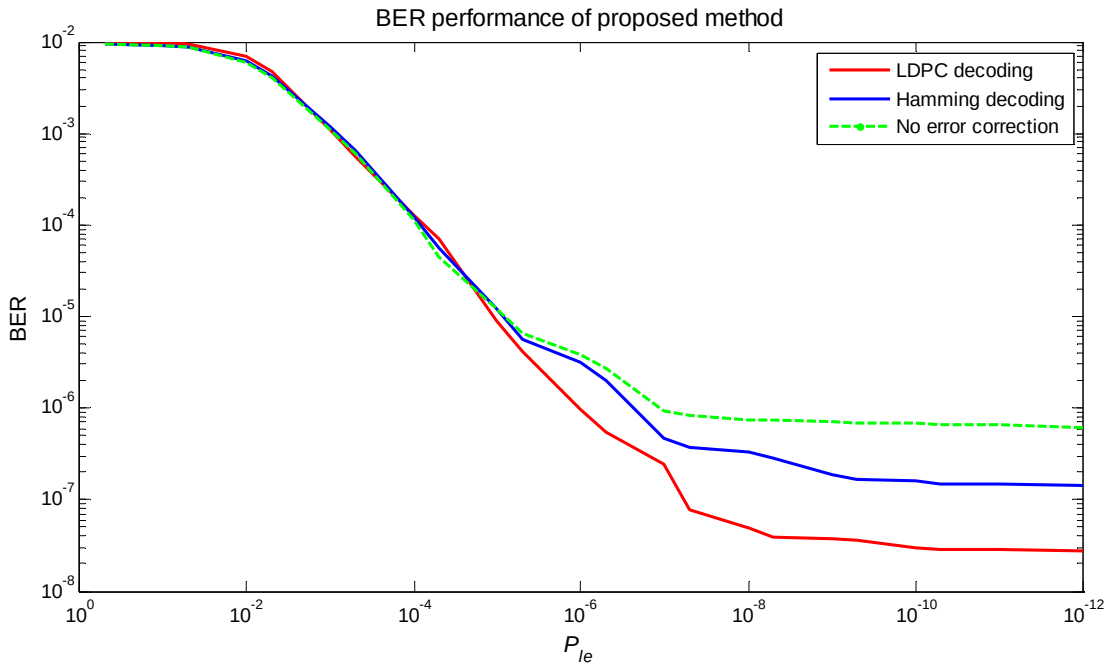


Figure 5-2: BER performance of proposed method with LDPC decoding

5.3 Discussion

5.3.1 Advantages

From the results obtained in Fig. 5-2, it can clearly be seen that by using a more effective error correcting code, the implicit error correction capabilities of a network can be harnessed more effectively.

This is due to the fact that LDPC code's iterative decoding algorithm can use more of the additionally collected channel packets for parity. This increases the probability of decoding of the information implicitly encoded in the network and therefore improves the method.

5.3.2 Disadvantages

The biggest disadvantage concerned with the use of LDPC decoding is the increase in decoding complexity. The iterative decoding algorithms are mathematically more complex and therefore more time consuming.

5.3.3 Observations

The result obtained in this simulation is an improvement to the result in Chapter 4, but can still be further improved.

1. The network size is reduced. For the network to create a definitive set of parity symbols, it can have about 30 nodes.

2. In this chapter, the information sent is still limited to bits. The network parameters must be altered to information packets containing symbols so that more data can be sent through the network more effectively.

3. LDPC codes are a more effective error correcting scheme than the Hamming code, but it must be replaced by a stronger code capable of correcting multiple errors in the finite field, $\mathbb{F}_q$.

5.4 Conclusion

In this chapter, we optimized the implicit encoding method by selecting a network parameter set that renders better results. It is clear that the use of a more effective error correcting code (LDPC code instead of Hamming code) exploits the implicit error correction capabilities of a network more effectively.

The work done in this chapter proves that this method, when thoroughly optimized, can present numerous advantages when compared to the existing concatenated network error correction schemes for network coding presented in the literature. The network parameters used in this chapter are summarised in Table 5-2.

Table 5-2: Network parameters

Network parameter	
FEC code	$(8, 4)$ LDPC code
Information packets	A single bit from finite field $\mathbb{F}_2$
Network *min-cut*	subset of edges, $\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut \geq k = 4$ subset of edges, $\mathcal{H}' \subset \mathcal{E}, \lvert\mathcal{H}\rvert \geq n = 8$ connected to $t \in \mathcal{V}$
Network size, r	$r \geq 30$

The final improvement iteration of these network parameters is presented in Chapter 6, along with the advantages and disadvantages compared with the existing methods.

Chapter 6:

Final improvement and analysis

The aim of this chapter is to produce an improved scheme by further developing the network parameters; summarising the characteristics of this method and comparing them with that of existing concatenated network error correction schemes. This chapter concludes by determining the advantages and disadvantages of the implicit error correction scheme.

6.1 Probability of decoding

The goal of our third and final improvement simulation is to provide favourable results so that the proposed method can be compared with the methods presented in the literature. Again, the network structure and attributes described in Section 4.3 remain the same for this simulation.

In the first set of simulations, we investigate the probability of receiving a set of valid parity symbols from a network.

6.1.1 Selection of network parameters for improvement

The following network parameters are chosen for the last improvement iteration.

Forward error correction code

LDPC codes are limited in the fact that they are mainly used in the finite field, $\mathbb{F}_2$. Reed Solomon (RS) codes are also linear block codes, but instead of containing bits, they contains symbols over the finite field $\mathbb{F}_q$ [9].

The values of the information and parity symbols of a RS code are elements of Galois fields. The RS code is implemented generically so that the user can specify the (n, k) block code desired. The (n, k) RS code has a symbol length of m in a Galois field of $GF = 2^m$. This RS code in the $GF\ (2^m)$ can be defined as a (n, k, t) code with a block length of n where $n = 2^m - 1$, and dimension $k = n - 2t$, where $2t$ are the number of parity-check symbols. The error correcting capability of an RS code is $\frac{n-k}{2}$.

Information packets

RS codes can be implemented over the finite field, $\mathbb{F}_q$, in a Galois field $GF\ (2^m)$. The information packets used for this simulation contain symbols of length m from a finite field $\mathbb{F}_q$, in $GF\ (2^m)$.

References [11] as well as [25] assume that the global encoding vector is sent along with $\boldsymbol{x}$ in its header. We adopt the same assumption and send the global encoding vector inside the

packet header. According to [25], however, the overhead (header) sent with the information packet is negligibly small if the packet size is sufficiently large.

Network min-cut

For the (n, k) block code used, the network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, must contain a subset of edges, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| \geq k$ to produce a $min - cut \geq k$ and at least k edge-disjoint paths between the source and receiver. Also, for this implementation, network, $\mathcal{G}$, must contain another subset of edges $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n > k$ connected to the receiver, $t \in \mathcal{V}$.

Network diameter

The network diameter will be approximately $D \geq 2\frac{\log(n-1)}{\log\sqrt{n}}$, and the average distance $\frac{rD}{2(r-1)} \leq D_{ave} \leq D$.

Network connectivity

The connections in the network will be randomly generated with a connectivity probability $p' = 0.5$.

6.1.2 Simulation setup

Random synchronous networks, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges in the network using random network coding are generated for network sizes, r, ranging from n intermediate nodes upward. For each network generation 100 000 simulation iterations are run for each parameter set.

The same simulation setup is used as in Section 4.5.1 with a few changes:

1. A block of input data, $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ (symbols in $GF\ (2^m)$), is generated at the source. These information packets contain information symbols as well as a global encoding vector.

2. The single source node, $S \in \mathcal{V}$, transmits the k information packets in to the network.

3. The information symbols are encoded within the network.

4. The receiver tests if it is able to collect k channel packets, $y_{data} = y_1, y_2, \dots, y_k$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{K=1}^{k} \boldsymbol{\alpha}_{iK} \boldsymbol{x}_K \ , i = 1,2, \dots k \tag{6-1}$$

5. If possible: the receiver tests if it is able to collect an additional n channel packets, $y_{parity} = y_{k+1}, \dots, y_n$, with independent global encoding vectors that will act as the parity packets:

$$y_{parity} = \sum_{\eta=k+1}^{n} \boldsymbol{\alpha}_{i\eta} \boldsymbol{x'}_{\eta} \ , i = (k = 1), (k + 2), \dots, n \tag{6-2}$$

6. If possible, the network is marked as a network where the received combinations can be used to decode the original information as well as form valid sets of parity symbols for error correction.

6.1.3 Simulation results

The average results are represented in Fig. 6-2 in respect to the mean of each set.

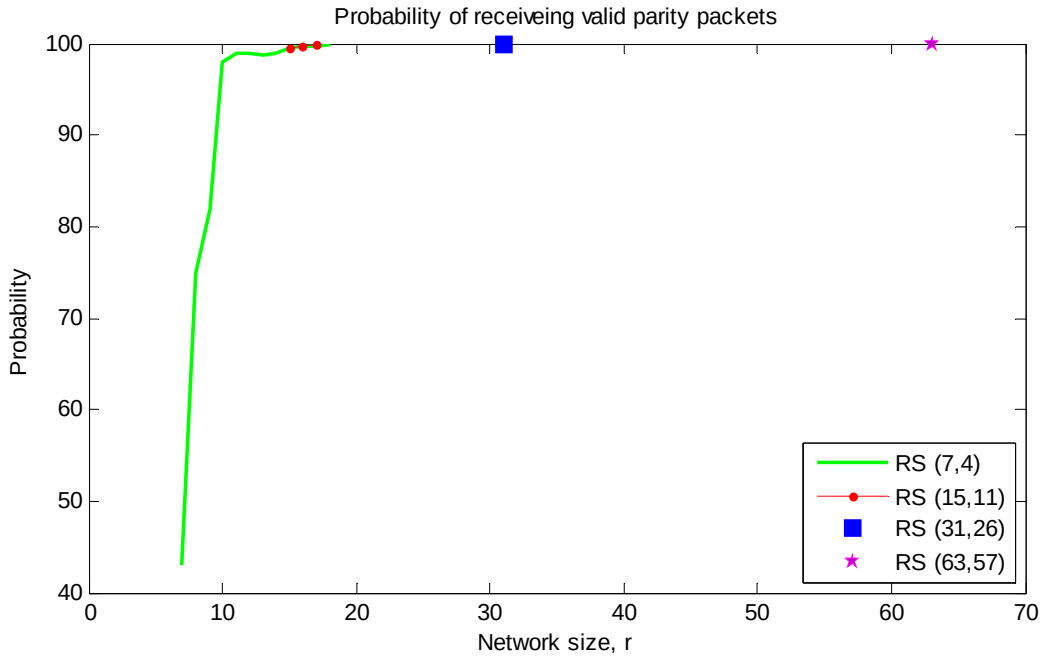


Figure 6-1: Probability of receiving valid parity packets

It can be seen from the results displayed in Fig. 6-1 that the improvement regarding the probability of decoding is successful. From the obtained graph, the probability of receiving valid parity packets for different (n, k) RS codes can be deduced for the network size r and represented in Table 6-1.

Table 6-1: Probability of receiving valid parity packets

Network size r	**(n, k) RS code**	**Probability of valid parity packets**
> 8	(7,4)	$> 75\ \%$
$\mathbf{8 - 18}$	(7,4)	$75\ \% - 98\ \%$
< 18	(7,4)	$> 99\ \%$
$\geq \mathbf{17}$	(15,11)	$> 99\ \%$
$\geq \mathbf{31}$	(31,26)	$> 99\ \%$
$\geq \mathbf{63}$	(63,57)	$> 99\ \%$

From the above simulation results for different RS codes, it can be seen that for codes with $GF\ (2^m),\ m > 4$ the minimum amount of intermediate nodes needed to satisfy the network constraints, $r = 2^m$, guarantees a valid set of parity symbols.

The network size parameter, r, for the following improvement simulations will be set to the minimum amount of intermediate nodes, $r = 2^m$, that guarantee the probability of decoding.

6.2 Error correction ability

The final simulations evaluating the error correction ability of the proposed method will be done on information symbols in different Galois fields. The network sizes, r, chosen for each simulation are represented in Table 6-2.

Table 6-2: Network size parameters

	$GF\ (2^3)$	$GF\ (2^4)$	$GF\ (2^5)$	$GF\ (2^6)$
(n, k) block code	(7, 4)	(15, 11)	(31, 26)	(63, 57)
Symbol length, m	3	4	5	6
Network size, r	18	17	31	63

6.2.1 Simulation setup

The implicit network encoding simulation will continue to be implemented as described in Section 4.5. As before, 100 000 simulation iterations are run for each chosen parameter pair.

The simulation runs in the following steps:

1. A block of input data, $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ (symbols in $GF\ (2^m)$), is generated at the source. These information packets contain information symbols as well as a global encoding

vector.

2. The single source node, $S \in \mathcal{V}$, transmits the k information packets in to the network.

3. The information symbols are encoded within the network.

4. The receiver collects k channel packets, $y_{data} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_k$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{K=1}^{k} \boldsymbol{\alpha}_K \boldsymbol{x}_K \tag{6-3}$$

5. The receiver decodes the source packets and forms the following matrix:

$$\begin{matrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_k} \end{matrix} \begin{vmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{6-4}$$

6. The BER is calculated.

7. The receiver collects an additional $(n-k)$ channel packets, $y_{parity} = \boldsymbol{y}_{k+1}, \dots, \boldsymbol{y}_n$, with linearly independent global encoding vectors that that will act as the parity packets.

$$y_{parity} = \sum_{\eta=k+1}^{n} \boldsymbol{\alpha}_{\boldsymbol{\eta}} \boldsymbol{x}'_{\eta} \tag{6-5}$$

8. These parity packets, along with the decoded data packets are used to form the Generator matrix, G, where

$$G = \begin{bmatrix} 1 & 0 & \dots & 0 & \alpha_{(1+k),1} & \cdots & \alpha_{(1+k),n} \\ 0 & 1 & \dots & 0 & \alpha_{(2+k),1} & \cdots & \alpha_{(2+k),n} \\ \vdots & \vdots & \ddots & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_{n,1} & \dots & \alpha_{n,n} \end{bmatrix} \tag{6-6}$$

where $\alpha_{i,j} \in \mathbb{F}_2$.

9. The Generator matrix, G, is used as parity symbols and the codeword vector, $\boldsymbol{r}'$, is formed where $\boldsymbol{r}' = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k, \boldsymbol{y}_{1+k}, \boldsymbol{y}_{2+k}, \dots, \boldsymbol{y}_n\}$.

10. The receiver applies a (n, k) error correction method to the codeword, $\boldsymbol{r}'$.

11. The receiver decodes the source information.

12. The BER is calculated.

6.2.2 Simulation results

The BER performance of the implicit encoding network with (n, k) Reed Solomon decoding is added to the results obtained in the previous two chapters and are represented in Fig. 6-3.

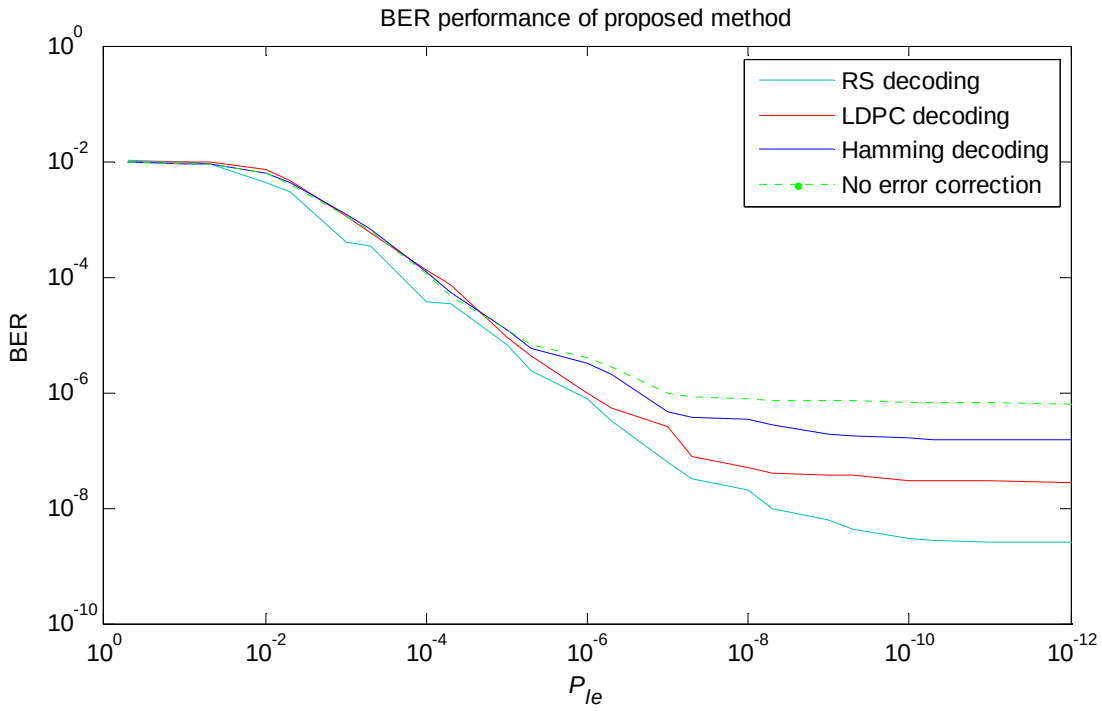


Figure 6-2: BER performance of proposed method with RS decoding

In Fig. 6-3, a definite improvement in the BER can be seen for a low link error probability, P_{le} of the RS scheme. At a low link error probability, the RS code can correct a large range of errors. It is clear from the results illustrated in Fig. 6-3 that the BER of the

network is improved. It can be seen that an order of nearly 10^3 can be gained by simply instructing the receiver to wait longer and obtain additional channel packets for error correction.

6.3 Discussion

In the previous three chapters we presented comprehensive results that the implicit encoding abilities of a network can be harnessed and used for error correction. We improved this method by using three improvement iterations where the iterations were tested, analyzed, and improved once more. A large range of parameters was tested by performing between 1000 and 1 000 000 iterations of each parameter set.

The final set of network parameters used in this chapter is summarised in Table 6-3.

Table 6-3: Chosen network parameters

Network parameter		
FEC code	(n, k) Reed Solomon code	
Information packets	k Packets containing symbols of length m from a finite field $\mathbb{F}_q$, in $GF\ (2^m)$	
Network *min-cut*	subset of edges, $\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut\ \geq\ k$ subset of edges, $\mathcal{H}' \subset \mathcal{E}, \lvert\mathcal{H}\rvert \geq\ n > k$ connected to $t \in \mathcal{V}$	
Network size, r	RS $(GF\ (2^m), m \leq 4)$ $r \geq 17$	RS $(GF\ (2^m), m > 4)$ $r \geq n$

With the conclusion of the final parameter improvement on the implicit error correction scheme, we summarise the characteristics of this method and compare it with that of existing

concatenated network error correction schemes. Comparing the implicit error correction method with the concatenated network error correction method will enable us to determine the implicit error correction method's requirements in Chapter 7.

6.3.1 Comparison of network error correction methods

The comparison between the implicit error correction method and the concatenated error correction method is shown in Table 6-4.

Table 6-4: Comparison of network error correction methods

	Concatenated error correction method	**Implicit error correction method**
Encoding	Concatenated encoding at source.	Implicit encoding in network.
Packets transmitted	k Linearly independent information packets coded to n coded packets.	k Linearly independent information packets.
Network	Small network. $\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut \geq n$	Large network, high connectivity. $\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut \geq k$ $\mathcal{H}' \subset \mathcal{E}, \lvert\mathcal{H}\rvert \geq n$ connected $t \in \mathcal{V}$
Decoding	Receiver can always decode.	Receiver cannot always decode.
Packets selected	n Channel packets.	n Channel packets.
FEC code	Predetermined.	Receiver decides based on channel packet reception.
Network error correction	Source and receiver communicate network error correction scheme. Both implement scheme.	Receiver implements independent decoding. Only receiver implements scheme, with no communication to source.

Symbol rate, *R*	$\frac{n}{n} = 1$	$\frac{k}{n} \leq 1 , k < n$

When comparing these two methods, the advantages and disadvantages of the implicit error correction method can be determined. These advantages and disadvantages are discussed subsequently.

6.3.2 Advantages

Independent decoding

One of the biggest advantages of exploiting the implicit encoding abilities of random network coding is the fact that the source does not implement error correction. Only the receiver applies error correction to the information. This means that the receiver can apply any error correction code it chooses (example: Hamming, LDPC, Reed Solomon etc.) without informing the source. The receiver can base its decision of the chosen decoding algorithm on the amount of redundancy packets received, the required decoding time or the computational complexity. The biggest constraint for this independent decoding is the size and connectivity of the network, which in our case act as the network error correction encoder.

The size of the network must be chosen so that the receiver has a high probability of receiving a valid set of parity packets. A valid set of parity symbols will ensure that the receiver will be able to apply an error correction code and decode the transmitted data. If the receiver cannot collect a valid set of parity symbols, the implicit encoding capabilities of the network will have no use.

In other words: For a network of sufficient size and connectivity, the receiver can practice independent decoding where error correction can be applied by simply collecting channel packets selectively. No communication regarding the chosen FEC code has to be established between the receiver and the source node. The receiver can apply any error correction scheme to the received data without informing the source. Its decision of error correction algorithm can be based solely on the channel packets it collects.

Energy efficiency

This method eliminates the redundant action of constructing parity using linear combinations of the information symbols at the source and then subjecting these same parity symbols to linear combinations again in the random network coding network. This means that less information packets are transmitted into the network and fewer packets in the network are subject to the linear random network coding combinations.

The saving of energy may be of interest to energy constraint networks such as wireless sensor networks.

Saving in bandwidth

This implicit error correction method maps a set of transmitted symbols to a set of channel symbols resulting in symbol rate, R, of less than 1, where

$$R = \frac{transmitted\ symbols}{received\ symbols} = \frac{k}{n} < 1\,, k\ <\ n\,. \tag{6-7}$$

Reduction in network min-cut requirements

Our proposed scheme allows for greater error correction ability for a random network coding network with a specific *min-cut*, compared with the concatenated network error correction methods proposed in literature.

6.3.3 Disadvantages

Probability of decoding

It is possible that the additional channel packets obtained by the receiver from an error prone network, may not be sufficient to act as valid parity symbols. This may prevent the receiver from decoding the information successfully and render the method ineffective.

Computational complexity

The code implemented by the receiver in the implicit error correction method to select k channel packets plus an additional set of $(n-k)$ packets to form codeword vector, $\boldsymbol{r}'$,

requires a more computational complex algorithm than used by the concatenated network error correction method.

6.4 Conclusion

The work done in this chapter presents comprehensive results that the improved implicit error correcting method can present numerous advantages compared to the existing error correction methods.

These advantages, however, can only be harnessed when the network environment adheres to specific network requirements. In Chapter 7 we determine the network requirements that will enable the implicit error correction scheme to obtain better results than that obtained by concatenated schemes presented in the literature. We also determine the characteristics associated with the implementation of this method.

Chapter 7:

Method requirements, characteristics and performance

In this chapter the proposed method is evaluated along with the existing network error correction scheme and a case without error correction by simulating it in the same network. We assess the computational complexity, error correction capacity and BER performance to determine the network requirements for and characteristics of the implementation of the implicit method. The BER performance of the implicit method is also analysed to determine if the method complies with the initial specifications.

7.1 Introduction

In order to see if the implicit error correction method can be used as an alternative to existing network error correction schemes, a series of comparisons of the different methods is performed. These comparisons are done so that the network requirements for and characteristics of the proposed method can be determined. The network requirements enable us to evaluate if the implicit error correction method can be successfully implemented in a specific scenario.

We investigate the following three methods:

1. Method without error correction implemented:

   k Information packets are transmitted over the network. The receiver collects k linearly independent channel packets for decoding.

2. Concatenated error correction method:

   This method is introduced in Chapter 3 where k information packets are encoded at the source, $S \in \mathcal{V}$, to form a codeword vector, $\boldsymbol{r}$, in $\mathbb{F}_q$ where $\boldsymbol{r}$ consists of n coded packets.

