

Collaborative Coding Techniques with Analog Network Coding  
in Wireless Y-Channel-Relay Networks

Issa Al-Fanek

A Thesis  
in  
The Department  
of  
Electrical & Computer Engineering

Presented in Partial Fulfillment of the Requirements  
for the Degree of Master of Applied Science (Electrical Engineering) at  
Concordia University  
Montreal, Quebec, Canada

August 2014

© Issa Al-Fanek, 2014

**CONCORDIA UNIVERSITY**

**School of Graduate Studies**

This is to certify that the thesis prepared

By: Issa Al-Fanek

Entitled Collaborative Coding Techniques with Analog Network Coding in  
Wireless Y-Channel-Relay Networks

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

**Master of Applied Science (Electrical Engineering)**

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

Signed by the final Examining Committee:

	_____ Dr. M. Z. Kabir _____	Chair
	_____ Dr. Y. R. Shayan _____	Examiner
	_____ Dr. H. Harutyunyan _____	Examiner
	_____ Dr. M. R. Soleymani _____	Supervisor

Approved by \_\_\_\_\_

Chair of Department or Graduate Program Director

\_\_\_\_\_ 2014

\_\_\_\_\_  
Dean of Faculty

# Abstract

## Collaborative Coding Techniques with Analog Network Coding in Wireless Y-Channel-Relay Networks

Issa Al-Fanek

After channel coding reached near Shannon-limit performance with the introduction of Turbo codes and Low-Density Parity-Check (LDPC) codes, research moved on to network coding techniques to enhance overall network performance. The most recent and novel of those approaches is the idea of Physical (Analog) Network Coding which embraces interference from other users, mixes signals in the channel rather than in a relay, and can theoretically increase throughput in the two-way relay channel by up to two folds. In this thesis, we explore this idea, and analyze the theoretical gains of using network coding in a Y-Channel problem - where three users communicate through a common relay. We study existing collaborative coding techniques for the Y-Channel like nested recursive convolutional codes, and Combined Network Channel (CNC). After that, we introduce enhanced nested codes based on turbo codes that achieve good performance in poor SINR conditions. In addition, we propose a novel equal-rate collaborative coding scheme based on algebraic linear block codes. This scheme is simpler to implement than CNC, yet its burst-traffic performance is better than any of the studied solutions. In theory, this code reduces the number of transmission timeslots by up to three folds. Finally, we put forward practical scenarios where physical network coding can be harnessed – mainly in Long Term Evolution (LTE) Multicast (eMBMS), and opportunistic routing in Wireless Mesh Networks.

## **Acknowledgements**

This thesis wouldn't have been possible without the support and guidance from many people.

First of all, I would like to express my deepest sense of gratitude to my supervisor, Dr. M. Reza Soleymani who always offered his guidance and advice whenever I needed it during the course of developing this research and writing this thesis.

I would also like to thank my brother, Basel, and my sister, Rund and her delightful family for their continuous support and encouragement. A very special thank you goes to my nephew, Sammy, whose spectacular smile always managed to lift my spirit up.

A big thank you goes to all my friends and co-workers who stood by me and encouraged me throughout this journey.

Last but certainly not least, I would like to thank the two closest persons to my heart, my lovely parents, who always believed in me, and who continuously provided me with their love, care and support in whatever I set out to do.