3. Implicit error correction method:

   This method is introduced in Chapter 4 where k information packets are transmitted over the network. The implicit encoding ability of the network is exploited and the receiver, $t \in \mathcal{V}$, collects channel packets to construct a codeword, $\boldsymbol{r}'$, of length n in $\mathbb{F}_q$. The added $(n-k)$ parity packets are additional channel packets used by the receiver for error correction.

These methods are compared to determine the following:

- the computational complexity algorithms used for error correction
- the network error correction capacity a network offers for different error correction methods
- the network requirements needed for method implementation in a network.

7.2 Computational complexity [41][43-46]

The first analysis done on the implicit and concatenated methods as well as no error correction scheme is the calculation of the computational complexity of the algorithms by performing an asymptotic analysis on the code used in the simulations.

Computational complexity helps to characterise the relationship between the number of data elements in the algorithm and the usage of available resources. This characterisation will help us to determine how much additional resources our proposed method would require in comparison with the existing concatenated schemes.

Note that different algorithms exist for a single process in the literature in the pursuit of algorithm optimisation. Our aim is not to evaluate the optimal algorithm for each of the processes, but to compare the complexity of the processes needed for the different methods when implemented in a random network coding network. The computational complexity of the random network coding network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is not calculated, because the same network is used in the simulations of all the methods.

7.2.1. Asymptotic Analysis

We perform asymptotic analysis on the following sections of code in terms of k and n used in the simulations:

- Encoding: the process of forming a codeword, $\boldsymbol{r}$, of length n from the k information symbols, $\boldsymbol{m}$, with the use of a $k \times n$ generator matrix, G.

- Transmission: the process of transmitting the symbols, $\boldsymbol{r}$, into the network over the subset of edges, $\mathcal{H} \subset \mathcal{E}$.

- Collection: the process where the receiver collects the channel packets from the subset of edges, $\mathcal{H}' \subset \mathcal{E}$ to form codeword, $\boldsymbol{r}'$.

- Linearly independent sets selection: the process performed by the receiver in the implicit error correction method where it must select k channel packets plus additional

$(n-k)$ channel packets as parity packets from the subset of edges, $\mathcal{H}' \subset \mathcal{E}$, to form codeword, $\boldsymbol{r}'$.

- Error correction and decoding: the process of obtaining a syndrome vector, $\boldsymbol{s}$, with the use of the $(n-k) \times n$ parity check matrix, H; and the correction of possible errors by using the syndrome vector to acquire the error free codeword, $\boldsymbol{r}$. Decoding is the process of obtaining the original k information symbols by matrix multiplication with the generator matrix, G.

Encoding

For our simulations, conventional encoding is used in order to form a codeword, $\boldsymbol{r}$, of length n from the k information symbols. The $k \times n$ generator matrix, G is used.

The given pseudo code is an extraction from the encoding algorithm used in the simulations. Note that this is not the complete algorithm, but simply a portion containing the needed information in order to perform an asymptotic analysis.

Algorithm 1

1: procedure $ENCODING\ IN\ (\mathcal{G}, m, r)$

2: $\boldsymbol{r} \leftarrow \boldsymbol{m} \times G$ ▷Matrix multiplication

This encoding process is only executed by the concatenated scheme. For the no error correction scheme as well as the implicit error correction scheme only the k information packets are transmitted into the network. This complexity is equal to $\mathcal{O}(n^2)$ for block codes with code length, n due to the matrix multiplication.

Transmission

All the methods transmit the symbols into the network over the subset of edges, $\mathcal{H} \subset \mathcal{E}$. The following is an extraction from the transmission algorithm used in the simulations.

Algorithm 2

1: procedure $TRANSMISSION\ IN\ (\mathcal{G}, \mathcal{V}, \mathcal{E}, r)$
2: **for all** $r \in \boldsymbol{r}$ **do** ▷all symbols tx into network
3: **for all** $\mathcal{H} \subset \mathcal{E}$ **do** ▷edges connected to $S \in \mathcal{E}$
4: $node(r, \mathcal{H}) \leftarrow data(r)$ ▷update data
5: $update\ overhead$
6: **end for**
7: **end for**

For the concatenated scheme, the n coded packets are transmitted over the subset of edges, $\mathcal{H} \subset \mathcal{E}$, where $|\mathcal{H}| \geq n$. It can be derived that the complexity of this process is $\mathcal{O}(n^2)$. For the implicit error correction and no error correction schemes, only k packets are transmitted over the subset of edges, $\mathcal{H} \subset \mathcal{E}$, where $|\mathcal{H}| \geq k$. The complexity equals $\mathcal{O}(k^2)$.

Collection

In the collection process, the receiver collects the channel packets from the subset of edges $\mathcal{H}' \subset \mathcal{E}$ and then selects a set of channel packets that it will use for error correction and decoding. The overhead information of the channel packets is stored in overhead matrix, T. The coding complexity of the collection of channel packets by the receiver can be calculated using this extraction of pseudo code.

Algorithm 3

1: procedure $COLLECTION IN\ (\mathcal{G}, \mathcal{V}, \mathcal{E}, r')$
2: **for all** $\mathcal{H}' \subset \mathcal{E}$ **do** ▷edges connected to $t \in \mathcal{V}$
3: $t\_node \leftarrow node(:, \mathcal{H}')$ ▷ receiver t collects packets
4: $T(:, \mathcal{H}') \leftarrow overhead(:, \mathcal{H}')$ ▷create overhead matrix, T
5: **end for**
6: **if** $|\mathcal{H}'| = rank\ (T)$ **then**
7: **for** $j \leftarrow 1, |r'|$ **do**
8: $r' \leftarrow T(:, |r'|\ )$ ▷form codeword
9: **end for**
10: **end if**

When looking at the pseudo code above, the complexity of the case without error correction equals $\mathcal{O}(k)$, where $|\boldsymbol{r}'| = k$ channel packets are selected by the receiver from edges $\mathcal{H}' \subset \mathcal{E}$. The existing concatenated method collects $|\boldsymbol{r}'| = n$ channel packets for error correction to form the codeword, $\boldsymbol{r}'$, where the complexity equals $\mathcal{O}(n)$. For the implicit error correction method, the collection of the channel packets works in the same way, so the computational complexity also equals $\mathcal{O}(n)$, but the selection of $|\boldsymbol{r}'| = n$ packets works somewhat differently.

Linearly independent sets selection

The receiver of the implicit error correction method requires the collection of a large set of channel packets and selection of two sets of linearly independent packets of size k and $(n-k)$ respectively from the collected set channel packets. A pseudo code extraction of the method is shown.

Algorithm 4

1: procedure $LINEARLY\ INDEPENDENT\ SETS\ SELECTION\ IN\ (\mathcal{G}, \mathcal{V}, \mathcal{E}, r')$
2: **for all** $\mathcal{H}' \subset \mathcal{E}$ **do**
3: $t\_node \leftarrow node(:, \mathcal{H}')$ ▷edges connected to $t \in \mathcal{V}$
4: $T(:, \mathcal{H}') \leftarrow overhead(:, \mathcal{H}')$ ▷packets receiver collects
5: **end for** ▷create overhead matrix, T
6: $[Q, R, E] = qr(T)$ ▷QR decomposition
7: $indep\_data \leftarrow E(1, k)$ ▷find lin. indep. columns
8: **for** $j \leftarrow 1, k$ **do**
9: $r'(:, j) \leftarrow T(:, indep\_data(:, j))$ ▷form codeword $r'1^{st}$ part
10: $T(indep\_data(:, indep\_data(:, j)) \leftarrow 0$ ▷update overhead matrix, T
11: **end for**
12: $[Q, R, E] = qr(T)$ ▷QR decomposition
13: $indep\_parity \leftarrow E(1, (n-k))$ ▷find lin. indep. columns
14: **for** $j \leftarrow k+1, n$ **do**
15: $r'(:, j) \leftarrow T(:, indep\_parity(:, j))$ ▷form codeword $r'2^{nd}$ part
16: **end for**

The implicit method also collects channel packets from edges $\mathcal{H}' \subset \mathcal{E}$ and stores the overhead information in overhead matrix, T. In order to obtain two sets of linearly independent channel packets, orthogonal-triangular decomposition is performed on the $m \times n'$ overhead matrix, T. The `[Q,R,E] = QR()`MATLAB function produces a permutation matrix, E, that helps us to determine the sets of linearly independent columns of the overhead matrix, T[40]. When the independent columns are determined, it is easy to produce a linearly independent channel packet set that is used for data decoding. Performing this function on the $m \times n'$ overhead matrix has a coding complexity of $\mathcal{O}(n^2)$.

This function has to be performed again to create a linearly independent channel packet set for parity packets from the columns of the remaining overhead matrix. These two sets of channel packets then form codeword, $\boldsymbol{r}'$. The computational complexity of the whole section equals $\mathcal{O}(k^2 + n^2)$.

Error correction and decoding

The conventional error correction process is used where a syndrome vector, $\boldsymbol{s}$, is obtained through matrix multiplication of the codeword vector, $\boldsymbol{r}$, with the $(n-k) \times n$ parity check matrix, H; and used for error correction to acquire the error free codeword vector, $\boldsymbol{r}$. The decoding of the corrected codeword vector to obtain the original k information symbols simply requires matrix multiplication with the generator matrix, G. This procedure is shown in the following pseudo code extraction that depicts only the computations that have significance regarding the computational complexity.

Algorithm 5

1:	procedure $ERROR\ CORRECTION\ \&\ DECODING\ IN\ (\mathcal{G}, \mathcal{V}, \mathcal{E}, r', r, m)$	
2:	$\boldsymbol{s} \leftarrow \boldsymbol{r}' \times H^T$	▷create syndrome vector, s
3:	$correct\ possible\ errors$	
4:	$\boldsymbol{m} \leftarrow \boldsymbol{r} \times G^T$	▷decode to original message

This process followed is a straightforward linear block code algorithm and the complexity of this code equals $\mathcal{O}(n^2)$.

7.2.2. Computational characteristics

Table 7-1 summarises the respective computational complexities of the three methods in network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The following can be seen from Table 7-1 regarding the computational complexities of the implicit error correction method:

1. Source node, $S \in \mathcal{E}$: The computational complexity of the processes executed at the source node is the same as for a method where no error correction is applied, because no encoding is done and only k packets are transmitted into the network.

2. Receiver node, $t \in \mathcal{E}$: The computational complexity at the receiver node for the collection, error correction and decoding algorithms is the same as for the concatenated scheme. The process of selecting linearly independent sets, however, is an additional process implemented by the implicit method that is computationally demanding.

Table 7-1: Summarisation of computational complexities

	No error correction	**Concatenated error correction**	**Implicit error correction**
Encoding	-	$\mathcal{O}(n^2)$	-
Transmission	$\mathcal{O}(k^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(k^2)$
Collection	$\mathcal{O}(k)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Linearly independent sets selection	-	-	$\mathcal{O}(k^2 + n^2)$
Error correction and decoding	-	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$

From the asymptotic analysis done, we see that higher computational complexity is required for the implicit method than for the concatenated error correction scheme. The higher computational complexity characteristic at the receiver node of the network can lead to a longer computational time needed for the implementation of the implicit method. Also, as

stated in Chapter 6, the implicit error correction method requires that the receiver must wait longer, compared with the concatenated method, to obtain a large enough set of channel packets from the network to construct two sets of linearly independent channel packets. This further increases the computational time characteristic for the implementation of the implicit error correction method.

7.3 Error correcting ability

In the conceptual design in Chapter 4, we determined that when a random network coding network has a $min - cut \geq k$, it is able to support the transmission of the k information packets over the network.

This is an advantage of the implicit error correcting method, because a network with $min - cut \geq k$ can provide the capacity for network error correction of a (n, k) linear code, where for the existing concatenated scheme, the network must have $min - cut \geq n$, to support the same network error correction. We calculate the maximum possible network error correction capacity that the network can provide for a (n, k) linear code while using the implicit and the existing concatenated method. This will enable us to determine the network *min-cut* requirement for the implementation of the implicit error correction method for (n, k) error correction, as well as the network characteristics associated with it.

Note that the *min-cut* of the network is not the only requirement for the network to provide a successful network error correction platform for the implementation of the implicit error correction scheme. In this section, however, we study the network error correction capacity that a network can offer for the implementation of a specific (n, k) linear code .

7.3.1 Simulation setup

Two parallel simulations are run to determine the network error correction capacity of a network when implementing the implicit method and the concatenated method respectively.

For this simulation, we do *not* take into account:

- the number of intermediate nodes, r, in the network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- the number subset of edges $\mathcal{H}' \subset \mathcal{E}$ connected to $t \in \mathcal{V}$
- the possibility of not receiving sufficient channel packets as valid parity packets

We consider the synchronous network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges. The network contains a subset of edges, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}|$ to produce a $min{-}cut \geq |\mathcal{H}|$. The source, $S \in \mathcal{V}$, generates a (n, k) classic linear code described in [**9**] where

- length, $n = 2^l - 1$
- dimension, $k = 2^l - l - 1$, where $(l > 2)$
- $d_{min} \leq n - k + 1$

The minimum distance of the linear code, d_{min}, can also be represented as

$$d_{min} \leq l + 1 \tag{7-1}$$

Implicit error correction method

For the implicit error correction method, we know that a network with $min{-}cut \geq k$ can provide the network error correction capacity for a (n, k) linear code, therefore

$$min - cut \geq k = 2^l - l - 1. \tag{7-2}$$

The simulation implementing the implicit error correction method runs in the following steps:

1. A network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is generated with a $min - cut \geq 2^l - l - 1$.

2. A (n, k) linear code is generated at the source, $S \in \mathcal{V}$, with:
   - length, $n = 2^l - 1$
   - dimension, $k = 2^l - l - 1$, where $(l \geq 2)$
   - $d_{min} \leq n - k + 1 = l + 1$.

3. The value of $l > 2$ is generated.

4. The *network min-cut parameter* for the specific l *parameter* is generated, where $min - cut \geq k = 2^l - l - 1$.

5. The corresponding k and n values for the specific *min-cut* and l *parameters* are determined, where

$$k = min - cut \tag{7-2}$$

and

$$n = k + l. \tag{7-3}$$

6. Steps 3 to 5 are repeated.

Concatenated error correction method

For the concatenated error correction method, we know that a network with $min-cut \geq n$ can provide the network error correction capacity for a (n, k) linear code, therefore

$$min - cut \geq n = 2^l - 1. \tag{7-4}$$

The simulation implementing the concatenated error correction method runs in the following steps:

1. A network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is generated with a $min - cut \geq 2^l - 1$.

2. A (n, k) linear code is generated at the source, $S \in \mathcal{V}$, with:
   - length, $n = 2^l - 1$
   - dimension, $k = 2^l - l - 1$, where $(l \geq 2)$
   - $d_{min} \leq n - k + 1$.

3. The value of $l > 2$ is generated.

4. The *network min-cut parameter* for the specific l *parameter* is generated, where $min - cut \geq n = 2^l - 1$.

5. The corresponding k and n values for the specific *min-cut* and l *parameters* are determined, where

$$n = min - cut \tag{7-5}$$

and

$$k = n + l. \tag{7-6}$$

6. Steps 3 to 5 are repeated.

7.3.2 Simulation results

The maximum possible network error correction capacity that a network can provide for the two network error correction methods can be seen in Fig. 7-1.

The black line provides the $min - cut = y = x$ reference. The blue area shows the network error correction capacity for a (n, k) linear code for a network implementing the implicit error correction scheme, where

$$k = min - cut \tag{7-7}$$

and

$$n \geq d_{min} + k - 1. \tag{7-8}$$

The red area shows the network error correction capacity for the same linear code for a network implementing a concatenated error correction scheme, where

$$n = min - cut \tag{7-9}$$

and

$$k \leq n - d_{min} + 1. \tag{7-10}$$

From this graph we can determine the largest (n, k) linear code whose transmission and error correction can be supported by a network with a specific *min-cut*.

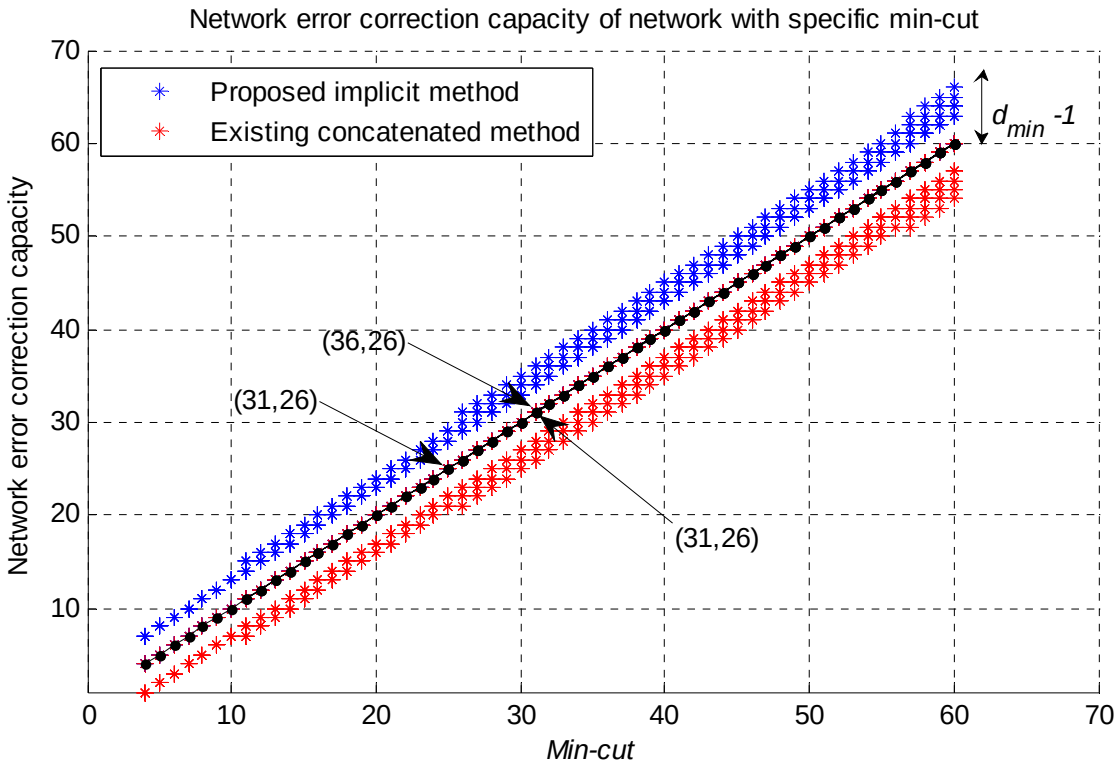


Figure 7-1: Network error correction capacity network for specific *min-cut*

For example: Assume that we have a network where $min - cut = 31$. When the concatenated network error correction scheme is used, the largest linear code that can be supported by the network is a (31,26) linear code. For the implicit error correction method, the network has the network error correction capacity to support the transmission of a(36,31) linear code.

More importantly, we can determine the *min-cut* requirement of a network by looking at the specific (n, k) linear code that must be transmitted over the network.

For example: Assume that we want to transmit a (31,26) linear code over the network. When the concatenated network error correction scheme is used, the network requirement regarding *min-cut* must be $min - cut \geq 31$ to support the code transmission. When using the implicit method, the *min-cut* requirement of the network must be only $min - cut \geq 26$.

7.3.3 Network *min-cut* requirement and characteristic

The *min-cut* requirement for the transmission of a (n, k) linear code, with minimum distance, $d_{min} - 1 \leq l,\ l > 2$, in a network implementing the implicit error correction method is:

$$min - cut \geq k, \qquad k = 2^l - l - 1. \tag{7-11}$$

For a concatenated error correction method, the *min-cut* requirement for the transmission of the same (n, k) linear code is:

$$min - cut \geq n, \qquad n = 2^l - 1. \tag{7-12}$$

The size of (n, k) linear code that can be implemented in the network where $min-cut=k$ for the implicit error correction method is:

$$n \geq d_{min} + k - 1. \tag{7-13}$$

For a concatenated error correction method, the size of (n, k) linear code that can be implemented in the network where $min - cut = n$ is:

$$k \leq n - d_{min} + 1. \tag{7-14}$$

7.4 Network requirements and BER performance

In this section, we assess the BER performance of the two network error correction methods and the case without error correction in a random network coding network. These simulations will enable us to determine further requirements regarding the network for the implementation of the implicit error correction scheme. The BER performance results will also enable us to determine if the implicit error correction is able to successfully reduce the effect of errors in a random network coding environment.

In order to obtain the network requirements for the implicit error correction method, we evaluate the BER performance of this method in networks where the existing schemes can successfully be implemented. The following network parameters are measured:

1. Number of nodes in the network, r,
2. Bit Error Rate of the method (BER)
3. Link error probability of the network (P_{le})

Note that the network size parameter differs for every (n, k) linear block code implemented in the network, as can be deduced from the previous evaluation. For the calculation of this network size requirement, we only evaluate a network that supports the transmission of a (n, k) linear block code where $n = 7$ and $k = 4$.

7.4.1 Simulation setup

The random network coding synchronous network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is constructed as described in Section 4.4. As before, 100 000 simulation iterations are run for each chosen network parameter set for all three methods. Three parallel simulations are run to determine the BER performance of the implicit method, concatenated method and no error correction method for the specified network parameters.

Method with no implemented error correction

The simulation for the case without error correction in the following steps:

1. A block of k information packets, $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ (symbols in $GF\ (2^m)$), is generated at the source, $S \in \mathcal{E}$. These information packets contain information symbols as well as a global encoding vector.

2. The source node transmits the k information packets into the network.

3. The information symbols are encoded within the network.

4. The receiver node, $t \in \mathcal{E}$, collects k channel packets, $y_{data} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_k$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{K=1}^{k} \boldsymbol{\alpha}_{iK} \boldsymbol{x}_K \ , i = 1,2, \dots, k \ . \tag{7-15}$$

5. The receiver decodes the source information by solving the linear system

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_{11}\boldsymbol{x_1} + \alpha_{12}\boldsymbol{x_2} + \cdots + \alpha_{1k}\boldsymbol{x_k} \\ \boldsymbol{y_2} &= \alpha_{21}\boldsymbol{x_1} + \alpha_{22}\boldsymbol{x_2} + \cdots + \alpha_{2k}\boldsymbol{x_k} \\ &\vdots \\ \boldsymbol{y_k} &= \alpha_{k1}\boldsymbol{x_1} + \alpha_{k2}\boldsymbol{x_2} + \cdots + \alpha_{kk}\boldsymbol{x_k} \end{aligned} \tag{7-16}$$

6. The BER is calculated.

Concatenated error correction method

The simulation for the concatenated error correction method in the following steps:

1. A block of k information packets, $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ (symbols in $GF\ (2^m)$), is generated at the source, $S \in \mathcal{V}$. These information packets contain information symbols as well as a global encoding vector.

2. The source node encodes the k information packets into a codeword vector, $\boldsymbol{r}$, consisting of n coded packets.

3. The source node transmits the n coded packets in to the network.

4. The information symbols are encoded within the network.

5. The receiver node, $t \in \mathcal{E}$, collects n channel packets to form the codeword, $\boldsymbol{r}' = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_k, \boldsymbol{y}_{1+k}, \boldsymbol{y}_{2+k}, \dots, \boldsymbol{y}_n\}$.

6. The receiver applies error correction to the codeword, $\boldsymbol{r}'$, and decodes the source information.

7. The BER is calculated.

Implicit error correction method

The simulation for the implicit error correction method in the following steps:

1. A block of input data, $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ (symbols in $GF\ (2^m)$), is generated at the source. These information packets contain information symbols as well as a global encoding vector.

2. The single source node, $S \in \mathcal{V}$, transmits the k information packets in to the network.

3. The information symbols are encoded within the network.

4. The receiver collects k channel packets, $y_{data} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_k$, with linearly independent global encoding vectors, where

$$y_{data} = \sum_{K=1}^{k} \boldsymbol{\alpha}_{iK} \boldsymbol{x}_K \,, i = 1,2, \dots, k \tag{7-17}$$

5. The receiver decodes the source packets and forms the following matrix:

$$\begin{array}{c|cccc} \boldsymbol{x_1} & 1 & 0 & \dots & 0 \\ \boldsymbol{x_2} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \boldsymbol{x_k} & 0 & 0 & 0 & 1 \end{array} \tag{7-18}$$

6. The receiver node, $t \in \mathcal{E}$, collects an additional $(n-k)$ channel packets, $y_{parity} = \boldsymbol{y}_{k+1}, \dots, \boldsymbol{y}_n$, with linearly independent global encoding vectors that that will act as the parity packets.

$$y_{parity} = \sum_{\eta=k+1}^{n} \boldsymbol{\alpha_{i\eta}} \boldsymbol{x'}_{\eta} \,, i = (k+1), \dots, n \tag{7-19}$$

7. These parity packets, along with the decoded data packets are used to form the Generator matrix, G, where

$$G = \begin{bmatrix} 1 & 0 & \dots & 0 & \alpha_{(1+k),1} & \cdots & \alpha_{(1+k),n} \\ 0 & 1 & \dots & 0 & \alpha_{(2+k),1} & \cdots & \alpha_{(2+k),n} \\ \vdots & \vdots & \ddots & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix} \quad (7\text{-}20)$$

where $\alpha_{i,j} \in \mathbb{F}_2$.

8. The Generator matrix, G, is used as parity symbols and the codeword vector, $\boldsymbol{r}'$, is formed where $\boldsymbol{r}' = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k, \boldsymbol{y}_{1+k}, \boldsymbol{y}_{2+k}, \dots, \boldsymbol{y}_n\}$.

9. The receiver applies error correction to the codeword, $\boldsymbol{r}'$, and decodes the source information.

10. The BER is calculated.

7.4.2 Selection of network parameters

The following network parameters are chosen for the BER performance evaluation of both network error correction methods and the no error correction method in a random network coding network.

Forward error correction code

The (n, k) RS code is with a symbol length of m in a Galois field of $GF = 2^m$.

Information packets

The information packets contain symbols of length m from a finite field $\mathbb{F}_q$, in $GF\ (2^m)$, but the number of packets differ:

1. *Implicit error correction method*: k information packets are transmitted over the network, and n channel packets collected.
2. *Existing network error correction method*: n coded packets are transmitted over the channel and n channel packets collected.
3. *No error correction method*: k information packets are transmitted over the network and k channel packets collected.

We assume that the global encoding vector is sent along with ***x*** in its header, but is seen as negligibly small if the packet size is sufficient large.

Network min-cut

The network, $\mathcal{G}$, contains a subset, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| \geq n$ to produce a $min - cut \geq n > k$ and least n edge-disjoint paths between the source and receiver. The *min-cut* of the network cannot be $min - cut \geq k$, because the network must be able to support the implementation of all three methods successfully. Therefore, the network *min-cut* value will be $min - cut \geq n > k$.

The network, $\mathcal{G}$, must also contain another subset of edges $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n > k$ connected to the receiver, $t \in \mathcal{V}$.

Network diameter

The network diameter will be approximately $D \geq 2\frac{\log(n-1)}{\log\sqrt{n}}$, and the average distance $\frac{rD}{2(r-1)} \leq D_{ave} \leq D$.

Network connectivity

The connections in the network will be randomly generated with a connectivity probability $p' = 0.5$.

7.4.3 Simulation results

The 3D scatter plot in Fig. 7-2 illustrates the BER performance of the different methods in a network of varying network size r and link error probability, P_{le}.

From the scatter plot, it can be seen that that better BER results are obtained by the implicit error correction scheme at network size $r \geq 17$ and link error probability $P_{le} \leq 2 \times 10^{-4}$.

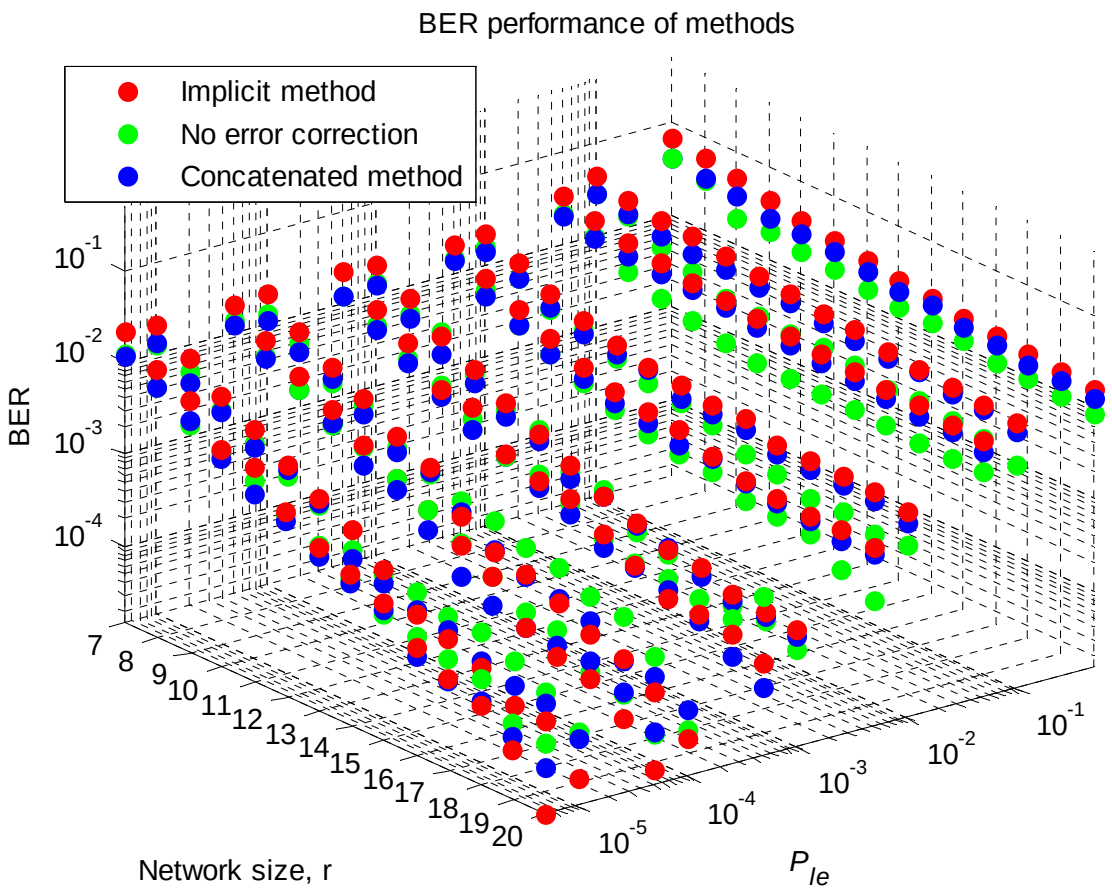


Figure 7-2: BER performance of network error correction schemes

The results represented in this scatter plot can be better evaluated by 2D plots that can be seen in Fig. 7-3 and Fig. 7-4. This scatter plot shows the BER performance of the three methods in terms of P_{le} and network size r.

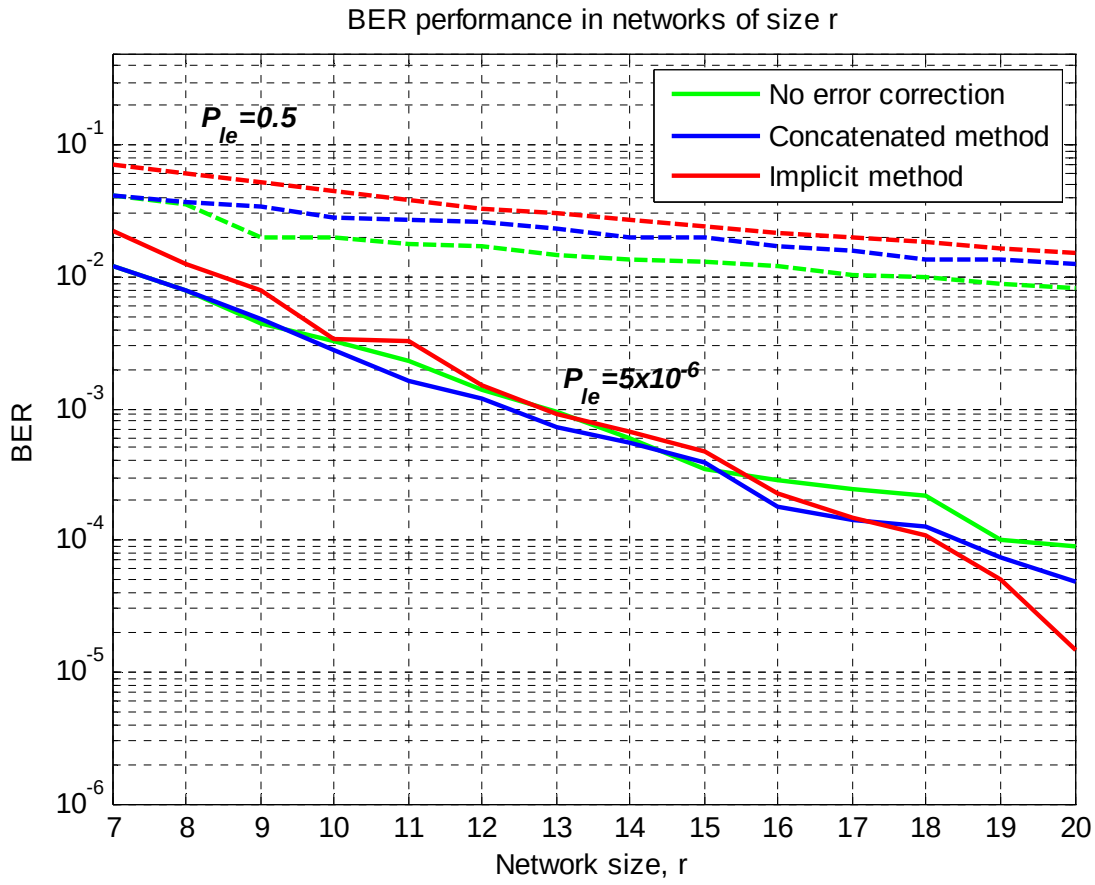


Figure 7-3: BER performance in networks of size r

Fig. 7-3 plots the BER performance of these methods in terms of network size r. The dotted lines represent the their performances at link error probability $P_{le} = 0.5$ and the solid lines at link error probability $P_{le} = 5 \times 10^{-6}$. It can be seen that for networks of size $r \geq 17$, the implicit error correction method offers an improved BER of 1.5×10^{-5} compared with the concatenated method BER of 4.9×10^{-5}.

These results are summarized in Table 7-2.

Table 7-2: BER performance in networks

Network size, r	**BER: Concatenated method**		**BER: Implicit method**		**BER: No error correction**	
	$P_{le} = 0.5$	$P_{le} = 5 \cdot 10^{-5}$	$P_{le} = 0.5$	$P_{le} = 5 \cdot 10^{-5}$	$P_{le} = 0.5$	$P_{le} = 5 \cdot 10^{-5}$
7	0.0412	1.2100×10^{-2}	0.0705	2.2100×10^{-2}	0.0412	1.2200×10^{-2}
10	0.028	2.8000×10^{-3}	0.0449	3.4000×10^{-3}	0.0198	3.3000×10^{-3}
15	0.0201	3.8772×10^{-4}	0.0238	4.7210×10^{-4}	0.0131	3.4777×10^{-4}
17	0.0160	1.4354×10^{-4}	0.0196	1.4588×10^{-4}	0.0102	2.4293×10^{-4}
20	0.0126	4.8485×10^{-5}	0.0152	1.4679×10^{-5}	0.0083	8.9648×10^{-5}

Fig. 7-4 plots the BER performances of the three methods in terms of link error probability, P_{le}. The dotted lines represent their performances at minimum network size $r = 7$ and the solid lines at large network sizes, $r \geq 17$. In Fig. 7-4 it can be seen that when the link error probability is less than $P_{le} \leq 2 \times 10^{-4}$, the proposed method renders better BER performance than the concatenated scheme for network sizes of $r \geq 17$.

The better BER performance of the implicit error correction scheme at network size $r \geq 17$ and link error probability $P_{le} \leq 2 \times 10^{-4}$, as seen in the scatter plot in Fig. 7-2 as well as in Fig. 7-3 and Fig. 7-4, is due to the fact that the network becomes large enough to generate enough channel packets so that the receiver node, $t \in \mathcal{V}$, can construct a valid parity packet set for the implementation of error correction. When the link error probability is $P_{le} \leq 2 \times 10^{-4}$, the error correction capability of the implicit error correction scheme is strong enough to correct most of the errors that occur.

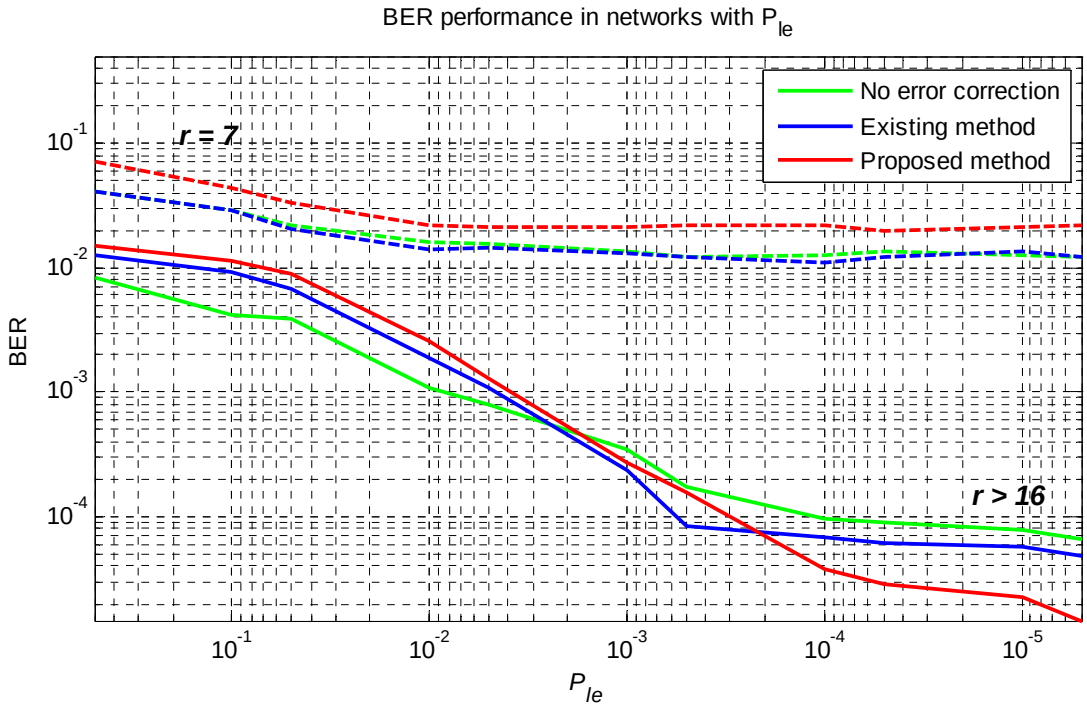


Figure 7-4: BER performance in networks with P_{le}

Based on a random network construction, as the network becomes larger, the number of edges in the subset $\mathcal{H}' \subset \mathcal{E}$ connected to the receiver node increases. By increasing the number of channel packets collected by the receiver, an erroneous packet has a smaller chance of being selected by the receiver to form part of the codeword vector, $\boldsymbol{r}'$. As the link error probability decreases, the probability of the receiver obtaining an erroneous packet also decreases.

7.4.4 Network requirements

From the BER performance analysis done, we see that the implicit error correction method delivers a better BER performance than the concatenated error correction scheme when the network has certain characteristics. The following two network requirements are necessary, but not sufficient, for the (n, k) implicit error correction method to render favourable results in a random network coding network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $n = 7$ and $k = 4$.

- Network size, r:

$$r \geq 17 \tag{7-21}$$

- Link error probability, P_{le}:

$$P_{le} \leq 2 \times 10^{-4} \tag{7-22}$$

7.4.5 Operational validation

From the obtained BER performances shown in Figures 7-2 to 7-4, we are able to determine that the implicit method's output is sufficient for its intended purpose.

The better BER performance of the implicit error correction method under certain network conditions proves that this method is able to:

- reduce the effect of errors in a random network coding network, by
- implementing successful error correction in the random network coding scenario
- without applying redundancy at the source.

It can clearly be seen that this implicit error correction method complies with the initial specified requirements and therefore we can validate this implicit error correction method.

7.5 Conclusion

This chapter presents a series of comparisons done on the implicit error correction method, concatenated error correction method and method without error correction. These comparisons make it possible for us to determine the network requirements for and characteristics of the implementation of the implicit error correction method in a network. When these requirements are known, we are able to analyse a specific network to determine if the implementation of the proposed method will render better results than the implementation

of a concatenated network error correction scheme. We are also able to determine certain network characteristics associated with the implementation.

The requirements determined in this chapter, as well as other requirements determined earlier in this dissertation, are summarised in Table 7-3.

Table 7-3: Requirements for implicit error correction method

Requirement	
Network synchronisation	non-cyclic, synchronous
Network *min-cut*	$\mathcal{H} \subset \mathcal{E}, \lvert\mathcal{H}\rvert = min - cut \geq k$
Network size, r	$r \geq 17$
Link error probability, P_{le}	$P_{le} \leq 2 \times 10^{-4}$
Receiver connections	$\mathcal{H}' \subset \mathcal{E}, \lvert\mathcal{H}\rvert \geq n$ connected $t \in \mathcal{V}$

The characteristics associated with the implementation of the implicit error correction method in a random network coding network, is summarised in Table 7-4.

Table 7-4: Characteristics of implicit error correction method

Characteristic	
Computational complexity at source, $S \in \mathcal{V}$	The same as for a network where no error correction is implemented.
Computational complexity at receiver, $r \in \mathcal{V}$	The execution of the additional $LINEARLY\ INDEPENDENT\ SETS\ SELECTION$ process leads to higher computational complexity than for concatenated schemes.
Computational time	High due to high computational complexity and waiting by

	receiver to collect sufficient channel packets.
$(\boldsymbol{n}, \boldsymbol{k})$ **linear code supported**	$k \leq min - cut$ $n \geq d_{min} + k - 1$

Chapter 8:

Conclusion

In this chapter, the addressed research problem is analysed and presented, along with relevant literature on the problem. We discuss the methodology followed to achieve the main objectives and give an overview of the literature presented. We summarise the results obtained in this dissertation which include network requirements, method characteristics and performance. Finally, we discuss further work that may exist in the continuation of this field and publications that resulted from this study.

8.1. Summary of contribution

In this dissertation, we developed a method that uses the redundant information implicitly generated inside a random network coding network to apply error correction to the transmitted message. The obtained results show that the developed implicit error correcting method can counter the effect of errors in a network without adding redundancy at the source. This method presents numerous advantages compared to the existing concatenated error correction methods.

In order to quantify those advantages, we compared the proposed method with the concatenated network error correction method. These comparisons allowed us to determine the requirements and associated characteristics of a random network coding network for the implicit error correction method to render better results than existing concatenated schemes.

When these requirements are known, we are able to analyse a network and determine if the implementation of the proposed method will perform better than the implementation of a concatenated network error correction scheme. We are also able to determine certain network characteristics associated with the specific implementation.

8.2. Achievement of objectives

The research performed in this dissertation aims to make a contribution in the field of network error correction. The research question posed in this dissertation was the following:
How can we reduce the effect of errors in random network coding by using the redundancy generated by a random network coding network?

The answer was obtained by completing the following objectives:

- Developing and constructing an error correction method that does not apply redundancy at the source.
- Verifying and validating the designed method.
- Determining the network requirements for the implementation of this method

- Commenting on the characteristics of this method.

By evaluating the work performed in this dissertation, it is clear that these objectives were achieved.

8.2.1 Developing and constructing an error correction method that does not apply redundancy at the source

By studying the fields of network coding, random network coding and forward error correction, we determined that there existed no documented network error correction method that addressed the problem of reducing the effects of errors in random network coding without applying redundancy at the source nodes.

We determined that random network coding environments have the tendency to generate redundancy inside the network which is normally discarded by the receivers. By exploiting this implicitly generated redundancy, we were able to construct a method that does not apply redundancy for error correction at the source.

We studied the literature further to assist us in designing this implicit error correction method and to determine which network parameters may possibly have an influence on this problem. These network parameters were considered and a conceptual model was designed based on the mathematical concepts in the literature.

From the conceptual model, a basic working model was developed to determine if the model was functional and to provide a basis for us to see how the model responded to the different network parameters. Experiments were conducted to determine the influence of the network parameters on the method. These results were used to revise and improve the working model so that better results could be obtained. Finally, a network parameter set was determined where the proposed method rendered satisfactory results.

In the final iteration implicit error correcting method was evaluated and we determined that the method presented numerous advantages compared with existing concatenated error correction methods.

One of the most significant advantages presented by the implicit error correction method is the fact that the source does not implement error correction. The receiver can implement independent decoding where it applies error correction by simply collecting channel packets selectively without having to communicate with the source regarding the selected FEC code. The receiver can apply an error correction code (example: Hamming, LDPC, Reed Solomon etc.) and can solely base its decision on the channel packets received, the network characteristics, required decoding time or the computational complexity.

The biggest constraint of this independent decoding is the size and connectivity of the network, which must act as the network error correction encoder. The size of the network must be chosen so that the receiver has a high probability of receiving a valid set of parity packets. A valid set of parity symbols will ensure that the receiver will be able to apply an error correction code and decode the transmitted data. If the receiver cannot collect a valid set of parity symbols, the implicit encoding capabilities of the network will have no use. The implicit error correction scheme also requires the network to be synchronised and non-cyclic.

8.2.2 Verifying and validating the designed method

We validated the conceptual model of the implicit error correction method by following the procedure represented in [17]. We showed that the conceptual design of the method is correctly based on the documented theory.

We verified the computerised model of the implicit error correction scheme by applying the static-testing procedure (structured walk-through) to the simulation code. We showed that the steps followed in the implementation of the computer code are mapped directly from the conceptual design of the implicit error correction method.

Finally, we determined that the implicit error correction scheme is able to render better BER results under certain network conditions than certain documented concatenated network error correction schemes. These results enabled us to validate the operation of the method, because it fulfils the initial requirements of reducing the effect of errors in random network coding without adding redundancy at the source.

8.2.3 Determining the network requirements for the implementation of this method

In order to see where the implicit error correction method could be used as an alternative to existing network error correction schemes to render better results, we determined the network requirements for this method. We determined these requirements by comparing the proposed method with the existing concatenated network error correction method.

For a (n, k) linear block code, we found that the network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, must contain a subset of edges, $\mathcal{H} \subset \mathcal{E}$ of size $|\mathcal{H}| \geq k$ to produce a $min - cut \geq k$. This *min-cut* requirement will be able to provide the network error correction capacity for successful error correction by the proposed method. The network must also contain a subset of edges $\mathcal{H}' \subset \mathcal{E}$ of size $|\mathcal{H}'| \geq n > k$ connected to the receiver to support the collection of at least n channel packets by the receiver.

Through the BER performance testing, we found that a large enough network could implicitly generate sufficient channel packets so that the receiver node could construct a valid parity packet set for successful error correction. A large enough network also has more edges connected to the receiver which decreases the chance of erroneous channel packets being selected by the receiver. As the link error probability decreases, the probability of the receiver obtaining an erroneous packet also decreases which improves the method's BER performance.

8.2.4 Commenting on the characteristics of the method

We established that the computational complexity of the implicit error correction method at the network source was the same as for a network where no error correction was implemented, but significantly higher at the receiver side of the network. This was due to the additional algorithm we executed at the receiver to obtain multiple sets of linearly independent channel packets for error correction.

Due to this high computational complexity characteristic at the receiver, a longer computational time was needed for the implementation of the proposed method. We saw that

the computational time of this method was increased further by the fact that the receiver had to wait longer in order to obtain a large enough set of channel packets from the network to construct two sets of linearly independent channel packets.

These results implies that when the implicit error correction method is implemented in a random network coding network, the computational complexity and computational time characteristics are higher than for a concatenated method.

8.3. Evaluation of our method

In this dissertation, we found that an error correction scheme can be implemented without adding redundancy at the source nodes. The decoding ability of this implicit error correction method is dependent on the characteristics of the network. We found that large networks with a high level of interconnectivity yield more redundant information allowing more advanced error correction schemes to be implemented.

Random network coding networks are prone to error propagation and we presented that our scheme outperforms concatenated error correction schemes for low link error probability.