# Table of Contents

<b>List of Figures.....</b>	<b>viii</b>
<b>List of Tables .....</b>	<b>xiii</b>
<b>List of Abbreviations .....</b>	<b>xvi</b>
<b>Chapter 1: Introduction .....</b>	<b>1</b>
1.1. The problem statement.....	3
1.2. Organization of the thesis .....	7
<b>Chapter 2: Background and Literature Review .....</b>	<b>8</b>
2.1. Wireless Mesh Networks .....	8
2.1.1. Properties of Wireless Mesh Networks .....	9
2.1.2. Applications of Wireless Mesh Networks.....	10
2.1.3. Opportunistic Routing in Wireless Mesh Networks.....	10
2.2. Relaying & Network Coding .....	11
2.2.1. Two-Way-Relay communication .....	11
2.2.2. Digital Network Coding.....	13
2.2.3. Analog (Physical) Network Coding .....	18
2.3. Y-Channel-Relay Communication.....	20
2.3.1. Y-Channel Two-Sender Topology .....	22
2.3.2. Y-Channel Three-Sender Topology .....	30
2.4. The Hybrid Error-Erasure Binary Channel.....	39
2.4.1. Capacity in Two-Way-Relay Scenario.....	41
2.4.2. Capacity in Two-Sender Y-Channel-Relay Scenario with One Common Receiver .....	42
2.4.3. Capacity in Three-Sender Y-Channel-Relay Scenarios .....	45
2.4.4. Y-Channel-Relay Analysis without Coding.....	49
2.5. Existing Coding Techniques for The Y-Channel-Relay Problem .....	56
2.5.1. Nested Codes.....	56
2.5.2. Combined Network Channel (CNC) Coding .....	60
<b>Chapter 3: Proposed Coding Techniques for the Y-Channel-Relay Network.....</b>	<b>66</b>
3.1. Nested Recursive Convolutional Codes Solution .....	66

3.1.1.	Encoding .....	66
3.1.2.	Decoding .....	68
3.1.3.	Simulation .....	73
3.2.	Nested Turbo Codes Based Solution .....	77
3.2.1.	Encoding .....	77
3.2.2.	Decoding .....	79
3.2.3.	Simulation Results .....	83
3.3.	Collaborative Algebraic Linear Block Codes Solution .....	87
3.3.1.	The Decoding Process.....	89
3.3.2.	Algorithm for Decoding Using Array Manipulation.....	99
3.3.3.	Algorithm for Decoding Using Table Lookup .....	107
3.3.4.	Theoretical Limit of Performance of Algebraic Linear Block Codes in a Noiseless Channel (Erasures-Only).....	121
3.3.5.	Simulation of the (7,4) Algebraic Linear Block Code Example .....	125
3.3.6.	Effect of Increasing Rate of the Code .....	126
3.4.	Comparison of The Different Coding Techniques.....	130
<b>Chapter 4: Applications to Modern Wireless Networks .....</b>		<b>132</b>
4.1.	LTE Multimedia Broadcast/Multicast Service (eMBMS).....	132
4.2.	Use in Wireless Mesh Network .....	134
4.3.	In Backbone of Wireless Networks .....	135
<b>Chapter 5: The Mesh Network Simulator .....</b>		<b>137</b>
5.1.	Design Objectives .....	138
5.2.	Software Architecture .....	138
<b>Chapter 6: Conclusions .....</b>		<b>141</b>
6.1.	Future Research .....	142
<b>Appendix A: Channel Coding Techniques .....</b>		<b>150</b>
A.1.	Linear block codes .....	150
A.1.1.	Hamming codes.....	151
A.1.2.	Bose-Chaudhuri-Hocquenghem (BCH) codes .....	151
A.2.	Convolutional (Trellis) Codes.....	152
A.2.1.	Description .....	152

A.2.2. Recursive Convolutional Codes .....	152
A.2.3. Encoding.....	153
A.2.4. Decoding .....	154
A.3. Turbo Codes.....	159
A.3.1. Encoding.....	159
A.3.2. Iterative Decoding.....	162
A.4. Concatenated Codes.....	163
A.5. LDPC Codes .....	164
<b>Appendix B: The Mesh Network Simulator Interface .....</b>	<b>165</b>
B.1. The Home Menu.....	165
B.2. The Main Interface .....	165
B.3. The Options Dialog.....	167
B.4. Link (Channel) Options Dialog.....	168
B.5. The Node Properties Dialog.....	169