8.4. Future Work

Future work may include the following:

1. The development of an adaptive decoding receiver node: This will enable a receiver node of a network to automatically choose between different decoding algorithms depending on certain network constraints that may include:
    - the contents of the channel packets received
    - the provided time for decoding
    - the network size, connectivity and *min-cut.*

2. The exclusion of packet headers: By using different error correction codes, we may be able to determine the contents of received channel packets without the need for packet headers. This will lead to a saving in network throughput, because less information will be transmitted over the network.

3. The implementation of a similar error correction method in non-synchronous networks or networks that may include cyclic components. This will make the error correction scheme more robust and adaptable to different network scenarios.

8.5. Conference contributions from this dissertation

The following conference contributions resulted from the research as performed in this dissertation. These contributions are available in Appendix A.

8.5.1 Engineering

Southern African Telecommunication Networks and Applications Conference (SATNAC)
Wild Coast Sun, South Africa
September 2008
Full Paper acceptance: ***Hamming Error Correction techniques for the Improvement of Robustness in networks using Random Network Coding.***

Southern African Telecommunication Networks and Applications Conference (SATNAC)
Royal Swazi Spa, Swaziland
September 2009
Full Paper acceptance: ***Error Correction with the Implicit Encoding Capability of Random Network Coding***

First International Conference on Ad Hoc Networks
Niagara Falls, Ontario, Canada
September 2008

Full Paper acceptance: ***Error Correction using Implicit Encoding Capabilities of Random Network Coding***

8.5.2 Information Technology

South African Institute of Computer Scientists and Information Technologists (SAICSIT) Conference
Riverside Hotel and Conference Centre, South Africa
October 2009
Main conference - Poster acceptance: ***Performance of the Implicit Error Correction in Network Coding in the presence of Link Errors***

South African Institute of Computer Scientists and Information Technologists (SAICSIT) Conference
Riverside Hotel and Conference Centre, South Africa
October 2009
Master's and Doctoral Research Symposium – Full paper acceptance: ***Performance of Implicit Error Correction of Network Coding networks using Random Network Coding***
Received award for the Best Paper at Research Symposium

References

[1] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network Information flow," *IEEE trans. on Information Theory, vol. 46*, pp. 1204-1216, 2000.

[2] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE Int. Symp. Information Theory*, Yokohama, July 2003, p. 442.

[3] D. Silva, F. R. Kschischang, and R. Koetter, "A Rank-Metric approach to error control in random network coding," in: *Proc. Information Theory for Wireless Networks, 2007 IEEE Information Theory Workshop*, Solstrand, Norway, July 2007, p. 1-5

[4] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2003.

[5] M. Médard and R. Koetter, "Beyond routing: An algebraic approach to network coding," *IEEE/ACM Transaction on Networking, vol. 11*, pp. 782-796, October 2003.

[6] R. W. Yeung and N. Cai, "Network coding, algebraic coding and network error correction," in *Proceedings of Information Theory and Applications Workshop*, San Diego, CA, February 2006, pp. 119-122.

[7] N. Cai and R. W. Yeung, "Network coding and error correction," in *Proceedings of IEEE Information Theory Workshop*, October 2002, pp. 119-122.

[8] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev., 36(1)*, pp. 63-68, 2006.

[9] S. Lin and D. J. Costello, *Error control coding: Fundamentals and applications*. Englewood Cliffs, N.J: Prentice-Hall, 1983.

[10] G. C. Clark Jr and J. B. Cain, *Error-correction coding for digital communications*. New York: Plenum Press, 1981.

[11] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding, "In: *Proc. IEEE Transactions on Information Theory, Volume 54, Issue 8*, August 2008, p. 3579 - 3591

[12] S. Jaggi, et al., "Resilient network coding in the presence of Byzantine adversaries," in *Proceedings of 26th IEEE Int Conf. on Computer Commun*, Anchorage, AK, May 2007, pp. 616-624.

[13] D. Silva and F. R. Kschischang, "On Metrics for Error Correction in Network Coding," *IEEE trans. on Information Theory*, 2008.

[14] G. C. Vining, *Statistical Methods for Engineers*. Pacific Grove, CA: Dexbury Press, 1998.

[15] IEEE, *IEEE Standard Dictionary of Elecrtical and Electronics terms*. New York, 1984.

[16] IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. New York, 1991.

[17] R. G. Sargent, "Verification and Validation of Simulation Models," Department of Electrical Engineering and Computer Science, Syracuse University, *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[18] W. L. Oberkampf, T. G. Trucano, and C. Hirsch, "Verification, Validation and Predictive Capability in Computational Engineering and Physics," in *Foundations for Verification and Validation in the 21st Century Workshop*, Johns Hopkins University, Laurel, Maryland, October 2002.

[19] Arab Urban Development Institute, "Verification and Validation," in *Systems Analysis and Design*, Oct 2003.

[20] N. Pappas, "Network Coding," Institute for Computer Science (ICS) , September 2007.

[21] A. H. Dekker and B. D. Colbert, "Network Robustness and Graph Theory," in *27th Australasian Computer Science Conference*, in Conferences in Research and Practice in Information Technology, Vol. 26, V. Estivill-Castro, Ed. 2004

[22] B. Bollobas, *Algebraic Graph Theory*. Cambridge: Cambridge University Press, 2001.

[23] C. Fragouli and E. Soljanin, "Nework coding fundamentals," *Foundations and trends in networking*, vol. 2, no. 1, 2007.

[24] T. Ho, "Networking from a Network Coding perspective," PhD Thesis, Massachusetts Institute of Technology, Dept of EECS, May 2004.

[25] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Medard, and N. Ratnakar, "Network coding for wireless applications: A brief tutorial," in *International Workshop on Wireless and Ad-hoc Networks (IWWAN)*, May 2005.

[26] D. J. C. MacKay, "Information theory, inference and learning algorithms," *Cambridge University Press*, 2003.

[27] T. Noguchi, T. Matsuda, and M. Yamamoto, "Performance Evaluation of New Multicast Architecture with Network Coding," *IEICE Trans. on Communications*, vol. E86-B, pp. 1788-1795, June,

[28] Y. Wu, P. Chou, and K. Jain, "A comparison of Network Coding and Tree Packing," in *IEEE International Symposium of Information Theory (ISIT 2004)*, Chicago, June 27-July 2 2004.

[29] T. Ho, M. Médard, and R. Koetter, "An Information- theoretic view of Network Management," *IEEE Trans on Information Theory*, vol. 51, no. 4, pp. 1295-1312, April 2005.

[30] B. Cohen, "Incentives build robustness in BitTorrent," in: Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June, 2003.

[31] N. Cai and R. W. Yeung, "Network error correction, part II: lower bounds," *Communications in Information Systems*, vol. 6, no. 1, pp. 37-54, 2006.

[32] L. Song, R. W. Yeung, and N. Cai, "A seperation theorem for single source network coding," *IEEE Trans. on Information Theory*, vol. 52, no. 5, pp. 1861-1871, 2006.

[33] R. W. Yeung, S. Y. Li, N. Cai, and Z. Zhang, *Network Coding Theory*. Now Publishers, 2005.

[34] D. G. Hoffman and D. A. Leonard, *Coding Theory: The Essentials*. Marcel Dekker Inc., 1991.

[35] T. K. Moon, *Error Correction Coding: Methematical Methods and Algorithms*. Utah State University: John Wiley & Sons Inc., 2005.

[36] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. on Information Theory, vol 49*, p. 371, Feb 2003.

[37] D. Silva and F. Kschischang, "Using Rank-Metric Codes for Error Correction in Random Network Coding," in *IEEE International Symposium on Information Theory*, June 24 – 26 2007.

[38] C. Langton. (2001) www.complextoreal.com. [Online]. http://www.complextoreal.com, accessed on 26 May 2008.

[39] D. Whitely, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizating connections and connectivity," in *Parallel Computing, vol 14*, pp. 347-361, 1990.

[40] The Mathworks, Inc. (2009) Matlab Central. [Online]. http://www.mathworks.com/matlabcentral/newsreader/, accessed on 10 November 2009.

[41] Wolfram Research, Inc. (2009) Wolfram Mathematica Documentation centre. [Online]. http://reference.wolfram.com/mathematica/ref/QRDecomposition.html?q=QRDecomposition&lang=en, accessed on 10 November 2009.

[42] M. Misra, H. B. T. I. Kanpur, and R. Moona, "Design of Systolic arrays for QR Decomposition," *in: Proc. International Conference on Computer System and Education*, Bangalore, June 22-25, pp. 247-255, 1994.

[43] H. Qi and N. Goertz, "Low-Complexity Encoding of LDPC Codes: A New Algorithm and its Performance," School for Engineering and Electronics, The University of Edinburgh Institute for Digital Communications, Joint Research Institute for Signal and Image processing , 2004.

[44] R. Bell. (2009) A Beginners' Guide to Big O Notation. [Online]. http://robbell.net/2009/06/a-beginners-guide-to-big-o-notation/, accessed on 9 November 2009.

[45] S. Jaggi, Y. Cassuto, and M. Effros, "Low Complexity Encoding for Network Codes," in IEEE International Symposium on Information Theory (ISIT '06), Seattle, WA, July 2006, pp. 40-44.

[46] F. Swartz. (2005) Java Notes: Algorithms: Big- Oh Notation. [Online]. http://leepoint.net/notes-java/index.html, accessed on 9 November 2009.

Appendix A:

Conference contributions from this dissertation

SATNAC (Southern Africa Telecommunication Networks and Applications Conference)

Wild Coast Sun, South Africa

September 2008

Full Paper acceptance: *Hamming Error Correction techniques for the Improvement of Robustness in networks using Random Network Coding.*

SATNAC (Southern Africa Telecommunication Networks and Applications Conference)

Royal Swazi Spa, Swaziland

September 2009

Full Paper acceptance: *Error Correction with the Implicit Encoding Capability of Random Network Coding*

First International Conference on Ad Hoc Networks

Niagara Falls, Ontario, Canada

September 2008

Full Paper acceptance: *Error Correction using Implicit Encoding Capabilities of Random Network Coding*

SAICSIT (South African Institute for Computer Scientists and Information Technologists) Conference

Riverside Hotel and Conference Centre, South Africa

October 2009

Masters and Doctoral Research Symposium - Paper acceptance: *Performance of Implicit Error Correction of Network Coding networks using Random Network Coding*

Received Best Paper award for the Masters and Doctoral Research Symposium

SAICSIT (South African Institute for Computer Scientists and Information Technologists) Conference

Riverside Hotel and Conference Centre, South Africa

October 2009

Main conference - Poster acceptance: *Performance of the Implicit Error Correction in Network Coding in the presence of Link Errors*

Hamming error correction techniques for the improvement of robustness in networks using random network coding

S von Solms, ASJ Helberg
School for Electric, Electronic and Computer Engineering
North West University, Potchefstroom Campus
Tel: (081) 299 1961, Fax: (081) 299 1977, E- mail: 12987611@nwu.ac.za

Abstract – In this paper, we introduce an algorithm for error detection and correction in Random Network Coding. The introduced technique exploits the encoding characteristics of random network coding and uses the well known Hamming Code as a decoding algorithm. For a network where random network coding is applied, this technique can be a useful error detecting and correcting method that will improve the network's robustness.

Index Terms— Error Correction, Hamming Code Network Coding, Random Network Coding, Robustness.

I. INTRODUCTION

The concept of network coding was first introduced by Ahlswede, Cai, Li and Yeung in 2000 [1]. Instead of simply forwarding data in a network, as in traditional routing, they proposed that nodes may recombine several input packets into one or more output packets by performing a logical x-or operation on it. The concept of Random Network Coding was introduced by Ho, Koetter, Médard, Karger, and Effros in [2].

Their approach provides an improvement in robustness [11] where the success of information reception does not depend on receiving packets that contain specific information, but on receiving enough independent packets [12].

Random Network Coding, described in [2] works as follows: All the nodes in the network, except the receiver node, perform independent random linear mappings of their inputs. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are made from all inputs to the node, as shown Figure 1. The outputs are chosen independently and randomly and must be non- zero.

The receiver node of the network then obtains a series of independent linear combinations which it can use to decode the transmitted data. The receiver node of the network only needs to know the overall linear combination of the source processes in each of the incoming packets. This information is provided by a coding vector that is included in each message overhead [14].

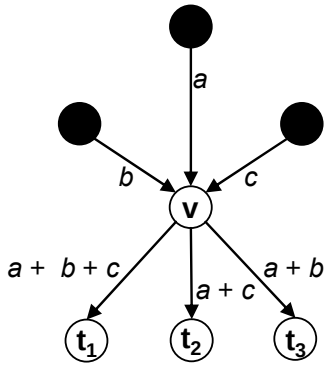


Figure 1. Random network coding subnetwork

The disadvantage of this characteristic is the fact that the network can be very sensitive to errors [4, 5]. A single error packet has the potential to infect the whole network and corrupt all the packets used by the receiver for decoding.

When a corrupt packet is linearly combined with legitimate packets, it can affect all the information gathered in that packet. Another problem that may occur is that an insufficient number of packets containing information of a single source node may reach the receiver, therefore preventing the receiver to decode the correct source messages.

It is possible to address these shortcomings by implementing error correction in the network. An error correction code will be able to correct and detect data packets corrupted due to additive errors.

Yeung and Cai [15] showed that network error correction can be executed in network coding, by using classical coding theory. Their study aims to manage errors that occur in networks by detecting and correcting it. This will enable the receiver of the network to receive the correct information sent over the network.

Koetter et al [10] as well as Silva et al [5, 6] introduced different approaches to error correction in Random Network

Coding. Their different approaches both consider networks here the encoded vector are generated by the source node and sent through the network. As the information vector is sent through the network, the packets may receive an error over any link in the network. These errors can be corrected by implementing the error correcting methods at the receiver end.

It can easily be seen that the topic of error correction in Random Network Coding is very popular. However, the construction of the error correcting codes are in a concatenated form where encoding takes place before transmission. The natural encoding capabilities of Random Network Coding for error correction codes are not considered. This fact opens up great possibilities in the field of network error detection and correction in Random Network Coding.

Several advantages can be achieved by using the natural encoding capabilities of Random Network Coding:

1. *Saving in Bandwidth*: Encoding the information within the network, instead of transmitting the already encoded codeword over the network leads to a saving in network bandwidth [16].
2. *Reduction in Congestion*: This encoding approach reduces the number of transmissions in the network which leads to reduction in network congestion [17].
3. *Higher Throughput:* By reducing the number of transmissions in the network, a higher throughput can be obtained [12, 17].

These factors all contribute to improving the robustness of the network.

II. Error Correction

A well known single bit error correction code is the Hamming Code, invented by Richard Hamming [7]. Hamming Codes have a length n, where $n = 2^m - 1$ and a dimension k, where $k = 2^m - m - 1 \quad (m \geq 2)$ [10].

The (n, k) Hamming Code produces n output bits out of k input bits. This code can detect and correct a single- bit error and detect (but not correct) up to two simultaneous bit errors [8].

The goal of the Hamming Code is to develop a set of logical parity bits so that errors (inverted bits) in data or parity bits can be detected and corrected.

In this paper, we present an algorithm for a network error correcting code based on the Hamming Code using random network coding. We provide an example where the (7, 4) Hamming Code is used.

We focus on multiple unicast networks where only the receiver node applies error correction techniques and the intermediate nodes only create linear combinations of the packets they received. Thus the operations of random network coding are not changed. The network itself acts as an encoder of the source information packets for the error correction to take place.

III. Model

We adopt the notation used in [2, 4] of an acyclic network model. The network is represented by a directed graph $G = (V, E)$. V is the set of nodes in the network and E the set of edges in G which represents the communication channels. $S = \{s_1, s_2, \dots, s_{|S|}\} \in V$ represents the source nodes and $t \in V$ the sink node in the multiple unicast network.

An edge from node a to b is indicated by $(a,b) \in E$. Node a is called the input node of edge (a,b) and edge (a,b) is called the input edge of node b, while node b is called the output node of edge (a,b) and edge (a,b) is called the output edge of node a.

Each edge in the network has unit capacity; therefore it is able to transmit a single unit of information per unit time.

Random Network Coding is implemented where each network node randomly and independently selects coefficients from a finite field F_2. The receiver also receives the overall linear combination of the source processes resulting in each information bit.

Let $\boldsymbol{X}(v) = \{X(v,1), X(v,2), \dots, X(v, \mu(v))\}$ be a collection of $\mu(v)$ random linear combinations received by node $v \in V$. These linear combinations consist of the processes sent by the source nodes. We want to create another non- zero random linear combination, $\boldsymbol{Z}(v)$, out of a subset of the random linear combinations $\boldsymbol{X}(v)$ and forward it to some different node $v' \in V$. This concept can be illustrated in Figure 2. This pattern is repeated until this information bit reaches the receiver.

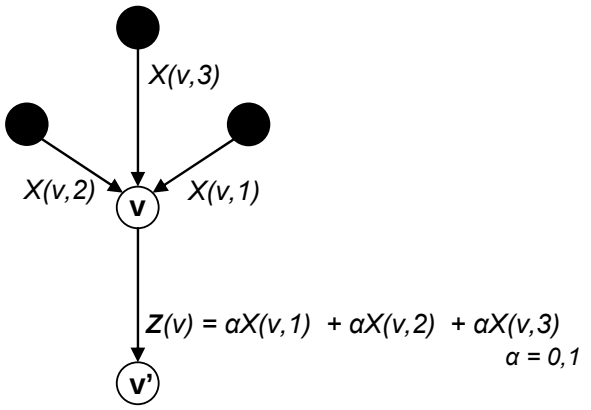


Figure 2. Random linear combinations in Random Network Coding

In this paper, a network is considered where the intermediate nodes do not correct errors, but only create the linear combinations of their inputs.

We assume that an error always occurs on an edge. When an error occurs on edge (a,b) the symbol sent by node a differs from the symbol received by node b. Only the receiver node $t \in V$ will be able to detect and correct errors.

For the implementation of the Hamming Code on Random Network Coding, a few specific constraints must be put in place.

1. We consider a finite field F_2, where the information contained in each packet is only a single bit. This means that each node can only send a single bit over an edge in a single time unit.
2. The network source nodes, $S = \{s_1, s_2, \dots, s_k\} \in V$ each contain one data bit $\{d_1, \dots, d_k\}$. The number of source nodes, k, depends on the (n, k) Hamming Code used.
3. No errors occur during the first set of transmissions, meaning that no errors occur on edges (s_i,b), $i = 1, \dots, k$.
4. For this method to be implemented successfully, only networks are considered where at most two errors occur.
5. This random network has a topology where the receiving node, t, receives n or more independent linear equations coming from n or more independent paths.

IV. Hamming Error Correction techniques in Random Network Coding

If we consider a network where no errors occur on the edges, the receiver has a very easy task decoding the information sent from the source nodes. The receiver only needs to find k linear independent polynomials, $\boldsymbol{Y}(r) = \{Y(r,1), Y(r,2), Y(r,3), \dots, Y(r,k)\}$ created in the network to decode the source information correctly.

A Random Network Coding environment is not necessarily error free. To ensure that the receiver still receives the correct information, the Hamming error correction technique is implemented by the receiver when decoding.

The network will act as an encoder, while the receiver will decode using the Hamming Code algorithm for the detection and correction of errors.

A. Encoding

The k source nodes $S = \{s_1, s_2, \dots, s_k\}$ each send one data bit $d = \{d_1, \dots, d_k\}$ into the network. As these data bits flow through the network, linear combinations are formed from them, shown in Figure 3. With the existence of n or more independent paths at the receiver, an error in one of the paths does not influence the correct flow of information in the other paths. The receiver waits until it receives n or more equations, which consist of two sets of linear independent equations of at least size k and $(n - k)$ respectively, $\boldsymbol{Y}(r) = \{Y(r,1), Y(r,2), Y(r,3), \dots, Y(r,n)\}$ containing $\{d_1, \dots, d_k\}$.

Firstly, the k data bits $\{d_1, \dots, d_k\}$ are decoded by the set of k linear independent equations, $\boldsymbol{Y}(r) = \{Y(r,1), Y(r,2), Y(r,3), \dots, Y(r,k)\}$ received.

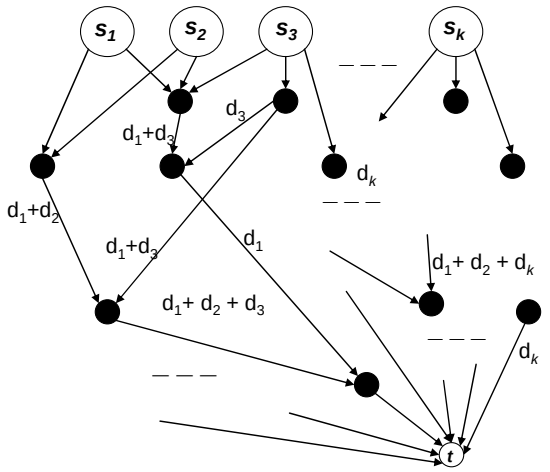


Figure 3. A network using Random Network Coding

The parity bits needed for the Hamming Code consist of the set of l linear independent equations, $\boldsymbol{Y}(r) = \{Y(r,k+1), Y(r,k+2), \dots, Y(r,n)\}$, where $l = n - k$. For the parity bits to provide the platform for error detection and correction, they must logically relate to the data bits**.**

This relationship can be expressed as follows [13]:

$$
\begin{aligned}
p_1 &= z_{11}d_1 + z_{12}d_2 + \dots + z_{1k}d_k \\
p_2 &= z_{21}d_1 + z_{22}d_2 + \dots + z_{2k}d_k \\
&\vdots \\
p_l &= z_{l1}d_1 + z_{l2}d_2 + \dots + z_{lk}d_k \qquad \dots (1)
\end{aligned}
$$

where $z = 0, 1$

Each parity bit must consist of a linear combination of $k - l$ data bits in order to form a legitimate set of parity bits.

These parity and data bits form a specific sequence, called the codeword $\boldsymbol{r}$. This codeword will be used in the process of detecting and correcting errors in the network.

In Hamming Codes, there exists $m = 2^k$ valid codewords which are expressed in the form $\boldsymbol{r} = \{d_1\ d_2\ d_3 \dots d_k\ p_1\ p_2 \dots p_l\}$ [13].

V. EXAMPLE

An example is given to illustrate the concept of error detection and correction techniques through the use of a Hamming Code in random network coding.

1. The Shortened Hamming Code or (7, 4) code will be used.
2. The network contains four source nodes, $\{s_1, s_2, s_3, s_4\} \in V$ each containing one data bit $\{d_1, d_2, d_3, d_4\}$.
3. No errors occur during the first set of transmissions, meaning that no errors occur on edges (s_i, b), $i = 1, \ldots, 4$.
4. This random network has a topology where the receiving node, t, receives two sets consisting of four and three independent linear equations each coming from seven or more independent paths.

Suppose the four data bits are sent from the source nodes over the randomized network. The receiver waits until it receives the two sets of linear independent equations. Because each equation is received from an independent path, a single error in one path does not affect the other equations received from different paths.

The first set of four linear equations is used to decode the four data bits sent from the source nodes $\{d_1, d_2, d_3, d_4\}$. The other set of three equations are used to calculate the parity bits. They are expressed as follows:

$$\begin{aligned} p_1 &= z_{11}d_1 + z_{12}d_2 + z_{13}d_3 + z_{14}d_4 \\ p_2 &= z_{21}d_1 + z_{22}d_2 + z_{23}d_3 + z_{24}d_4 \\ p_3 &= z_{31}d_1 + z_{32}d_2 + z_{33}d_3 + z_{34}d_4 \end{aligned} \quad \ldots (2)$$

The sequence of data and parity bits forms the codeword $\boldsymbol{r} = \{d_1, d_2, d_3, d_4, p_1, p_2, p_3\}$. □

A. Error detection and correction

To check if an error has occurred in the network, a parity check matrix $\boldsymbol{H}$ must be generated. Every (n, k) code has an associated $(n - k)$ x n parity check matrix $\boldsymbol{H}$. This matrix has the property of $\boldsymbol{H}*\boldsymbol{r} = \boldsymbol{0}$, where $\boldsymbol{r}$ is the codeword [10].

This parity matrix $\boldsymbol{H}$ is not unique and must be generated every time a new codeword is formed. The linear equations used for calculating the parity bits provide the information needed to generate $\boldsymbol{H}$.

A $\boldsymbol{Z}$ matrix is defined:

$$\boldsymbol{Z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1k} \\ z_{21} & z_{22} & \cdots & z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ z_{l1} & z_{l2} & \cdots & z_{lk} \end{pmatrix} \quad \ldots (3)$$

The $\boldsymbol{H}$ matrix consists of the $\boldsymbol{Z}$ matrix, as well as the identity matrix $\boldsymbol{I}$ so that $\boldsymbol{H} = (\boldsymbol{Z} \mid \boldsymbol{I})$.

$$\boldsymbol{H} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1k} & 1 & 0 & \ldots & 0 \\ z_{21} & z_{22} & \cdots & z_{2k} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{l1} & z_{l2} & \cdots & z_{lk} & 0 & 0 & \ldots & 1 \end{pmatrix} \quad \ldots (4)$$

All parity check matrices' columns are binary representations of the numbers, though not in order, from 1 through to n.

The receiver must multiply (modulo 2) $\boldsymbol{H}$ and $\boldsymbol{r}$ to obtain a syndrome vector $\boldsymbol{z}$, which indicates if an error has occurred. This vector also indicates which codeword bit has been logically inverted in the network (additive error).

In an error free network, the syndrome vector $\boldsymbol{z} = \boldsymbol{0}$ to indicate that no bit in the codeword is incorrect and that $\boldsymbol{r}_{correct} = \boldsymbol{r}$. In a network where a single error has occurred, the syndrome vector will be equivalent to one of the columns of the $\boldsymbol{H}$ matrix. The error position i corresponds to column i of $\boldsymbol{H}$ and $\boldsymbol{r}_{correct} = \boldsymbol{r} + e_i$ where e_i is a zero vector except for a 1 in position i. [10, 13]

In other words, the bit error can easily be detected by comparing the syndrome vector to the columns of $\boldsymbol{H}$. Once the incorrect bit is determined, the received codeword can be corrected by simply inverting the erroneous bit.

When the correct codeword has been found, the codeword can simply be decoded by using the decoding algorithm of the Hamming Code [10].

B. Multiple bit errors

As shown above, this application of the Hamming Code works effectively for any single bit error correction in random network coding. This Hamming Code application can, however, also be used for single and double bit error detection.

When we multiply the codeword received with the parity check matrix $\boldsymbol{H}$, the syndrome vector $\boldsymbol{v}$, will be non- zero whenever errors have occurred.

This method will only tell us if a single or double bit error has occurred, but cannot assist us in the correction of a double bit error, because we cannot distinguish between single or double bit errors.

VI. MAIN RESULTS

By implementing this error correction method in Random

Network Coding, many advantages are achieved.

a) No increase in bandwidth: To correct a potential error, n codeword bits are needed, but only k ($k < n$) data bits are sent into the network from the source nodes. The other (n - k) bits are generated by the network. By using the network itself as an encoder, the bandwidth usage (information rate) remains the same, although up to double the information can be received. In other words, the network itself provides the redundant information required to perform error correction and detection.

This saving of bandwidth can clearly be seen in the Hamming (7, 4) example in Figure 4 where (*a*) is a network where the network is not used as an encoder and (*b*) where the encoding takes place in the network itself.

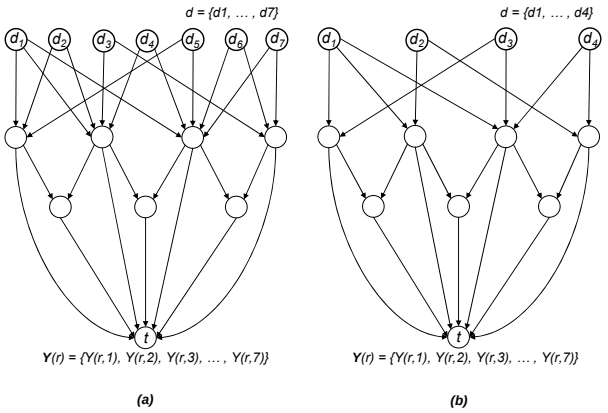


Figure 4. (7, 4) Hamming network (example)

b) Congestion and Throughput: It can also be seen in Figure 4 that this coding method reduces the number of transmissions in the network. This reduces network congestion and accordingly produces a higher throughput [17].

c) Single error detection and correction: The detection and correction of a single error in a network can be ensured when enough linear independent equations are received from the network to create a codeword. An error occurrence in any of the independent paths of the network can be detected and corrected by this algorithm.

For a network where certain constraints have been placed, this technique can be a useful error correcting method. With no control over the network, the receiver is not guaranteed to receive enough linear independent equations to ensure that this method works. With Random Network Coding, the probability of receiving linear dependant equations becomes considerably small when the field size is large [6]. Therefore, a wider network with a larger depth and high connectivity will increase the probability of success of this method.

d) Double error detection: When two single bit errors occur, this method will enable us to detect it. These errors cannot be corrected, but the algorithm will enable us to detect erroneous information.

VII. Conclusion

In this paper, we have presented an algorithm for error detection and correction in random network coding. The introduced technique exploits the encoding characteristics of random network coding and uses the well known Hamming Code as a decoding algorithm.

This technique introduces the concept of the random network coding as the generator of a forward error correction code. Further research should be carried out on how to use packets in the network, instead of bits, as well as implementing other forward error correction codes in random network coding.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, pp. 1204-1216, 2000.

[2] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, 29 June-4 July 2003, p. 442.

[3] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[4] R. Koetter and M. Médard, "Beyond routing: An algebraic approach to network coding," *IEEE/ACM Transaction on Networking*, vol. 11, pp. 782–796, October 2003.

[5] Danilo Silva and Frank R. Kschischang, "Using Rank-Metric Codes for Error Correction in Random Network Coding," *ISIT2007*, June 24 – June 29, 2007

[6] Danilo Silva, Frank R. Kschischang, and Ralf Koetter "A Rank-Metric Approach to Error Control in Random Network Coding," *IEEE International Symposium on Information Theory*, Nice, France, June 2007

[7] R. Hamming, "Error detecting and error correcting codes," Bell *Syst. Tech. Journal,* vol. 29, pp. 41-56, 1950.

[8] David J.C. MacKay, "Information theory, inference and learning algorithms," Cambridge University Press, Cambridge, 2003

[9] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding*," IEEETrans. Inform. Theory*, vol. 49, p. 371, Feb. 2003.

[10] T. K. Moon, "Error Correction Coding Mathematical Methods and Algorithms," John Wiley & Sons, Hoboken, NJ, USA, 2005

[11] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "Toward a random operation of networks," submitted to *IEEE Trans. Inform. Theory*, 2004

[12] C Fragouli, J Le Boudec, and J Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun.* Rev., 36(1):63–68, 2006.

[13] A Leon- Garcia, I Widjaja, "Communication Networks. Fundamental Concepts and Key Architectures," McGraw- Hill Professional, 2003.

[14] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", *in International Symposium on Information Theory (ISIT)*, June 2007.

[15] N. Cai and R. W. Yeung, "Network coding and error correction," Proceedings of IEEE Information Theory Workshop, pages 119-122, October 2002.

[16] D. Axel, "Network Coding: an Overview," *Institute for Communications Engineering (LNT)*, January 2005, pp. 1-19.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft,"Xors in the air: Practical wireless network coding," in *Proc. of ACM Sigcomm 2006*, Pisa, Italy, Sept. 2006.

Error Correction with the Implicit Encoding Capability of Random Network Coding

Suné von Solms, Albert S. J. Helberg
TeleNet Research Group
School for Electric, Electronic and Computer Engineering
North West University, Potchefstroom Campus
Tel: (018) 299 1961, Fax: (018) 299 1977
E- mail: sune.vonsolms@nwu.ac.za, albert.helberg@nwu.ac.za

Abstract—We introduce a novel error correction scheme that uses the implicit encoding capability of Random Network Coding. This scheme does not add redundancy to the data prior to transmission, like existing error correcting schemes. Random Network Coding within a large network generates enough redundant information to perform error correction on transmitted data.

Index Terms— Error Correction, Network Coding, Random Network Coding

I. Introduction

THE concept of Network Coding was first introduced by Ahlswede et al. in 2000 [1]. Instead of simply forwarding data in a network, as in traditional routing, they proposed that nodes may recombine several input packets into one or more output packets. In [1], the combinations formed by the nodes are based on a specific topology.

The concept of Random Network Coding was introduced by Ho et al. in [2]. Their approach provides an improvement in robustness [5] where the success of information reception does not depend on receiving packets that contain the specific transmitted information, but on receiving enough linearly independent packets [6]. Knowledge of the linear combinations of the information contained in each data packet is used to solve a set of simultaneous equations to obtain the transmitted data.

Random Network Coding, described in [2] works as follows: All the nodes in the network, except the receiver node, perform independent random linear mappings of their inputs. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are formed from the inputs of the node. The outputs are chosen independently and randomly and must be non- zero. The receiver node of the network then obtains a series of independent linear combinations which it can use to decode the transmitted data. The receiver has to wait a certain amount of time in order to receive a set of equations that can be used to decode the transmitted message. The receiver node of the network only needs to know the overall linear combination of the source processes in each of the incoming packets. This information is provided by a coding vector that is included in each message overhead [7].

A Random Network Coding environment is not necessarily error free. This disadvantage means that the network can be very sensitive to errors [3]. A single error packet has the potential to infect the whole network and corrupt other packets used by the receiver for decoding. When a corrupted packet is linearly combined with legitimate packets, it can corrupt all the information contained in that packet.

It is possible to address these shortcomings by implementing error correction in the network. An error correction code will be able to correct and detect data packets corrupted due to additive errors. This will improve the robustness of the network where we will be able to obtain the correct information, even when only partially correct information is received.

Yeung and Cai [8] constructed such a linear network code with error-correcting capabilities. In erroneous network channels, Network Error Correction can be applied so that errors occurring in the network can be detected and corrected. Lower and upper bounds are also defined for the specific error-correcting capability of the code. Jaggi et al. [10] addressed the problem of error correction by adding redundancy to the source information that satisfies certain constraints and achieves optimal rates. This redundancy will enable the receiver node to correct network errors.

It can be seen that the topic of error correction in Random Network Coding is of current interest. However, the construction of the error correcting codes is in a concatenated form where error correcting encoding takes place prior to transmission. The implicit encoding capabilities of Random Network Coding for error correction codes are not considered. This fact opens up possibilities in the field of network error detection and correction in Random Network Coding which we aim to exploit.

In this paper, we aim to exploit the implicit encoding capabilities of a network implementing Random Network Coding. This method will encode the information sent by the source within the network, instead of transmitting a codeword encoded at the source.

II. Network Model

We adopt the notation used in [2], [3] of an acyclic network

model. The network is represented by a directed graph $G = (V, E)$. V is the set of nodes in the network and E the set of edges in G which represents the communication channels. $S \in V$ represents the source node and $T \in V$ the sink node in the multiple unicast network. The source node sends messages selected from a source alphabet, Z. Let X be the finite set of code alphabet for the network where $X \in Z$ in a finite field F_q. Each edge, $(a, b) \in E$, in the network has unit capacity; therefore it is able to transmit a single unit of information per unit time.

Definition 1: Information Rate (R) is a measure of the average amount of information that is being carried by a symbol [1]. Information Rate is represented by information symbols sent from the source node into the network, and channel symbols received by the receiver from the network.

$$R = \frac{information\ symbols}{channel\ symbols}$$

Theorem 1: Min-Cut Max-Flow: A network with a single source, S, and receiver, T, is given. This connection can be described as $c = \big(S, T, X(S, T)\big)$. This network problem can only be solved if and only if the rate of the connection *R(c)* is less than or equal to the minimum value of all the cuts between S and T [3]

$$\boldsymbol{maxflow}(T) \geq R(c). \quad (1)$$

Fragouli et al. stated in [11] that when a network $G = (V, E)$ with node $S \in V$ and any non-source node $T \in V$ has a min-cut between S and T of K and a connection of c; then the information can be sent from S to T at a maximum rate $R(c)$ of K. They prove that there exist exactly K edge-disjoint paths between S and T when the min-cut between them is K, therefore:

$$R(c) = K \quad (2)$$

and

$$\boldsymbol{maxflow}(T) \geq K. \quad (3)$$

By the min-cut max-flow theorem, it can be seen that equation (3) is a required condition for any node T to solve the message sent from the source node. This means that for K independent information packets to be sent successfully to the receiver, K edge-disjoint paths must exist in the network that connects the source to the receiver.

According to [7], if K linearly independent packets are sent into the network, the min-cut between the source and the receiver in the network must be large enough to support their transmission. This means that the rate of information transmission between the source node and the receiver node is upper-bounded by the min-cut between the nodes in the network [10].

A. Random Network Coding

Assume that a packet contains a sequence of n symbols from the finite field F_q. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ be the K information packets (vectors of length n over finite field F_q) transmitted by the source node S into the network with a $min - cut \geq K$. Let $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{K'}$ be the K' channel packets received by the receiver node [4].

Random Network Coding is implemented where each network node randomly and independently selects coefficients from a finite field F_q. As described in [12], each information packet $\boldsymbol{y}_i$ in the network formed are random linear combinations of $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$, $\boldsymbol{y} = \sum_{k=1}^{K} \alpha_k \boldsymbol{x}_k$, where $\boldsymbol{\alpha}$ is called the *global encoding vector* of $\boldsymbol{x}$. References [12] as well as [7] assume that this vector is sent along with $\boldsymbol{x}$ in its header. We will adopt the same assumption and send the global encoding vector inside the information packet.

Example 1: Suppose that the source node, S, sends K information packets into the network with a $min - cut \geq K$. Each information packet $\boldsymbol{x}_i, i = 1,2, \dots, K$ has a length n that consists of a information message (length k) along with the header (length $n - k$), as can be seen in Fig 1.

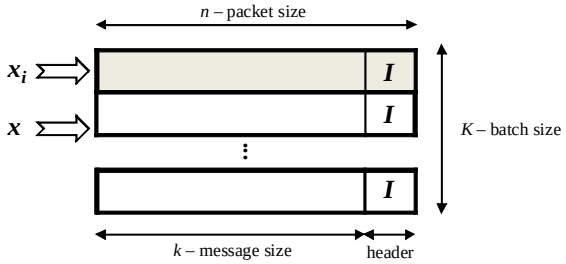


Figure 1: Information Packets sent from the source through the network.

The overhead (header) sent with the information packet has a size $n - k = K \log_2 q$ bits. This is negligibly small if the packet size, n, is sufficiently large [12].

Let the receiver obtain K' channel packets:

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_1 \boldsymbol{x_1} \\ \boldsymbol{y_2} &= \alpha_2 \boldsymbol{x_2} \\ &\vdots \\ \boldsymbol{y_{K'}} &= \alpha_3 \boldsymbol{x_K} \end{aligned} \quad (4)$$

where $\boldsymbol{x}_i, i = 1,2, \dots, K$ are the K information packets sent from the source and α_i are random coefficients. Equation (4) can also be written as

$$\boldsymbol{y} = \boldsymbol{\alpha x}, \quad (5)$$

where $\boldsymbol{x} = [\boldsymbol{x}_{ij}]$ is the $K \times n$ transmitted array formed by stacking the information packets $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ as the rows of $\boldsymbol{x}$, where the subscript of $\boldsymbol{x}_{ij}$ indicates the j'th entry of packet $\boldsymbol{x}_i, i = 1,2, \dots, K$. Also, $\boldsymbol{y} = [\boldsymbol{y}_{ij}]$ is the $K' \times n$ received array formed by stacking the received channel packets $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{K'}$ as the rows of $\boldsymbol{y}$ where the subscript of $\boldsymbol{y}_{ij}$ indicates the j'th entry of packet $\boldsymbol{y}_i, i = 1,2, \dots, K'$. $\boldsymbol{\alpha}$ is a $K' \times K$ matrix over F_q corresponding to the overall transfer function of the network from the source to the receiver [4].

Linearly dependent packets (packets with linearly dependent global encoding vectors) are useless for the decoding of the channel messages at the receiver. The min-cut between the source node and receiver nodes must therefore be large enough to support the transmission of the K linearly independent packets. When the receiver receives K channel

packets with linearly independent global encoding vectors, it will be able to decode the *K* message packets [12].

It can be clearly seen that the information rate of this network is $R = \frac{K}{K} = 1$, where K information symbols are sent into the network ($min - cut \geq K$) and K channel symbols are received.

III. Error Correction in Random Network Coding: Existing Method

We now assume that errors occur in the network. We assume that packet errors occur on the edge of the network. If K information packets are sent over the network, let $\boldsymbol{z}_k$ denote error packets applied to the packet $k \in \{1, \dots, K\}$. Equation (5) then becomes

$$y = \sum_{k=1}^{K} \alpha_k x_k + \sum_{k=1}^{K} \beta_k z_k \tag{6}$$

or

$$\boldsymbol{y} = \boldsymbol{\alpha x} + \boldsymbol{\beta z}, \tag{7}$$

where $\boldsymbol{z} = [\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, \dots, \boldsymbol{z}_K^T]$ is an array consisting of all the erroneous packets introduced in the network and $\boldsymbol{\beta}$ is the overall transfer matrix of these packets from the source to destination. If $\boldsymbol{z}_k = 0$, no errors were applied to message packet $i \in \{1, \dots, K\}$.

A. Redundant symbols

Jaggi et al. described in [10] that the errors that occur in the network can be thought of as a second source. The information received at the receiver is linear combinations of the information of the source as well the error information. This can be seen in (6).

Reference [10] addressed the topic of extracting the source information from the received mixture of channel information and errors. He addressed this problem by adding redundancy to the source information that satisfies certain constraints. This information packet is constructed as in Fig. 2 [10]:

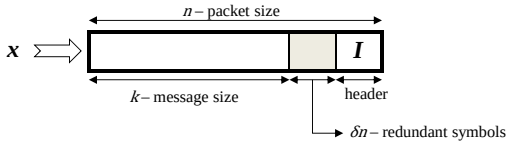


Figure 2: Information Packet with redundancy symbols.

Each packet contains a sequence of n symbols from the finite field F_q. Out of the n symbols in the information packet; δn symbols are redundancy added by the source. The δn redundant symbols are chosen as parity symbols in order for the receiver to decode the channel packet. Also included in the packet is the identity matrix, I, that acts as the global encoding vector by reflecting the linear combinations formed on the channel packet.

B. Redundant packets

Definition 2: A network code is *t-error-correcting* if it can correct all γ-errors for $\gamma \leq t$, i.e., if the total number of errors in the network are at most *t*, then the source message can be recovered by the sink node $T \in V$ [8].

Definition 3: A block code is a rule for converting a sequence of source symbols of length K into a transmitted sequence of length *N* symbols [13].

Yeung and Cai [9] correct errors in network coding not by adding redundancy to each information packet sent, but by adding redundant packets at the source to be sent over the network. They use (N, K) linear block codes and consider these as a linear network code. The source node takes K information packets as its input and outputs N coded message packets, basically adding packets as parity.

We describe this process as follows: Let the information packets $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ (vectors of length n over finite field F_q) be the K message packets of $\boldsymbol{x} = [\boldsymbol{x}_{ij}]$ that must be transmitted over the network. The source node uses a (N, K) block code to encode these K information packets into N outgoing coded packets, denoted as $\boldsymbol{x'}_1, \boldsymbol{x'}_2, \dots, \boldsymbol{x'}_N$, where $N > K$ and $\boldsymbol{x}_i^{'} = \sum_{j=1}^{K} g_{ij} \boldsymbol{x}_j$. These redundant packets are generated by using randomly generated coefficients g_{ij} from a finite field F_{2q}. The set of coefficients $g_{i1}, g_{i2}, \dots, g_{iK}$ can be referred to as the *encoding vector* for $\boldsymbol{x}_i$ [6] and are sent in the information packet as the overhead.

This method can be represented by Fig. 3.

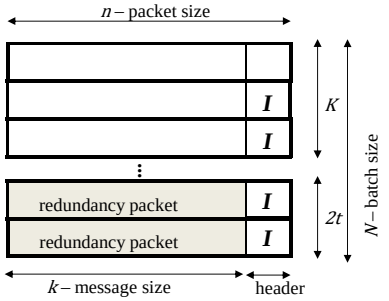


Figure 3: Information packets with redundancy (parity) packets.

Example 2: Assume in this example that the global encoding vector is sent along with the information packet, $\boldsymbol{x}$, in its header. This overhead, however, is negligible because the information packets are sufficiently large, therefore $n \approx k$.

Let $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ (vectors with the length of k symbols over finite field F_q) be the K information packets that must be transmitted over the network. The source node applies a (N, K) forward error correcting block code to linearly combine these K information packets into N coded packets, as can be seen in Fig. 3.

These packets are then transmitted by the source node, S, into the network with a $min - cut \geq N$. Let $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N$ be the N channel packets received by the receiver node, where

$$y = \sum_{n=1}^{N} \alpha_n x_n + \sum_{n=1}^{N} \beta_n z_n \tag{8}$$

The receiver only has to decode the (N, K) block code to successfully regenerate the sent data. The receiver can

decode the message correctly when at most t errors occur, where $t = (N-K)/2$. This means that $\boldsymbol{x}$ is a classical error correcting code that can detect (N, K) and correct $(N, K)/2$ errors.

For Example 2, the information rate of the network $(min - cut \geq N)$ is $R = \frac{N}{N} = 1$, where N information symbols are sent into the network and N channel symbols are received.

IV. Error Correction in Random Network Coding: Proposed Method

Our proposed scheme for network error correction aims to eliminate:

1. the redundancy added to the source packet, or
2. the redundant packets added at the source.

By waiting for more channel packets, the receiver obtains additional information for decoding that may be used for error correction. Thus, the network acts as an error correction encoder. The coded information packets obtained by the receiver provides the redundancy required for error correction. This method differs from that in Section III, because no redundant information is added at the source node.

A. Network Configurations

We introduce a method where the min-cut between the source and the receiver nodes in the network must be large enough to support the transmission of K linearly independent information packets, although N channel packets will be used by the receiver node.

According to [7], the receiver node would normally collect as many channel packets as possible in order to decode the source message. Because of network properties, such as the min-cut between source and receiver node, more than K linearly independent equations are redundant information.

We propose to use this redundant information received by the receiver node to apply error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because the network will transmit sufficient redundancy to the receiver for error correction.

K independent information packets are sent from the source node where the packets propagate through the network. The min-cut of the network remains K, therefore there exist K edge-disjoint paths. The receiver then obtains N channel packets. The extra $(N-K)$ received packets are what we intend to use in order to correct any possible errors. The values of N and K are determined by the specific (N, K) linear block code used. These N channel packets must consist of two sets of linearly independent packets, of size K and (N, K), respectively.

The first set of linearly independent packets is the traditional K packets needed to decode the sent message packets.

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_1 \boldsymbol{x_1} \\ \boldsymbol{y_2} &= \alpha_2 \boldsymbol{x_2} \\ &\vdots \\ \boldsymbol{y_K} &= \alpha_3 \boldsymbol{x_K} \end{aligned} \tag{9}$$

or

$$y_{data} = \sum_{k=1}^{K} \boldsymbol{\alpha_k} x_k, \tag{10}$$

where $y_{data} = [y_{ij}]$ is a $K\ x\ n$ array formed by stacking the received message packets $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K$ as the rows of y_{data}, where the subscript of y_{ij} indicates the j'th entry of packet y_i, $i = 1, 2, \ldots, K$.

The other set of linearly independent packets must be of size $(N-K)$ and will be used for error correction.

$$\begin{aligned} \boldsymbol{y_{1+K}} &= \alpha_1 \boldsymbol{x_{1+K}} \\ \boldsymbol{y_{2+K}} &= \alpha_2 \boldsymbol{x_{2+K}} \\ &\vdots \\ \boldsymbol{y_N} &= \alpha_3 \boldsymbol{x_N} \end{aligned} \tag{11}$$

or

$$y_{parity} = \sum_{k=1+K}^{N} \alpha_k x_k\ , \tag{12}$$

where $y_{parity} = [y_{ij}]$ is a $(N-K) \times n$ array formed by stacking the received message packets $\boldsymbol{y}_{1+K}, \boldsymbol{y}_{2+K}, \ldots, \boldsymbol{y}_N$ as the rows of y_{parity} where the subscript of y_{ij} indicates the j'th entry of packet y_i, $i = 1+K, 2+K, \ldots, N$.