## List of Figures

Figure 1: The Y-Channel-Relay problem .....	4
Figure 2: Two-Sender vs. Three-Sender networks.....	5
Figure 3: Y-Channel-Relay extreme scenario with 3 different packets sent to six receivers .....	6
Figure 4: Mesh Network with a gateway .....	8
Figure 5: Two-Way-Relay problem.....	11
Figure 6: Two-way communication with a relay in the middle.....	12
Figure 7: Butterfly Network Topology .....	14
Figure 8: Transmission flow diagram in a butterfly network without network coding .....	14
Figure 9: Transmission flow diagram in a butterfly network with network coding .....	15
Figure 10: Using network coding for large scale content distribution.....	17
Figure 11: Two-way communication with relay and digital network coding.....	18
Figure 12: Two-way communication with relay and analog network coding .....	19
Figure 13: An example of a Three-Node communication through a relay in a Wireless Mesh Network.....	21
Figure 14: The Y-Channel-Relay problem .....	22
Figure 15: A Y-Channel-Relay problem in a Two-Sender topology. Example 1. ....	23
Figure 16: Transmission in Two-Sender network (example 1) without the use of network coding .....	24
Figure 17: Decision regions at receiver in a Two-Sender Y-Channel-Relay scenario (non adaptive (left) vs. adaptive (right) power transmissions).....	26



Figure 18: Transmission flow in a Two-Sender Y-Channel-Relay problem (example 1). Three transmissions are required.....	27
Figure 19: Transmission flow in a Two-Sender Y-Channel-Relay problem with physical network coding. Only two transmissions are required.....	27
Figure 20: Two-Sender Y-Channel-Relay problem (example 2). .....	28
Figure 21: Transmission flow in Two-Sender Y-Channel-Relay problem (example two). Four transmission slots are required.....	28
Figure 22: Transmission flow in Two-Sender Y-Channel-Relay topology with Digital Network Coding (example 2). Three transmissions are required. ....	29
Figure 23: Transmission flow in Two-Sender Y-Channel-Relay topology with Analog (Physical Network Coding) (example 2). Only two transmissions are required. ....	30
Figure 24: Three-Sender Y-Channel-Relay scenario (example 1). ....	31
Figure 25: Transmission flow diagram for Three-Sender Y-Channel-Relay scenario (example 1). .....	31
Figure 26: Transmission flow diagram Three-Sender Y-Channel-Relay Scenario with Digital Network Coding (example 1).....	34
Figure 27: Transmission flow diagram for O=Shape Y-Channel-Relay Scenario with Analog (Physical) Network Coding (example 1) .....	35
Figure 28: Three-Sender Y-Channel-Relay scenario (example 2) .....	35
Figure 29: The hybrid error-erasure binary channel.....	40
Figure 30: Transmission and decoding in a Two-Sender Y-Channel-Relay scenario with one common receiver.....	43
Figure 31: Transmission and decoding in the maximum configuration of a Two-Sender network with two sender and three receivers.....	44
Figure 32: Transmission and decoding in Three-Sender Y-Channel-Relay scenario.....	46

Figure 33: Decision regions on the receiver side .....	50
Figure 34: Theoretical BER performance curve of the uncoded solution .....	54
Figure 35: The simulated BER curve for the uncoded solution.....	56
Figure 36: Packet Error Rate when using nested codes in Two-Sender Network with two senders and a common receiver .....	59
Figure 37: Throughput (information bits/sec) when using Nested Codes in Two-Sender Y- Channel-Relay problem with two senders and one common receiver.....	59
Figure 38: Throughput (information bits/sec) when using Nested Codes in Three-Sender Y- Channel-Relay problem and each sender needs to send the same packet to the other two users. ....	60
Figure 39: System throughput (info bits/sec) for full transmission scheme in an Three-Sender Network using CNC.....	64
Figure 40: Recursive Convolutional Codes based solution to the Y-Channel-Relay with ANC problem .....	67
Figure 41: Recursive Convolutional Encoder of Node 1 .....	67
Figure 42: Recursive Convolutional Encoder of Node 2.....	67
Figure 43: Recursive Convolutional Encoder of Node 3.....	68
Figure 44: Iterative Decoder for the Recursive Convolutional Codes based solution.....	69
Figure 45: The Decision Regions for estimating erasures .....	70
Figure 46: Simulated BER performance of the RSC based solution example with $K=7$ .....	74
Figure 47: RSC encoder of Node 1 with $K=5$ .....	75
Figure 48: RSC encoder of Node 2 with $K=5$ .....	75
Figure 49: RSC encoder of Node 3 with $K=5$ .....	75