These redundant $(N-K)$ symbols are linear functions of the original K message packets and will act as the parity symbols providing the platform for error detection and correction. When the receiver obtains both sets of channel packets, it decodes the messages as a $(N-K)$ block code. The receiver can decode the message correctly when at most t errors occur, where $t = (N, K)/2$.

B. Example 3

Example 2 revisited: Assume in this example that the global encoding vector is sent along with information packet, $\boldsymbol{x}$, in its header. This overhead, however, is negligible because the packets are sufficiently large, therefore $n \approx k$.

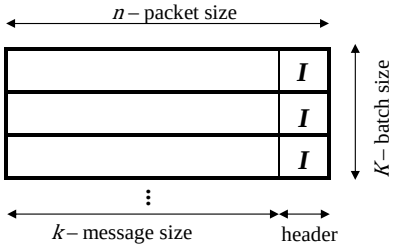


Figure 4: Information packets without redundancy packets.

Let $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K$ (vectors with the length of k symbols over finite field F_q) be the K information packets that must be transmitted over the network. The source node ***does not*** apply a (N, K) forward error correcting block code to the K information packets. The K information packets are transmitted by the source node, S, into the network with a $min - cut \geq K$. The K information packets propagate

through the network; linear combinations are formed from them by intermediate nodes and the receiver waits until it receives N or more channel packets.

Let $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$ be the N message packets received by the receiver node, where

$$y = \sum_{n=1}^{N} \alpha_n x_n + \sum_{n=1}^{N} \beta_n z_n \qquad (13)$$

The receiver then only has to decode the (N, K) block code to successfully regenerate the sent data. The receiver can decode the message correctly when at most t errors occur, where $t = (N-K)/2$. This means that $\boldsymbol{x}$ is an error correcting code that can detect $(N-K)$ and correct $(N-K)/2$ errors.

It can be seen that exactly the same decoding process is used at the receiver end. The difference is that less information is injected into the network. The min-cut of the network is smaller and the information rate is $R = \frac{K}{N}, K < N$.

This method offers benefits in terms of energy efficiency, because less information is injected into the network: the source node transmits K packets instead of N, into a network with a $min - cut \geq K$, instead of $min - cut \geq N$. Example 3 can by summarized by Fig. 5:

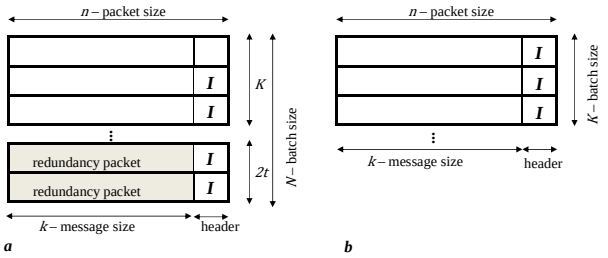


Figure 5: Sent information messages for (a) concatenated and (b) Implicit Error Correction scheme.

V. Analysis

In order to analyze the performance of the proposed and existing schemes, we investigate four aspects of the methods:

1. Probability of receiving enough valid parity packets for decoding
2. Complexity of the decoding algorithms
3. Time delay of the decoding algorithms
4. The error correction capability of the methods in a network with a specific min-cut.

For the proposed method, the possibility exists that the channel packets obtained by the receiver may not contain valid parity packets. This will prevent the receiver from decoding the information successfully. We investigate the probability of receiving a set of valid parity packets from a network so that this Implicit Error Correction method can be applied effectively.

We assume that the network under consideration is a non-cyclic, generic, random network as illustrated in Fig. 6. This network contains a single source node $S \in V$, and a single sink node, $T \in V$. The source node sends the K data packets to the network, which consist of p intermediate nodes. The nodes in the network can send multiple encoded packets to other nodes in the network, but only a selection of $q \leq p$ nodes are connected to the receiver. The receiver node only receives a single encoded packet from each of the q nodes.

We generated a set of 1000 randomly generated networks in order to analyze the Implicit Error Correction capabilities of it. These networks have the following properties:

1. Finite field, F_q.
2. Network size (number of nodes).
3. $Min - cut \geq K$.
4. Intermediate nodes consisting of p nodes, where q is connected to the receiver, $q \leq p$.
5. Each node in the network randomly and independently generates a linear combination of its inputs.

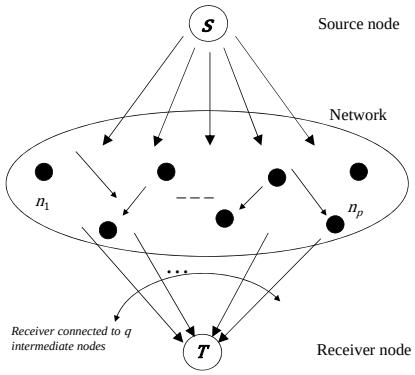


Figure 6: Random Network which contains K a single source node, p intermediate nodes and a single receiver.

A. Probability of Decoding

The linear equations obtained by the receiver are evaluated to determine if the received combinations are valid sets of parity packets. The average percentage of valid sets received in each size network can be viewed in Fig. 7.

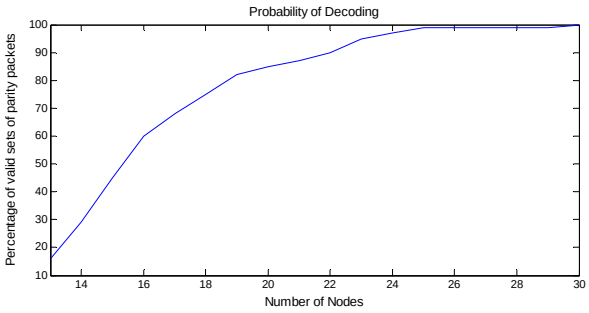


Figure 7: Valid sets of Parity packets received in the network.

It is clear that one can only expect to receive a guaranteed set of parity packets with a network containing about 30 nodes. We have chosen a 30-node network for all the following calculations and simulations, because a valid set of parity packets will be guaranteed, under the parameters for network size and complexity as discussed.

B. Complexity

In the Error Correction method proposed in Section III, the time complexity of the decoding method is estimated to be $O(N)$, where all operations are in F_q. As discussed in Section IV.A, the receiver of the Implicit Error Correction

method obtains N packets from the network. From these, data packets and valid parity packets must first be calculated, and then decoded. The estimated complexity of this decoding algorithm is $O(N^2 - K^2)$.

It can be seen that the computing complexity of this method is higher than that of the existing decoding method.

C. Time Delay

The cost of the higher computing complexity is an extended waiting period for decoding. Fig. 8 shows the time of the decoding algorithm of the Implicit Error Correcting method relative to the time for traditional decoding.
It is clearly visible that the time consumption of the implicit method is higher for decoding. However, for very large block codes ($n < 2000$), the time consumption for the Implicit Error Correcting method approaches that of the existing method.

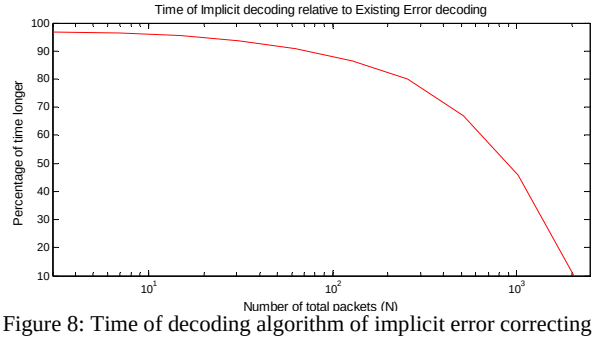


Figure 8: Time of decoding algorithm of implicit error correcting scheme relative to the time for decoding described in Section III.

D. Error correcting Capability

One advantage of the Implicit Error Correcting method is the fact that a *t*-error-correcting code can be applied successfully to a network with $min - cut \geq K$ instead of a network with $min - cut \geq N$, where $t = (N - K)/2$. This advantage can be seen in Fig. 9.

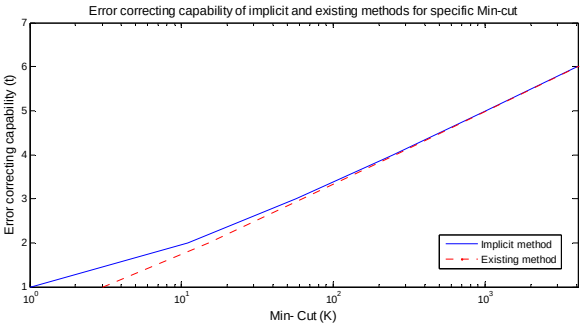


Figure 9: Error correcting capability of the Implicit and Existing methods for a specified *t*- error correcting code.

VI. Conclusion

We have introduced a novel scheme that makes effective use of information implicitly generated in the network to perform error correction. The redundant information needed for error correction is generated in the network, and not sent from the source node. This forward error correction method maps a set of information symbols to a set of code symbols resulting in an information rate of less than 1.

The biggest advantage achieved by using the implicit encoding capabilities of Random Network Coding is the fact that the source does not implement error correction. Only the receiver applies error correction codes. This means that the receiver can apply any error correction code it chooses (example: Hamming, Reed Solomon etc.) without informing the source. The receiver bases the decision purely on the information it receives. Another advantage of the Implicit Error Correcting method is that the scheme allows greater error correcting capability than the existing scheme for a network with the same min-cut, or vice versa. The requirements of the network are reduced, both in connectivity (min-cut) and bandwidth required.

The time consumption due to decoding complexity concerning this method is the biggest trade-off for the advantage of effective error correction in this scheme.

Encoding the information within the network, instead of transmitting the already encoded codeword over the network leads to a saving in network bandwidth. This method leads to an improvement in the network's information rate. The saving of energy may be of interest to energy constraint networks such as wireless sensor networks.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000.

[2] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in Proc. IEEE Int. Symp. Information Theory, Yokohama, 29 June-4 July 2003, p. 442.

[3] R. Koetter and M. Médard, "Beyond routing: An algebraic approach to network coding," IEEE/ACM Transaction on Networking, vol. 11, pp. 782–796, October 2003.

[4] D. Silva, F.R. Kschischang, and R. Koetter "A Rank-Metric approach to error control in random network coding," IEEE International Symposium on Information Theory, Nice, France, June 2007

[5] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "Toward a random operation of networks," submitted to IEEE Trans. Inform. Theory, 2004

[6] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: an instant primer," SIGCOMM Comput. Commun. Rev., 36(1):63–68, 2006.

[7] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", IEEE Transactions on Information Theory, vol. 54, no. 8, pp. 3579-3591, Aug. 2008.

[8] N. Cai and R. W. Yeung, "Network coding and error correction," Proceedings of IEEE Information Theory Workshop, pages 119-122, October 2002.

[9] R.W. Yeung and N Cai, "Network Coding, Algebraic Coding, and Network Error Correction", in Proc. Information Theory and Applications Workshop, La Jolla, CA, February 2006

[10] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of Byzantine adversaries," in Proc. 26th IEEE Int. Conf.

on Computer Commun., Anchorage, AK, May 2007, pp. 616–624.

[11] C. Fragouli and E. Soljanin, "Network Coding Fundamentals," Foundations and Trends in Networking, Vol. 2, No. 1, 2007

[12] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Médard and N. Ratnakar "Network Coding for Wireless Applications; A Brief Tutorial", International Workshop on Wireless and Ad-hoc Networks (IWWAN), May 2005.

[13] D. J. C. MacKay, "Information Theory, Inference, and Learning Algorithms," Cambridge University Press, 2003

Error Correction with the Implicit Encoding Capability of Random Network Coding

Suné von Solms, MJ Grobler and Albert SJ Helberg

School for Electric, Electronic and Computer Engineering, North West University – Potchefstroom Campus, Potchefstroom, South Africa
{sune.vonsolms,albert.helberg}@nwu.ac.za

Abstract. In this paper, we present a technique for a network error correcting code using Random Network Coding. We introduce a novel error correction scheme that uses the implicit encoding capability of Random Network Coding. This scheme does not add redundancy to the data prior to transmission, but ads redundancy based on information already contained in the network. Random Network Coding within a large network generates enough redundant information to perform error correction on transmitted data.

Keywords. Error Correction, Network Coding, Random Network Coding

1 Introduction

The concept of Network Coding was first introduced by Ahlswede et al. in 2000 [1]. Instead of simply forwarding data in a network, as in traditional routing, they proposed that nodes may recombine several input packets into one or more output packets. In [1], the combinations formed by the nodes are based on a specific topology.

The concept of Random Network Coding was introduced by Ho et al. in [2]. Their approach provides an improvement in robustness [5] where the success of information reception does not depend on receiving packets that contain the specific transmitted information, but on receiving enough linearly independent packets [6]. Knowledge of the linear combinations of the information contained in each data packet is used to solve a set of simultaneous equations to obtain the transmitted data.

Random Network Coding, described in [2] works as follows: All the nodes in the network, except the receiver node, perform independent random linear mappings of their inputs. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are formed from the inputs of the node. The outputs are chosen independently and randomly and must be non- zero. The receiver node of the network then obtains a series of independent linear combinations which it can use to decode the transmitted data. The receiver has to wait a certain amount of time in order to receive a set of equations that can be used

to decode the transmitted message. The receiver node of the network only needs to know the overall linear combination of the source processes in each of the incoming packets. This information is provided by a coding vector that is included in each message overhead [7].

A Random Network Coding environment is not necessarily error free. This disadvantage means that the network can be
very sensitive to errors [3]. A single error packet has the potential to infect the whole network and corrupt other packets used by the receiver for decoding. When a corrupted packet is linearly combined with legitimate packets, it can corrupt all the information contained in that packet.

It is possible to address these shortcomings by implementing error correction in the network. An error correction code will be able to correct and detect data packets corrupted due to additive errors. This will improve the robustness of the network where we will be able to obtain the correct information, even when only partially correct information is received.

Yeung and Cai [8] constructed such a linear network code with error-correcting capabilities. In erroneous network channels, Network Error Correction can be applied so that errors occurring in the network can be detected and corrected. Lower and upper bounds are also defined for the specific error-correcting capability of the code. Jaggi et al. [10] addressed the problem of error correction by adding redundancy to the source information that satisfies certain constraints and achieves optimal rates. This redundancy will enable the receiver node to correct network errors.

It can be seen that the topic of error correction in Random Network Coding is of current interest. However, the construction of the error correcting codes is in a concatenated form where error correcting encoding takes place prior to transmission. The implicit encoding capabilities of Random Network Coding for error correction codes are not considered. This fact opens up possibilities in the field of network error detection and correction in Random Network Coding which we aim to exploit.

In this paper, we aim to exploit the implicit encoding capabilities of a network implementing Random Network Coding. This method will encode the information sent by the source within the network, instead of transmitting a codeword encoded at the source.

2 Network Model

We adopt the notation used in [2], [3] of an acyclic network model. The network is represented by a directed graph $G = (V, E)$. V is the set of nodes in the network and E the set of edges in G which represents the communication channels. $S \in V$ represents the source node and $\mathrm{T} \in \mathrm{V}$ the sink node in the multiple unicast network. The source node sends messages selected from a source alphabet, Z. Let X be the

finite set of code alphabet for the network where $X \in Z$ in a finite field F_q. Each edge, $(a,\ b) \in E$, in the network has unit capacity; therefore it is able to transmit a single unit of information per unit time.

Definition 1: Information Rate (R) is a measure of the average amount of information that is being carried by a symbol [1]. Information Rate is represented by information symbols sent from the source node into the network, and channel symbols received by the receiver from the network.

$$R = \frac{information\ symbols}{channel\ symbols}$$

Theorem 1: Min-Cut Max-Flow: A network with a single source, S, and receiver, T, is given. This connection can be described as $c = \big(S,\ T,\ X(S,\ T)\big)$. This network problem can only be solved if and only if the rate of the connection *R(c)* is less than or equal to the minimum value of all the cuts between S and T [3]

$$maxflow(T) \geq R(c). \quad \textbf{(1)}$$

Fragouli et al. stated in [11] that when a network $G = (V,\ E)$ with node S ∈ V and any non-source node $T \in V$ has a min-cut between S and T of K and a connection of c; then the information can be sent from S to T at a maximum rate $R(c)$ of K. They prove that there exist exactly K edge-disjoint paths between S and T when the min-cut between them is K, therefore:

$$R(c) = K \quad \textbf{(2)}$$

and

$$maxflow(T) \geq K. \quad \textbf{(3)}$$

By the min-cut max-flow theorem, it can be seen that equation (3) is a required condition for any node T to solve the message sent from the source node. This means that for K independent information packets to be sent successfully to the receiver, K edge-disjoint paths must exist in the network that connects the source to the receiver.

According to [7], if K linearly independent packets are sent into the network, the min-cut between the source and the receiver in the network must be large enough to support their transmission. This means that the rate of information transmission between the source node and the receiver node is upper-bounded by the min-cut between the nodes in the network [10].

2.1 Random Network Coding

Assume that a packet contains a sequence of n symbols from the finite field F_q. Let $x_1, x_2, \ldots, x_K$ be the K information packets (vectors of length n over finite field F_q)

transmitted by the source node S into the network with a $min - cut \geq K$. Let $y_1, y_2, \dots, y_{K'}$ be the K' channel packets received by the receiver node [4].

Random Network Coding is implemented where each network node randomly and independently selects coefficients from a finite field F_q. As described in [12], each information packet y_i in the network formed are random linear combinations of $x_1, x_2, \dots, x_K$,

$$y = \sum_{k=1}^{K} \alpha_k \boldsymbol{x}_k \tag{4}$$

where α is called the global encoding vector of x. References [12] as well as [7] assume that this vector is sent along with x in its header. We will adopt the same assumption and send the global encoding vector inside the information packet.

Example 1: Suppose that the source node, S, sends K information packets into the network with a $min - cut \geq K$. Each information packet $x_i, i = 1,2, \dots, K$ has a length n that consists of a information message (length k) along with the header (length $n - k$), as can be seen in Fig 1.

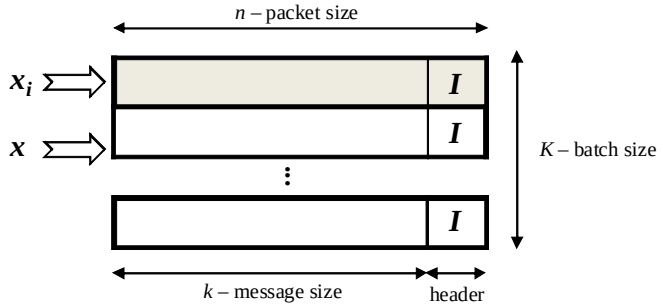


Fig 1: Information Packets sent from the source through the network.

The overhead (header) sent with the information packet has a size $n - k = K \log_2 q$ bits. This is negligibly small if the packet size, n, is sufficiently large [12].

Let the receiver obtain K' channel packets:

$$\begin{aligned} y_1 &= \alpha_1 x_1 \\ y_2 &= \alpha_2 x_2 \\ &\vdots \\ y_{K'} &= \alpha_3 x_K \end{aligned} \tag{5}$$

where $x_i, i = 1,2, \dots, K$ are the K information packets sent from the source and α_i are random coefficients. Equation (5) can also be written as

$$y = \alpha x, \tag{6}$$

where $x = [x_{ij}]$ is the $K \times n$ transmitted array formed by stacking the information

packets $x_1, x_2, \dots, x_K$ as the rows of x, where the subscript of x_{ij} indicates the *j*'th entry of packet $x_i, i = 1,2, \dots, K$. Also, $y = [y_{ij}]$ is the $K' \times n$ received array formed by stacking the received channel packets $y_1, y_2, \dots, y_{K'}$ as the rows of y where the subscript of y_{ij} indicates the j'th entry of packet $y_i, i = 1,2, \dots, K'$. α is a $K' \times K$ matrix over F_q corresponding to the overall transfer function of the network from the source to the receiver [4].

Linearly dependent packets (packets with linearly dependent global encoding vectors) are useless for the decoding of the channel messages at the receiver. The min-cut between the source node and receiver nodes must therefore be large enough to support the transmission of the K linearly independent packets. When the receiver receives K channel packets with linearly independent global encoding vectors, it will be able to decode the K message packets [12].

It can be clearly seen that the information rate of this network is

$$R = \frac{K}{K} = 1,$$

where K information symbols are sent into the network ($min - cut \geq K$) and K channel symbols are received.

3 Error Correction in Random Network Coding: Traditional Method

We now assume that errors occur in the network. We assume that packet errors occur on the edge of the network. If K information packets are sent over the network, let z_k denote error packets applied to the packet $k \in \{1, \dots, K\}$. Equation (6) then becomes

$$y = \sum_{k=1}^{K} \alpha_k x_k + \sum_{k=1}^{K} \beta_k z_k \quad \textbf{(7)}$$

or

$$y = \alpha x + \beta z, \quad \textbf{(8)}$$

where $z = [z_1^T, z_2^T, \dots, z_K^T]$ is an array consisting of all the erroneous packets introduced in the network and β is the overall transfer matrix of these packets from the source to destination. If $z_k = 0$, no errors were applied to message packet $i \in \{1, \dots, K\}$.

3.1 Redundant Symbols

Jaggi et al. described in [10] that the errors that occur in the network can be thought

of as a second source. The information received at the receiver is linear combinations of the information of the source as well the error information. This can be seen in (7).

Reference [10] addressed the topic of extracting the source information from the received mixture of channel information and errors. He addressed this problem by adding redundancy to the source information that satisfies certain constraints. This information packet is constructed as in Fig. 2 [10]:

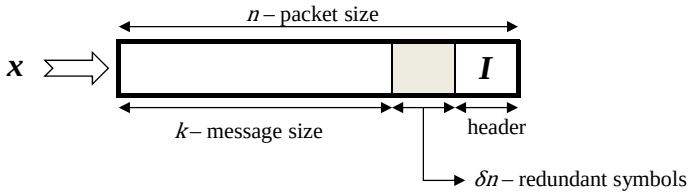


Fig 2: Information Packet with redundancy symbols.

Each packet contains a sequence of n symbols from the finite field F_q. Out of the n symbols in the information packet; δn symbols are redundancy added by the source. The δn redundant symbols are chosen as parity symbols in order for the receiver to decode the channel packet. Also included in the packet is the identity matrix, I, that acts as the global encoding vector by reflecting the linear combinations formed on the channel packet.

3.2 Redundant Packets

Definition 2: A network code is *t-error-correcting* if it can correct all γ-errors for $\gamma \leq t$, i.e., if the total number of errors in the network are at most *t*, then the source message can be recovered by the sink node $T \in V$ [8].

Definition 3: A block code is a rule for converting a sequence of source symbols of length K into a transmitted sequence of length N symbols [13].

Yeung and Cai [9] correct errors in network coding not by adding redundancy to each information packet sent, but by adding redundant packets at the source to be sent over the network. They use (N, K) linear block codes and consider these as a linear network code. The source node takes K information packets as its input and outputs N coded message packets, basically adding packets as parity.

We describe this process as follows: Let the information packets $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K$ (vectors of length n over finite field F_q) be the K message packets of $\boldsymbol{x} = [\boldsymbol{x}_{ij}]$ that must be transmitted over the network. The source node uses a (N, K) block code to encode these K information packets into N outgoing coded packets, denoted as $\boldsymbol{x'}_1, \boldsymbol{x'}_2, \ldots, \boldsymbol{x'}_N$, where $N > K$ and

$$x'_i = \sum_{j=1}^{K} g_{ij} \boldsymbol{x}_j \quad \textbf{(9)}$$

These redundant packets are generated by using randomly generated coefficients g_{ij} from a finite field F_{2q}. The set of coefficients $g_{i1}, g_{i2}, \dots, g_{iK}$ can be referred to as the *encoding vector* for $\boldsymbol{x}_i$ [6] and are sent in the information packet as the overhead.

This method can be represented by Fig. 3.

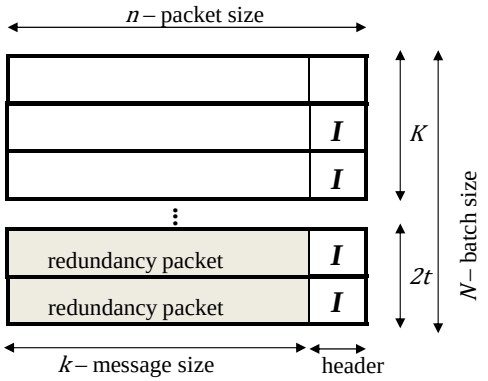


Fig 3: Information packets with redundancy (parity) packets.

Example 2: Assume in this example that the global encoding vector is sent along with the information packet, $\boldsymbol{x}$, in its header. This overhead, however, is negligible because the information packets are sufficiently large, therefore $n \approx k$.

Let $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ (vectors with the length of k symbols over finite field F_q) be the K information packets that must be transmitted over the network. The source node applies a (N, K) forward error correcting block code to linearly combine these K information packets into N coded packets, as can be seen in Fig. 3.

These packets are then transmitted by the source node, S, into the network with a $min - cut \geq N$. Let $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N$ be the N channel packets received by the receiver node, where

$$y = \sum_{n=1}^{N} \alpha_n x_n + \sum_{n=1}^{N} \beta_n z_n \quad \textbf{(10)}$$

The receiver only has to decode the (N, K) block code to successfully regenerate the sent data. The receiver can decode the message correctly when at most t errors occur, where $t = (N - K)/2$. This means that $\boldsymbol{x}$ is a classical error correcting code that can detect (N, K) and correct $(N, K)/2$ errors.

For Example 2, the information rate of the network ($min - cut \geq N$) is

$$R = \frac{N}{N} = 1,$$

where N information symbols are sent into the network and N channel symbols are received.

4 Error Correction in Random Network Coding: Proposed Method

Our proposed scheme for network error correction aims to eliminate:

1. the redundancy added to the source packet, or
2. the redundant packets added at the source.

By waiting for more channel packets, the receiver obtains additional information for decoding that may be used for error correction. Thus, the network acts as an error correction encoder. The coded information packets obtained by the receiver provides the redundancy required for error correction. This method differs from that in Section 3, because no redundant information is added at the source node.

4.1 Network Configurations

We introduce a method where the min-cut between the source and the receiver nodes in the network must be large enough to support the transmission of K linearly independent information packets, although N channel packets will be used by the receiver node.

According to [7], the receiver node would normally collect as many channel packets as possible in order to decode the source message. Because of network properties, such as the min-cut between source and receiver node, more than K linearly independent equations are redundant information.

We propose to use this redundant information received by the receiver node to apply error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because the network will transmit sufficient redundancy to the receiver for error correction.

K independent information packets are sent from the source node where the packets propagate through the network. The min-cut of the network remains K, therefore there exist K edge-disjoint paths. The receiver then obtains N channel packets. The extra $(N - K)$ received packets are what we intend to use in order to correct any possible errors. The values of N and K are determined by the specific (N, K) linear block code used. These N channel packets must consist of two sets of linearly independent packets, of size K and (N, K), respectively.

The first set of linearly independent packets is the traditional K packets needed to decode the sent message packets.

$$\begin{aligned} \boldsymbol{y_1} &= \alpha_1 \boldsymbol{x_1} \\ \boldsymbol{y_2} &= \alpha_2 \boldsymbol{x_2} \\ &\vdots \\ \boldsymbol{y_K} &= \alpha_3 \boldsymbol{x_K} \end{aligned} \quad \textbf{(11)}$$

or

$$y_{data} = \sum_{k=1}^{K} \boldsymbol{\alpha_k} x_k, \quad \textbf{(12)}$$

where $y_{data} = [y_{ij}]$ is a $K\ x\ n$ array formed by stacking the received message packets $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K$ as the rows of y_{data}, where the subscript of y_{ij} indicates the *j*'th entry of packet y_i , $i = 1, 2, \ldots, K$.

The other set of linearly independent packets must be of size $(N - K)$ and will be used for error correction.

$$\begin{aligned} \boldsymbol{y_{1+K}} &= \alpha_1 \boldsymbol{x_{1+K}} \\ \boldsymbol{y_{2+K}} &= \alpha_2 \boldsymbol{x_{2+K}} \\ &\vdots \\ \boldsymbol{y_N} &= \alpha_3 \boldsymbol{x_N} \end{aligned} \quad \textbf{(13)}$$

or

$$y_{parity} = \sum_{k=1+K}^{N} \alpha_k x_k\ , \quad \textbf{(14)}$$

where $y_{parity} = [y_{ij}]$ is a $(N - K) \times n$array formed by stacking the received message packets $\boldsymbol{y}_{1+K}, \boldsymbol{y}_{2+K}, \ldots, \boldsymbol{y}_N$ as the rows of y_{parity} where the subscript of y_{ij} indicates the *j*'th entry of packet y_i , $i = 1 + K, 2 + K, \ldots, N$.

These redundant $(N - K)$ symbols are linear functions of the original K message packets and will act as the parity symbols providing the platform for error detection and correction. When the receiver obtains both sets of channel packets, it decodes the messages as a $(N - K)$ block code. The receiver can decode the message correctly when at most *t* errors occur, where $t = (N, K)/2$.

4.2 Example 3

Example 2 revisited: Assume in this example that the global encoding vector is sent along with information packet, $\boldsymbol{x}$, in its header. This overhead, however, is negligible because the packets are sufficiently large, therefore $n \approx k$.

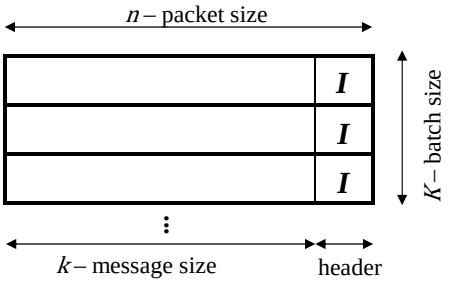


Fig 4: Information packets without redundancy packets.

Let $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K$ (vectors with the length of k symbols over finite field F_q) be the K information packets that must be transmitted over the network. The source node ***does not*** apply a (N, K) forward error correcting block code to the K information packets. The K information packets are transmitted by the source node, S, into the network with a $min - cut \geq K$. The K information packets propagate through the network; linear combinations are formed from them by intermediate nodes and the receiver waits until it receives N or more channel packets.

Let $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N$ be the N message packets received by the receiver node, where

$$y = \sum_{n=1}^{N} \alpha_n x_n + \sum_{n=1}^{N} \beta_n z_n \qquad \textbf{(15)}$$

The receiver then only has to decode the (N, K) block code to successfully regenerate the sent data. The receiver can decode the message correctly when at most t errors occur, where $t = (N - K)/2$. This means that $\boldsymbol{x}$ is an error correcting code that can detect $(N - K)$ and correct $(N - K)/2$ errors.

It can be seen that exactly the same decoding process is used at the receiver end. The difference is that less information is injected into the network. The min-cut of the network is smaller and the information rate is

$$R = \frac{K}{N} < 1 \, , K < N$$

This method offers benefits in terms of energy efficiency, because less information is injected into the network: the source node transmits K packets instead of N, into a network with a $min - cut \geq K$, instead of $min - cut \geq N$. Example 3 can by summarized by Fig. 5:

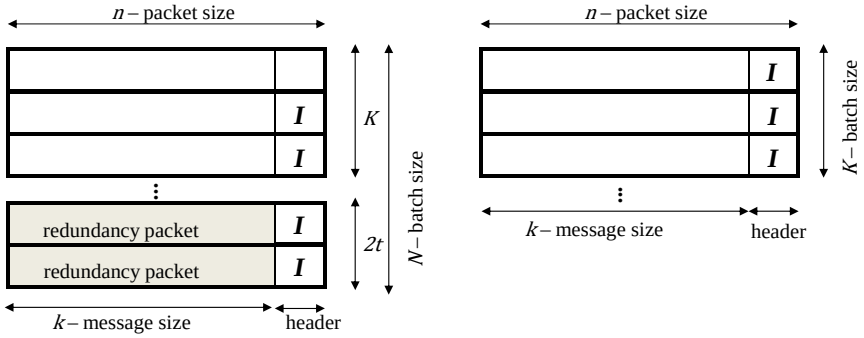


Fig 5: Sent information messages for (a) concatenated and (b) Implicit Error Correction scheme.

5 Analysis

In order to analyze the performance of the proposed and existing schemes, we investigate four aspects of the methods:

1. Probability of receiving enough valid parity packets for decoding
2. Complexity of the decoding algorithms
3. Time delay of the decoding algorithms
4. The error correction capability of the methods in a network with a specific min-cut.

For the proposed method, the possibility exists that the channel packets obtained by the receiver may not contain valid parity packets. This will prevent the receiver from decoding the information successfully. We investigate the probability of receiving a set of valid parity packets from a network so that this Implicit Error Correction method can be applied effectively.

We assume that the network under consideration is a non-cyclic, generic, random network as illustrated in Fig. 6. This network contains a single source node $S \in V$, and a single sink node, $T \in V$. The source node sends the K data packets to the network, which consist of p intermediate nodes. The nodes in the network can send multiple encoded packets to other nodes in the network, but only a selection of $q \leq p$ nodes are connected to the receiver. The receiver node only receives a single encoded packet from each of the q nodes.

We generated a set of 1000 randomly generated networks in order to analyze the Implicit Error Correction capabilities of it. These networks have the following properties:

1. Finite field, F_q.
2. Network size (number of nodes).
3. $Min - cut \geq K$.

4. Intermediate nodes consisting of p nodes, where q is connected to the receiver, $q \leq p$.
5. Each node in the network randomly and independently generates a linear combination of its inputs.

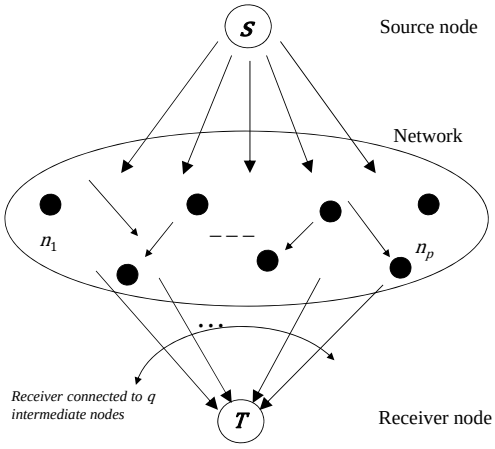


Fig 6: Random Network which contains K source nodes, p + q intermediate nodes (layer 1 and 2) and a single receiver.

5.1 Probability of Decoding

The linear equations obtained by the receiver are evaluated to determine if the received combinations are valid sets of parity packets. The average percentage of valid sets received in each size network can be viewed in Fig. 7.

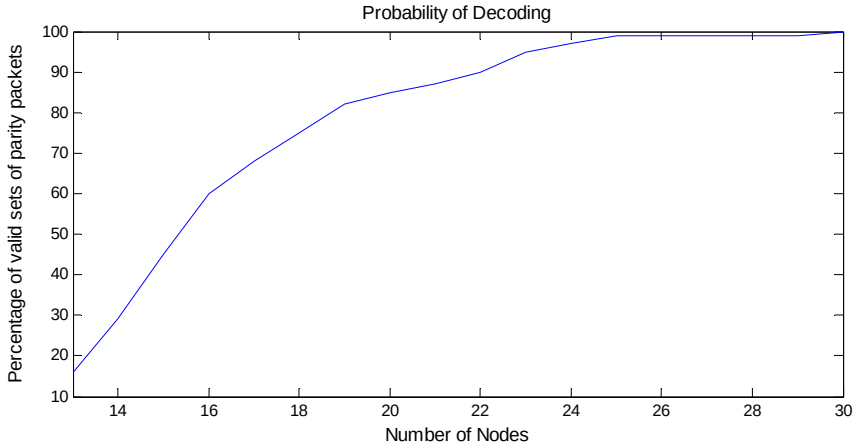


Fig 6: Valid sets of Parity packets received in the network.

It is clear that one can only expect to receive a guaranteed set of parity packets with a network containing about 30 nodes. We have chosen a 30-node network for all the following calculations and simulations, because a valid set of parity packets will be guaranteed, under the parameters for network size and complexity as discussed.

5.2 Discussion of Complexity

In the Error Correction method proposed in Section III, the time complexity of the decoding method is estimated to be $O(N)$, where all operations are in F_q. As discussed in Section IV.A, the receiver of the Implicit Error Correction method obtains N packets from the network. From these, data packets and valid parity packets must first be calculated, and then decoded. The estimated complexity of this decoding algorithm is $O(N^2 - K^2)$.

It can be seen that the computing complexity of this method is higher than that of the existing decoding method.

5.3 Time Delay

The cost of the higher computing complexity is an extended waiting period for decoding. Fig. 8 shows the time of the decoding algorithm of the Implicit Error Correcting method relative to the time for traditional decoding.

It is clearly visible that the time consumption of the implicit method is higher for decoding. However, for very large block codes ($n < 2000$), the time consumption for the Implicit Error Correcting method approaches that of the existing method.

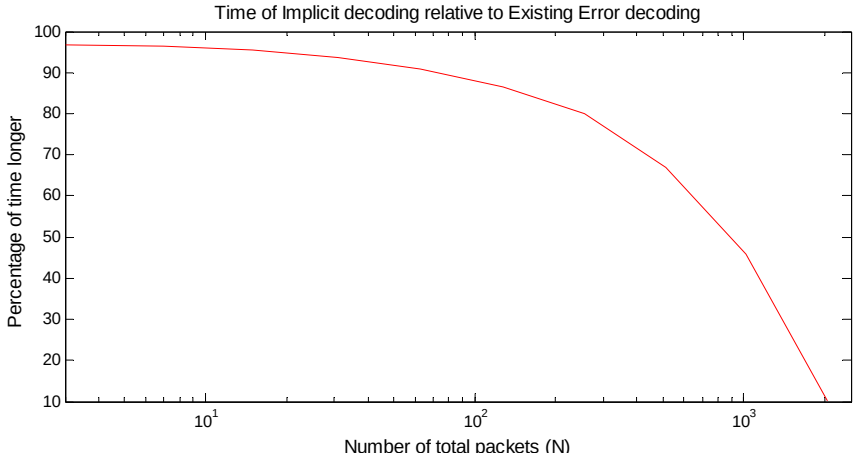


Fig 7: Time of decoding algorithm of implicit error correcting scheme relative to the time for decoding described in Section 3.

5.4 Error Correcting Capability

One advantage of the Implicit Error Correcting method is the fact that a *t*-error-correcting code can be applied successfully to a network with $min - cut \geq K$ instead of a network with $min - cut \geq N$, where $t = (N - K)/2$. This advantage can be seen in Fig. 9.

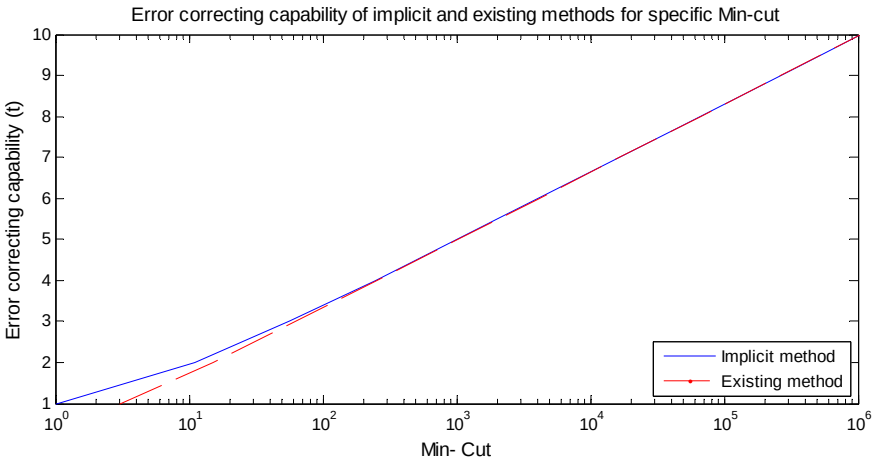


Fig 8: Error correcting capability of the Implicit and Existing methods for a specified *t*- error correcting code.

6 Advantages

We have introduced a novel scheme that makes effective use of information implicitly generated in the network to perform error correction. The redundant information needed for error correction is generated in the network, and not sent from the source node. This forward error correction method maps a set of information symbols to a set of code symbols resulting in an information rate of less than 1.

The biggest advantage achieved by using the implicit encoding capabilities of Random Network Coding is the fact that the source does not implement error correction. Only the receiver applies error correction codes. This means that the receiver can apply any error correction code it chooses (example: Hamming, Reed Solomon etc.) without informing the source. The receiver bases the decision purely on the information it receives. Another advantage of the Implicit Error Correcting method is that the scheme allows greater error correcting capability than the existing scheme for a network with the same min-cut, or vice versa. The requirements of the network are reduced, both in connectivity (min-cut) and bandwidth required.

The time consumption due to decoding complexity concerning this method is the biggest trade-off for the advantage of effective error correction in this scheme.

6 Conclusion

Encoding the information within the network, instead of transmitting the already encoded codeword over the network leads to a saving in network bandwidth. This method leads to an improvement in the network's information rate. The saving of energy may be of interest to energy constraint networks such as wireless sensor networks.

References

1. R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000.
2. T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in Proc. IEEE Int. Symp. Information Theory, Yokohama, 29 June-4 July 2003, p. 442.
3. R. Koetter and M. Médard, "Beyond routing: An algebraic approach to network coding," IEEE/ACM Transaction on Networking, vol. 11, pp. 782–796, October 2003.
4. D. Silva, F.R. Kschischang, and Ralf Koetter "A Rank-Metric approach to error control in random network coding," IEEE International Symposium on Information Theory, Nice, France, June 2007
5. T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "Toward a random operation of networks," submitted to IEEE Trans. Inform. Theory, 2004
6. C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: an instant primer," SIGCOMM Comput. Commun. Rev., 36(1):63–68, 2006.
7. R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", in International Symposium on Information Theory (ISIT), June 2007.
8. N. Cai and R. W. Yeung, "Network coding and error correction," Proceedings of IEEE Information Theory Workshop, pages 119-122, October 2002.
9. R.W. Yeung and N Cai, "Network Coding, Algebraic Coding, and Network Error Correction", in Proc. Information Theory and Applications Workshop, La Jolla, CA, February 2006
10. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of Byzantine adversaries," in Proc. 26th IEEE Int. Conf. on Computer Commun., Anchorage, AK, May 2007, pp. 616–624.
11. C. Fragouli and Emina Soljanin, "Network Coding Fundamentals," Foundations and Trends in Networking, Vol. 2, No. 1, 2007
12. S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Médard and N. Ratnakar, "Network Coding for Wireless Applications; A Brief Tutorial", International Workshop on Wireless and Ad-hoc Networks (IWWAN), May 2005.
13. D. J. C. MacKay, "Information Theory, Inference, and Learning Algorithms," Cambridge University Press, 2003

Performance of Implicit Error Correction of Networks using Random Network Coding

S von Solms, ASJ Helberg
School for Electric Electronic and Computer Engineering
North West University, Potchefstroom Campus
27 (0)18 299 1961
sune.vonsolms@nwu.ac.za

ABSTRACT

We investigate the performance of the Implicit Error Correcting scheme applied to networks utilizing Random Network Coding. This scheme does not add redundancy to the data prior to transmission, like existing error correcting schemes. It exploits the implicit encoding capabilities of the network in order to create enough redundancy for successful error correction. A large network using Random Network Coding generates enough redundant information in order to perform error correction on transmitted data.

A Random Network Coding environment, however, is not necessarily error free. Error propagation in networks is a major weakness in Random Network Coding and has an influence on the performance of the Implicit Error Correction ability of networks.

Firstly, we introduce the Implicit Error Correction capabilities of networks using Random Network Coding and how this capability is used. Secondly, the performance and reliability of Implicit Error Correction in the presence of errors is evaluated, as well as the effect of error propagation on the scheme.

Categories and Subject Descriptors

General Terms

Performance, Design, Reliability

Keywords

Error Correction, Error Propagation, Network Coding, Random Network Coding

1. INTRODUCTION

The traditional way of managing a network, by routing, was to attempt to avoid collisions in data streams as far as possible. In 2000 R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung [1] proposed that if intermediate nodes in a network are allowed to process the information it receives, the achievable rate of a multicast network can increase compared to straightforward routing. This approach, named network coding, requires the intermediate nodes to perform linear combinations on the received information. In [1], the combinations formed by the nodes are based on a specific topology.

The main advantages of Network Coding, compared to traditional routing techniques, are improvements in throughput, ease of management and a higher degree of robustness. These advantages are achieved by the better use of resources in the network where each node implemented with network coding receives the information from all the input nodes, encodes it, and sends it out to the receiver nodes

The concept of Random Network Coding was introduced by Ho, Koetter, Médard, Karger, and Effros in [7]. They describe Random Network Coding as follows: "*Network nodes independently and randomly select linear mappings from inputs onto output links over some field*." This means that all the nodes in the network, except the receiver node, perform independent random linear mappings of their inputs. This creates independent linear combinations that are then forwarded to the next node, where once again random linear combinations are formed from the inputs of the node. The outputs are chosen independently and randomly and must be non- zero.

According to [11], the receiver node would normally collect as many channel packets as possible in order to decode the source message. The additional packets collected by the receiver are redundant information and discarded.

In their paper, they present a randomized coding approach of information in networks that provides a series of advantages over traditional routing based approaches. Their approach was to exploit the maximum capacity of the network by spreading the information over the available network capacity in order to keep the network flexible. By doing so, changes in the network's topology can be accommodated and therefore make the network more robust.

The improvement in robustness lead to the fact that the success of information reception does not depend on receiving packets that

contain the specific transmitted information, but on receiving enough linear independent packets [5]. Knowledge of the linear combinations of the information contained in each data packet is used to solve a set of simultaneous equations to obtain the transmitted data [12].

However, Network Coding environments are subjected to a variety of hostile factors like packet-losses, link failures or the presence of errors [16]. According to [10], networks implementing Network Coding can be very sensitive to errors, defined by [14] as an occurrence where the output symbol of a channel differs from the corresponding input symbol.

The need for reliability and the capability of networks to counter the effect of errors are therefore important characteristics of networks required today.

These needs are widely addressed by implementing error correction in networks. Error correction for a network that implements Network Coding is of current interest since 2002 [2]-[4]. According to [13] the main reason for this interest in error correction is the problem of error propagation. Error propagation is an occurrence present in networks due to the inherent recombination characteristic of Network Coding. A single packet corrupted by an error may continue to be transmitted through the network and contaminate other legitimate packets.

An error correction code is able to correct and detect data packets corrupted due to additive errors and error propagation. Network Error Correction will improve the robustness of the network where the correct information can be obtained, even when only partially correct information is received.

Jaggi et al [9] addresses this problem by adding redundancy to the source information that satisfies certain constraints and achieves optimal rates. This method is illustrated in Figure 1.

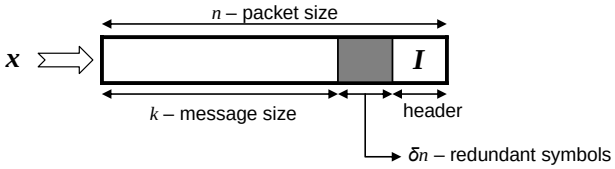


Figure 1: Information Packet with Redundancy symbols

In [14], Yeung and Cai correct errors in Network Coding, not by adding redundancy to each information packet, but by adding redundant packets at the source to be sent over the network. They use (n, k) linear block codes and consider it as a linear network code. The source node takes K information packets as its input and outputs N coded message packets, basically adding packets as parity, as illustrated in Figure 2. This redundancy will enable the receiver node to correct network errors.

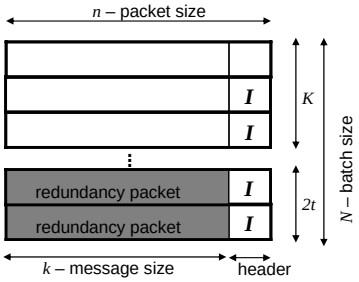


Figure 2: Information packets with redundancy (parity) packets

In [2], the authors concluded their paper by naming that the codes used for network coding are block codes. These codes are ideal types of codes to use, because they operate on packets of a fixed, predetermined size.