Figure 50: Effect of used RSC constraint length on BER performance .....	76
Figure 51: Turbo encoder of node 1 .....	78
Figure 52: Turbo encoder of node 2 .....	78
Figure 53: Turbo encoder of node 3 .....	79
Figure 54: High-level Turbo based solution decoder .....	80
Figure 55: Low level turbo-codes based solution decoder .....	80
Figure 56: Decision Region used to estimate erasures in Turbo based Nested Codes solution ...	81
Figure 57: Simulated BER performance of Turbo based solution.....	85
Figure 58: Comparison in BER performance between Turbo and RSC based solutions .....	86
Figure 59: Bubble notation for the Collaborative Hamming Code encoder used in each terminal .....	88
Figure 60: Theoretical performance limits of an algebraic linear block codes solution with (n=200).....	124
Figure 61: The simulated BER curve of the (7,4) Hamming code solution .....	126
Figure 62: Increasing the length of the code lowers the error floor of guessing erasures .....	127
Figure 63: SNR versus Information Bit Throughput in Y-Channel-Relay Scenario.....	129
Figure 64: System throughput performance proposed solutions and existing solutions .....	130
Figure 65: eMBMS deployment example.....	133
Figure 66: Example of a carrier configuration when ANC and algebraic linear block codes are used .....	134
Figure 67: A Mesh Network .....	134

Figure 68: The proposed practical system for VOIP applications using Network Coding .....	136
Figure 69: The IF-Mesh network simulation app .....	137
Figure 70: High-Level Software Architecture of the IF Mesh Network Simulator .....	139
Figure 71: Convolutional Encoder example .....	153
Figure 72: Turbo Encoder .....	160
Figure 73: Turbo Decoder .....	163
Figure 74: A communication system with concatenated codes .....	164
Figure 75: The Home Menu (Hub) of the IF Mesh Network Simulator .....	165
Figure 76: The Main Interface of IF Mesh Network Simulator .....	166
Figure 77: The Sliding Menu (Drawer) .....	167
Figure 78: Options Dialog .....	168
Figure 79: Link (Channel) Options Dialog .....	169
Figure 80: Node Properties Dialog .....	170

## List of Tables

Table 1: Possible binary outcomes of the two-way communication with relay and digital network coding.....	18
Table 2: Received signal at Node 3 when using digital network coding in a Two-Sender Y-Channel-Relay scenario .....	25
Table 3: Forwarded signal from the relay when using network coding in a Two-Sender Y-Channel-Relay scenario .....	26
Table 4: Transmitted signal of relay when using network coding in a Three-Sender Y-Channel-Relay scenario.....	32
Table 5: Transmitted signal of relay when using network coding in a Three-Sender Y-Channel-Relay scenario and adapted power scheme to match the power of a transmitting node.....	33
Table 6: Summary of the properties and scenarios of Three-Sender and Two-Sender Y-Channel-Relay networks.....	36
Table 7: In depth look at all possible scenarios involving three terminals and a relay .....	37
Table 8: Capacity of Two-Way-Relay Scenario.....	46
Table 9: Capacity of Two-Sender Y-Channel-Relay Scenarios .....	47
Table 10: Capacity of Three-Sender Y-Channel-Relay Scenarios .....	48
Table 11: Message of node 1 .....	50
Table 12: Message of node 2 .....	50
Table 13: Message of node 3 .....	50
Table 14: Received signal at relay.....	51
Table 15: Transmission Flow Diagram in an Three-Sender Y-Channel-Relay network using Combined Network Channel (CNC) coding.....	61

Table 16: Encoding table for collaborative linear block code solution example.....	89
Table 17: Node 1 message .....	90
Table 18: Node 2 message .....	90
Table 19: Node 3 message .....	90
Table 20: Received mixed signal at relay .....	90
Table 21: Received mixed signal at Node 1 after deducting its message.....	91
Table 22: Initial estimation of Node 2 message at Node 1 .....	91
Table 23: Initial estimation of Node 3 message at Node 1 .....	91
Table 24: Node 2 decoded message at Node 1 .....	93
Table 25: Node 3 decoded message at Node 1 .....	93
Table 26: Received mixed signal at Node 2 after deducting its message.....	93
Table 27: Initial estimation of Node 1 message at Node 2 .....	93
Table 28: Initial estimation of Node 3 message at Node 2.....	94
Table 29: Node 1 decoded message at Node 2 .....	96
Table 30: Node 3 decoded message at Node 2 .....	96
Table 31: The recieved mixed signal at node 3 after deducting its message.....	96
Table 32: Initial estimation of Node 1 message at Node 3 .....	96
Table 33: Initial estimation of Node 2 message at Node 3 .....	97
Table 34: Node 1 decoded message at Node 3 .....	99
Table 35: Node 2 decoded message at Node 3 .....	99

Table 36: Decoding lookup table at Node 1 .....	108
Table 37: Decoding lookup table at Node 2 .....	112
Table 38: Decoding lookup table at Node 3 .....	116
Table 39: Theoretical limits for different information bit sequence lengths, and different algebraic linear block code rate .....	124
Table 40: Interleaver Input.....	161
Table 41: Interleaver Output.....	161

## List of Abbreviations

ANC	Analog Network Coding
BER	Bit Error Rate
CNC	Combined Network Channel
DNC	Digital Network Coding
DoF	Degree of Freedom
eMBMS	Evolved Multimedia Broadcast Multicast System
LDPC	Low-Density Parity-Code
LTE	Long Term Evolution
MAC	Medium Access Control
OSI	Open System Intercommunication model
PNC	Physical Network Coding
RSC	Recursive Systematic Convolutional Code
SINR	Signal to Interference & Noise Ratio
SISO	Soft-Input Soft-Output
SNR	Signal to Noise Ratio
WMN	Wireless Mesh Network



# Chapter 1: Introduction

As channel-coding techniques on the physical layer reached near Shannon-limit performance, after the introduction of robust practical solutions like Turbo Codes and Low-Density Parity-Check (LDPC) codes, the research community shifted their attention to increasing the performance of overall network throughput. This meant blurring the boundaries between the physical layer and higher layers of the OSI model: the Data Link and Network layers. One of the outcomes of this was the introduction of Network Coding.

Network Coding procedures rely on intuitively mixing received packets at the relay using modulo-add operation, and later forwarding the resulting packet to the next hop rather than transmitting each packet individually. The recipient extracts information based on his partial a priori knowledge of the data. Hence, reducing the number of required transmissions in the network, and increasing overall network throughput. This proved highly efficient in point-to-multipoint (multicast) applications.

Traditionally, concurrent transmissions from multiple users were always considered harmful interference that needs to be avoided. Multiple-Access procedures in the Data Link layer of the OSI model are in place to prevent that from happening. That is until Physical/Analog Network Coding (PNC) emerged - a novel idea that embraces wireless interference by allowing multiple users to transmit simultaneously, assuming that the recipient receives the relative bits from each sender at roughly the same time, and the received power after path loss from each sender is roughly equal. The result at the receiver is an analog signal that combines the concurrent transmissions over the air. That is equivalent to mixing packets at the relay with Digital Network Coding, albeit achieved with lower number of transmissions. This idea was conceived in 2008 and was limited to Two-Way-Relay networks. In this scenario, ANC reduces the number of required transmissions from four to two when compared to traditional routing methods, and from three to two when compared to classical network coding. A few practical solutions that add the capability of using PNC in Wireless Mesh Networks with opportunistic routing were developed.

One of these systems, named MIXIT, was designed and tested in MIT. They saw significant improvement in overall network throughput when compared to traditional networks.

The discussion in literature expanded to the study of the feasibility and gains of using PNC in more complicated scenarios like the Y-Channel-Relay problem – where three users communicate through a common relay. In this case, PNC reduces the number of required transmission slots from six when using traditional routing to only two, given the assumption that each receiver has the ability to separate the combined received signal, and extract data belonging to the two other users. The complexity of this approach is that after deducting its information sequence, the receiver ends up with a packet that is a result of adding the relative bits of the other two users. This introduces erasures into the equation, and thereby, transforming the problem to a separate Hybrid Erasure-Error Channel at each receiver. Therefore, requiring collaborative coding schemes to complete the solution. This means that while the number of required transmissions drops by three folds, the actual throughput of the network may not increase by that factor due to the added redundancies from coding.

There exist a few practical collaborative coding schemes tailored to the Y-Channel-Relay with PNC problem. We will explore two of the top performing solutions in this thesis; the first is based off the idea of Nested Codes. In this approach, each sender uses a unique RSC channel encoder. Upon the reception of the mixed signal from the relay, each recipient decodes the received sequence iteratively by first calculating the extrinsic value for each bit using a SISO decoder based on the encoder of the first user. Then, the extrinsic information sequence is traversed and treated via a flipping operation based on whether the bit is supposedly erased. The process is repeated for a number of iterations before a soft decision is made.

The second idea we study is called Combined Network Coding (CNC). In this approach, two users at any given time transmit at full channel rate, while the third transmits at half channel rate. The users take turns in a cyclic fashion transmitting at half-rate while the other two users transmit at full rate. The best performance of this technique has a degree of freedom equal to two and it is realized when each sender has a continuous stream of data to transmit. As of this

writing, CNC is the top performing technique for solving the Y-Channel-Relay with PNC problem.

In this thesis, we will explore network coding and analyze the Y-Channel-Relay problem. Then, we propose an enhancement to the Nested Codes solution by nesting turbo codes rather than RSC codes. We will present simulation results showing increased performance in low SINR region when compared to RSC Nested Codes.

After that, we introduce an equal rate algebraic linear block codes based solution where each sender uses a linearly independent generator matrix to encode its information bits. Once a user receives a mixed packet, it applies algebraic procedures to decode at the receiver. We show that we can have more than rate half code per node while maintaining zero Bit-Error-Rate. Even though this novel scheme is simpler to implement in practice relative to CNC, it achieves competitive performance. Moreover, this method applies to any of the transmission scenarios in a Y-Channel-Relay problem, and performs well with bursty traffic patterns, while CNC works best when users have data to transmit at all times.

At the end, we propose a practical solution to reduce the bandwidth requirement in certain deployment scenarios of the LTE broadcast standard – Multimedia Broadcast Multicast Service (eMBMS). The proposed mechanism reduces used bandwidth by up to 15%. Furthermore, we recommend enhancements to the MIT MIXIT system by adding opportunistic routing techniques that recognize Y-Channel-Relay scenarios. Thus, improving overall throughput of Wireless Mesh Networks by a large margin.

In the following section, we define the Y-Channel-Relay model used throughout this thesis.

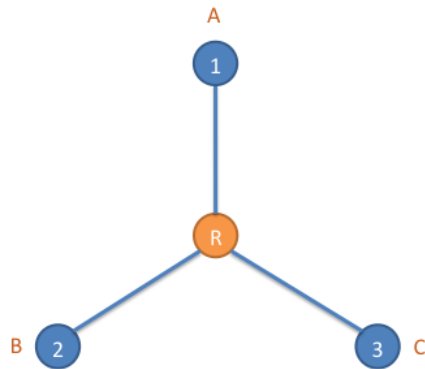
## **1.1. The problem statement**

This research looks in-depth into the scenario where three terminals communicate through a common relay in a wireless network. This situation is not uncommon in wireless mesh networks

with opportunistic routing, or wireless multicast applications such as LTE eMBMS (Multimedia Broadcast Multicast Service).