The previous examples of error correction methods are in a concatenated form where error correcting encoding takes place prior to transmission. The disadvantage of these schemes is that more than just information packets are injected into the network. The extra information needed for error correction, must be generated at the source, and this redundant information must be sent into the network.

2. IMPLICIT ENCODING IN RANDOM NETWORK CODING

In [15], we presented a scheme where we focus on using the additional channel packets, usually discarded by the receiver node, to apply error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because the redundancy implicitly created in the network provides enough parity information for the receiver to correct errors.

One concern, however, is that the propagation of errors in a network where Random Network Coding is applied will cause the implicit encoding capability of the network to be disrupted.

Our work builds on the results obtained in [15] toward the probability of decoding for implicit error correction methods. Our main contribution is to evaluate the influence of error propagation on the implicit encoding capabilities of networks.

In the existing error correction networks $(min - cut \geq K)$ the receiver node would normally collect as many channel packets as possible. It will, however, only use K channel packets to calculate K linear equations for error correction and discard the additional channel packets.

Our method proposes to use this redundant information obtained by the receiver node to apply error correction to the source message. This means that redundant information transmitted by the source node will no longer be necessary, because suitable networks transmit sufficient redundancy to the receiver for error correction.

Effectively, the proposed method utilizes network using Random Network Coding where the network effectively acts as an error

correction encoder. In these networks, the min- cut between the source and the receiver nodes are large enough to support the transmission of K linear independent information packets in a network ($min - cut \geq K$), although N channel packets will be utilized by the receiver node.

2.1 Implicit Encoding

In the implicit encoding method, K independent information packets are sent from the source nodes where the packets propagate through the network. It is assumed that the global encoding vector is contained in the information packet header, but is negligibly small because the information packets are sufficiently large. Because the header/overhead is negligibly small, $n \approx k$. The construction of the information packets is illustrated in Figure 3.

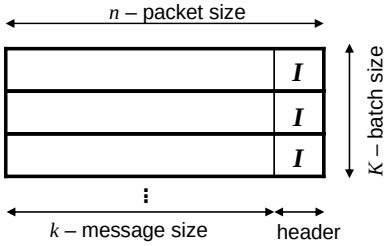


Figure 3: Information packets without redundancy

K Information packets, $x = x_1, x_2, \ldots, x_K$ (vectors with the length of k symbols over finite field F_q), are transmitted by the source node S over the network with a $min - cut \geq K$. No forward error correction is applied to the K information packets and no redundancy is added at the source node.

As the K information packets propagate through the network, linear combinations are formed by the intermediate nodes and the receiver waits until it collects N or more channel packets.

2.2 Decoding

The receiver waits until it obtains N channel packets consisting of two sets of linear independent packets, of size K and $(N - K)$, respectively. The values of N and K are determined by the specific (N, K) linear block code used.

$$y = \sum_{n=1}^{N} \alpha_n x_n = y_{data} + y_{parity}$$

The first set of linear independent packets is the traditional K packets needed to decode the sent message packets

$$y_{data} = \sum_{k=1}^{K} \alpha_k x_k$$

and the other set of linear independent packets is of size $(N - K)$ and is used for error correction by serving as parity symbols. These redundant $(N - K)$ symbols are linear functions of the original K message packets and acts as the parity symbols providing the platform for error detection and correction.

$$y_{parity} = \sum_{k=1+K}^{N} \alpha_k x_k$$

The receiver can decode the message correctly when at most t errors occur, where $t = (N - K)/2$ [24].

As soon as the two sets of linear independent packets are obtained, the same decoding processes are used as the existing methods described in the literature.

2.3 Advantages and Disadvantages

The main advantages of the implicit encoding method are the following [15]:

1. *Saving in bandwidth:* This forward error correction method maps a set of information bits to a set of channel bits resulting in an information rate, R, of less than 1, where $R = \frac{information\ symbols}{channel\ symbols} = \frac{K}{N} \geq 1, K < N.$
2. *No error correction at the source:* The receiver is free to apply any forward error correction code without affecting the source process at all. The receiver can base the choice of code purely on the information collected.
3. *Reduction in network requirements:* The discussed scheme allows for greater error correcting capabilities for a network with a specific min-cut of bandwidth, compared to the other methods proposed in literature, or vice versa.
4. *Energy efficiency:* Less information is injected into the network.

One of the biggest disadvantages concerned with the Implicit Encoding Method is that it is possible that the additional channel packets obtained by the receiver may not be sufficient to act as valid parity symbols. This prevents the receiver from decoding the information successfully and renders the method ineffective.

3. ANALYSIS

We investigated the probability of receiving a set of valid parity symbols from a network, but only in a network where no errors occur. As [16] rightly acknowledged, information packets in real networks are subjected to hostile factors, such as link errors. To determine how this method performs in an environment where errors do occur, the following is investigated:

1. The Probability of decoding for error free and error prone networks in various Galois Fields.
2. Error Propagation by computing the Bit Error Rate (BER) at specific Link error probabilities, P_{le}, in a network and a Binary Symmetrical Channel (BSC) using Reed Solomon (RS) codes in various Galois Fields.

We assume that the network under consideration is a non- cyclic, generic, random network as illustrated in Figure 4. This network contains a single source node $S \in V$, and a single sink node, $t \in V$. The source node sends the K data packets to the network, which consist of p intermediate nodes.

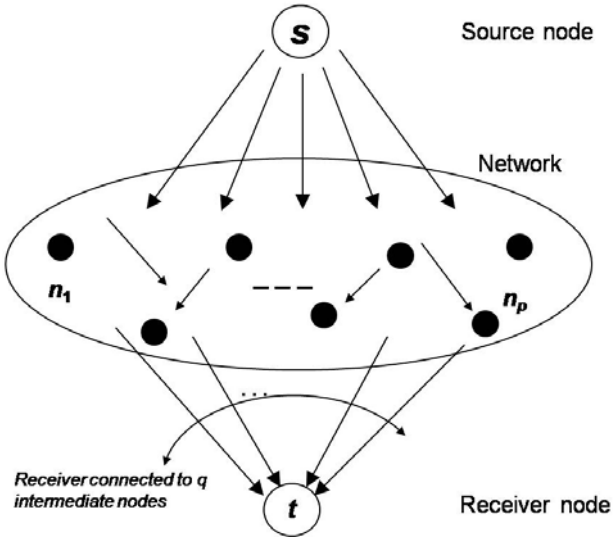


Figure 4: Random Network which contains K source node, p intermediate nodes and a single receiver

The nodes in the network can send multiple encoded packets to other nodes in the network, but only a selection of $q \leq p$ nodes are connected to the receiver. The receiver node only receives a single encoded packet from each of the q nodes.

We generated a set of 1000 randomly generated networks in order to analyze the Implicit Error Correction capabilities of it. These networks have the following properties:

1. Finite field, F_q.
2. Network size, S.
3. $Min - cut \geq K$
4. Intermediate nodes consisting of p nodes, where q is connected to the receiver, $q \leq p$.
5. Link error probability, P_{le}.
6. Each node in the network randomly and independently generates a linear combination of its inputs.

3.1 Probability of Decoding

The probability of decoding in an error free environment was determined for codes in different Galois Fields.

Random networks implementing Random Network Coding are generated for network sizes ranging from N intermediate nodes upward. Although a minimum of K intermediate nodes are needed to obtain a network with connectivity equal to K $(min - cut \geq K)$, at least N intermediate nodes are needed to create the additional $(N - K)$ channel packets needed for error correction.

For each network generation, 100 000 simulation iterations for each parameter set are run. Simulations are repeated until a network size is determined that renders a guaranteed set of parity bits.

The average results are represented in Figure 5 in respect to the mean of each set.

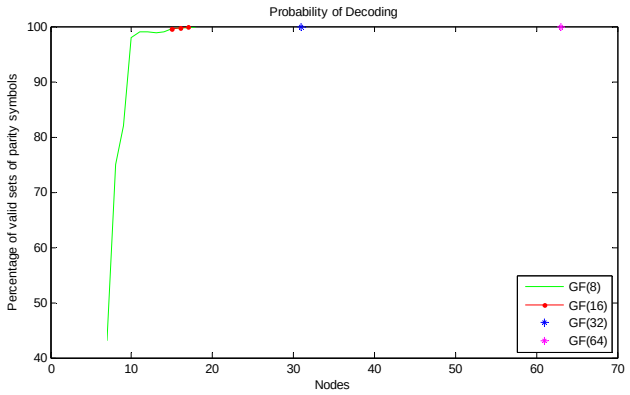


Figure 5: Valid sets of parity symbols received in the Network

The empirical relationship between the network size and the Probability of decoding can be seen in the results presented above. When the block codes are large $GF\ (2^m), m > 4$ the minimum amount of intermediate nodes will guarantee a valid set of parity bits, which gives a guaranteed probability of decoding.

3.2 Error Propagation

The second analysis done on the network is to test the error propagation in the network. In [3], the authors state that the Random Network Coding environment is not error free and that a single error packet has the potential to infect the whole network and corrupt other packets used by the receiver for decoding. When a corrupt packet is linearly combined with legitimate packets, it can corrupt all the information contained in that packet.

Error propagation can be a serious problem in error prone networks where the implicit encoding method is implemented.

To test the error propagation characteristic of the network, another set of networks were constructed. These networks have a $Min - cut \geq K$, where each node in the network randomly and independently generates a linear combination of its inputs. Errors are introduced to the network where each link acts as a binary symmetrical channel (BSC) with a specified link error probability, P_{le}.

Figure 6 shows the BER performance of the implicit encoding networks using a Reed Solomon code with varying P_{le}. The dotted line represents the BER of the minimum size network that renders a guarenteed set of parity bits, while the solid line represents the BER of a very large network (> 20 nodes) also where a valid set of parity bits is guarenteed.

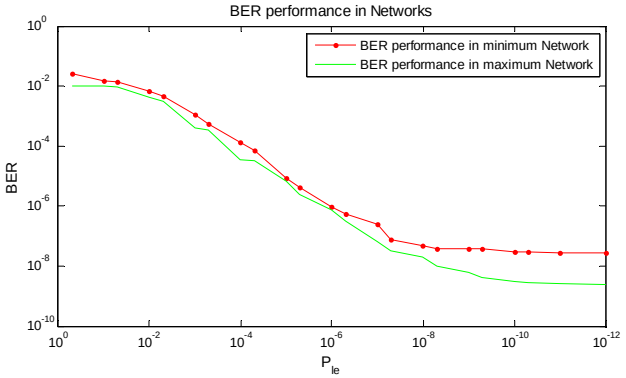


Figure 6: BER performance in the Networks in the presence of link errors

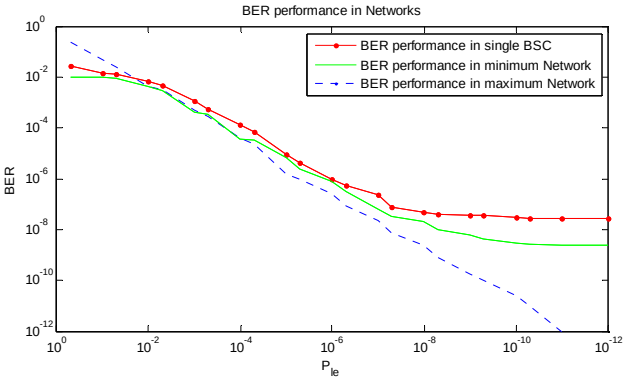


Figure 7: BER performance in the networks and BSC in the presence of link errors

It can be seen that the RS code can correct a large range of errors, but clearly not all. At low P_{le} where hardly any errors occur, all the errors cannot be corrected. This occurrence is due to the fact that errors propagate through the network. Even if a single error occurs, which the RS code is able to correct, it propagates through the network and the receiver obtains channel packets containing more errors than the RS code can correct.

The difference between the BER performances in the two networks, at a low P_{le}, is due to the fact that for the large network, the receiver has the choice of several channel packets to use for decoding. If an error occurred in the network, there is a chance that the receiver node might never use the packet corrupted by the error for decoding. This characteristic of Network Coding in large networks reduces the influence of error propagation on error correction. In the small network, however, the receiver uses all the channel packets collected from the network for error correction.
The reason for the difference in BER performance at a high P_{le} can be a result of the same Network Coding characteristic. In the small network, all the errors that occur and propagate through the network have an influence on the receiver's error correction capability. In the large networks, however, not all the channel packets corrupted by error propagation is used for error correction. The use of a larger network makes the chances of successful error correction much higher.

The problem associated with error propagation can clearly be seen when the same simulation is repeated for a BSC instead of the network, presented as the broken line added in Figure 7. A BSC cannot implicitly encode the data, so the redundancy is added at the source of the channel.

The broken line added to Figure 7 represents a concatenated BSC where several transmissions are necessary between source and receiver. In this BSC, only limited error propagation is possible through the channel: An error can propagate through the channel, but only has an effect on the single channel packet collected by the receiver at the end of the channel, unlike a network where a single error can corrupt a range of channel packets.

It is clear to see that at a low P_{le} the Reed Solomon code is able to correct all the errors occurring in the channel. This is due to the fact that when a single error occurs, the receiver obtains only a single channel packet containing an error and it can be corrected.

At a high P_{le}, the BER of the BSC is higher than the BER of the network. This is due to two reasons:

1. The receiver node of the BSC receives the exact amount of channel packets needed for decoding, unlike in a larger network where the receiver can make a selection from a large amount of channel packets received.

2. The inherent recombination characteristic responsible for error propagation, discussed in Section I, can lead to an advantage in error correction in Network Coding when viewed alongside Binary Symmetrical Channels.
   The intermediate nodes in the network independently and randomly choose a selection of their inputs to encode into an output to transmit. The probability exists that an erroneous packet may not be chosen by the intermediate node for recombination. This action will stop the error from propagating and reaching the receiver. In a BSC, the received data is simply transmitted through the channel where an error simply propagates until it reaches the receiver.

It is evident that a network which implements Random Network Coding is sensitive to errors and the propagation thereof. The propagation of errors reduces the probability of the network to produce valid sets of parity bits that make the decoding possible.

4. CONCLUSION

Encoding the information within the network, instead of transmitting the already encoded codeword over the network leads

to advantages such as a saving in network bandwidth, reduction in network requirements and network size. The source node does not implement error correction and the receiver node can apply any error correction code it chooses based on the information it receives.

However, utilizing the implicit encoding capabilities of a network makes the network very sensitive to errors and the propagation thereof.

The work we presented flows from the work done in [15] and evaluates the Implicit Encoding Scheme in the presence of link errors and its propagation. Error propagation has a major influence on networks utilizing Random Network Coding, because a single error can corrupt a range of channel packets needed by the receiver for decoding.

The effect of error propagation is evaluated by comparing the network utilizing Random Network Coding to a simple Binary Symmetrical Channel where errors cannot propagate. The results show that error propagation has an effect on error correction. When the error probability in the network is low and only a single occur, the receiver may be unable to successfully apply error correction to obtain the sent data. This is due to the fact that a single error can cause the corruption of several channel packets used by the receiver.

In larger networks, however, the effect of error propagation in networks using Random Network Coding is reduced due to the fact that the receiver does not use all the channel packets collected from the network to apply error correction. In a network where only a set of channel packets are utilized by the receiver, the BER is much lower, because some of the infected packets are discarded.

5. REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000.

[2] N. Cai and R. W. Yeung, “Network coding and error correction,” Proceedings of IEEE Information Theory Workshop, pages 119-122, October 2002.

[3] [15] N. Cai and R. W. Yeung, “Network error correction, part I: Basic concepts and upper bounds,” Commun. Inform. Syst.,vol. 6, no. 1, pp. 19–36, 2006.

[4] [16] N. Cai and R. W. Yeung, “Network error correction, part II: Lower bounds,” Commun. Inform. Syst., vol. 6, no. 1, pp.37–54, 2006.

[5] C. Fragouli, J. Le Boudec, and J. Widmer, “Network coding: an instant primer,” SIGCOMM Comput. Commun. Rev., 36(1):63–68, 2006.

[6] C. Fragouli and Emina Soljanin, “Network Coding Fundamentals,” Foundations and Trends in Networking, Vol. 2, No. 1, 2007

[7] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in Proc. IEEE Int. Symp. Information Theory, Yokohama, 29 June-4 July 2003, p. 442.

[8] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, “Toward a random operation of networks,” submitted to IEEE Trans. Inform. Theory, 2004

[9] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, “Resilient network coding in the presence of Byzantine adversaries,” in Proc. 26th IEEE Int. Conf. on Computer Commun., Anchorage, AK, May 2007, pp. 616–624.

[10] R. Koetter and M. Médard, “Beyond routing: An algebraic approach to network coding,” IEEE/ACM Transaction on Networking, vol. 11, pp. 782–796, October 2003.

[11] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", in International Symposium on Information Theory (ISIT), June 2007.

[12] D. Silva, F.R. Kschischang, and Ralf Koetter “A Rank-Metric approach to error control in random network coding,” IEEE International Symposium on Information Theory, Nice, France, June 2007

[13] D. Silva, F. R. Kschischang: On Metrics for Error Correction in Network Coding CoRR abs/0805.3824: (2008)

[14] R.W. Yeung and N Cai, “Network Coding, Algebraic Coding, and Network Error Correction”, in Proc. Information Theory and Applications Workshop, La Jolla, CA, February 2006

[15] S. von Solms and A.S.J Helberg, “Error Correction with the Implicit Encoding Capability of Random Network Coding,” in Proc. SATNAC, Swaziland, August 2009

[16] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2003.

ABSTRACT: Poster

Performance of Implicit Error Correction of Network Coding networks in the presence of Link Errors

S von Solms, ASJ Helberg

School for Electric Electronic and Computer Engineering

North West University, Potchefstroom Campus

27 (0)18 299 1961

sune.vonsolms@nwu.ac.za

We investigate the performance of the Implicit Error Correcting scheme applied to networks utilizing Random Network Coding.

Existing Network Error Correction schemes applies error correction by sending redundant information over the network that acts as parity for error correction. In the existing error correction networks ($min - cut \geq n$) the receiver node would normally collect as many channel packets as possible. It will, however, only use n channel packets to calculate the sent massage and correct possible errors. The receiver will discard the additional channel packets, because it is redundant.

The introduced scheme does not add redundancy to the data prior to transmission, like existing error correcting schemes. It exploits the implicit encoding capabilities of the network in order to create enough redundancy for successful error correction. A large network using Random Network Coding generates enough redundant information in order to perform error correction on transmitted data. For this scheme, the additional channel packets obtained by the receiver are not discarded, but used as the parity packets needed for error correction. This means that redundant information transmitted by the source node will no longer be necessary, because suitable networks transmit sufficient redundancy to the receiver for error correction.

Effectively, the proposed method utilizes network using Random Network Coding where the network effectively acts as an error correction encoder. In these networks, the min- cut between the source and the receiver nodes are large enough to support the transmission of k linear independent information packets in a ($min - cut \geq n$), although k channel packets will be utilized by the receiver node. A Random Network Coding environment is not necessarily error free. Error propagation in networks is a major weakness in Random Network Coding and has an influence on the performance of the Implicit Error Correction ability of networks.

Firstly, we introduce the Implicit Error Correction capabilities of networks using Random Network Coding and how this capability can be used for error correction. Secondly, the performance and reliability of Implicit Error Correction in the presence of errors is evaluated, as well as the effect of error propagation on the scheme.

General Terms

Performance, Design, Reliability

Keywords

Error Correction, Error Propagation, Network Coding, Random Network Coding

Performance of Implicit Error Correction of Network Coding networks using Random Network Coding

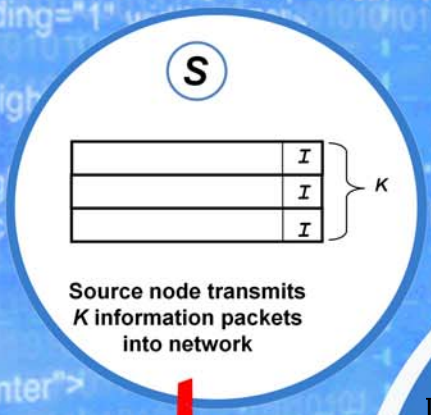
S von Solms and ASJ Helberg
School for Elecric, Electronic and Computer Engineering
North-West University, Potchefstroom Campus

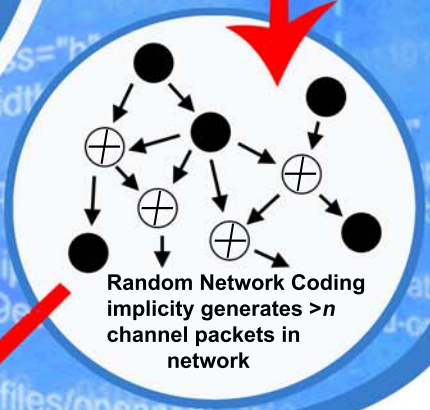
Introduction

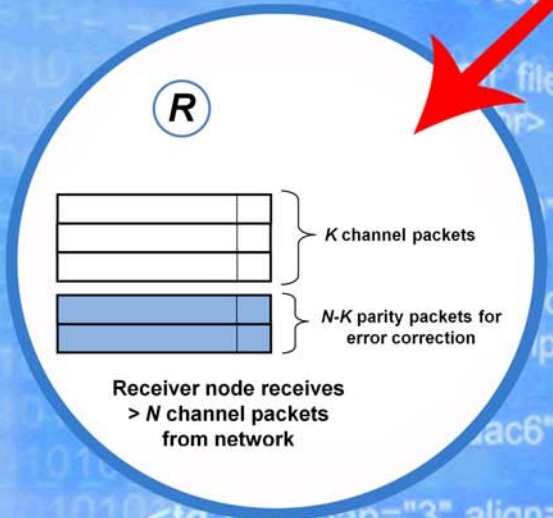
This Network Error Correcting scheme exploits the implicit encoding capabilities of the network in order to create enough redundancy for successful error correction. It does not add redundancy(parity) to the data prior to transmission like existing error correcting schemes.

In Random Network Coding networks, the nodes independently and randomly select linear mappings from its inputs and create independent non-zero linear combinations that are then forwarded to the next node.

Random Network Coding environments are not error free. Error propagation in networks has an influence on the performance of the Implicit Error Correction ability of networks.







Problem Statement

Evaluating the Performance of the Implicit Error Correction Scheme in error-prone networks.

Objectives

Implement the Network Error Correction methods in Random Network Coding network.

Compare the performance of the Implicit Error Correction Scheme to that of Existing schemes.

Methodology

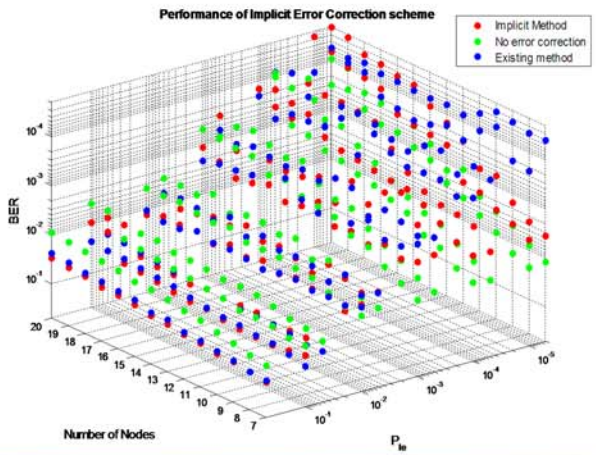
Simulate Network Error Correction schemes in a Random Network Coding network environment in MATLAB.

Compare the BER performance of the schemes at specific Link Error Probabilities (P_{le}) and network size(N).

Results

For networks of size $n \geq 15$ and link error probability $P_{le} \leq 10^{-5}$, the proposed method renders a smaller BER.

This is due to the fact that the network becomes large enough to generate enough valid parity packets so that successful error correction can be implemented. At a specific P_{le}, the proposed scheme has greater error correcting capabilities compared to the other error correction method.



Acknowledgments

This research is financially supported by Telkom SAAB Grintek COE and the NRF.

S von Solms and ASJ Helberg
TeleNet Research group
School for Elecric, Electronic and Computer Engineering
North-West University, Potchefstroom Campus
Tel: (018) 299 1961, Fax: (018) 299 1977
sune.vonsolms@nwu.ac.za

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT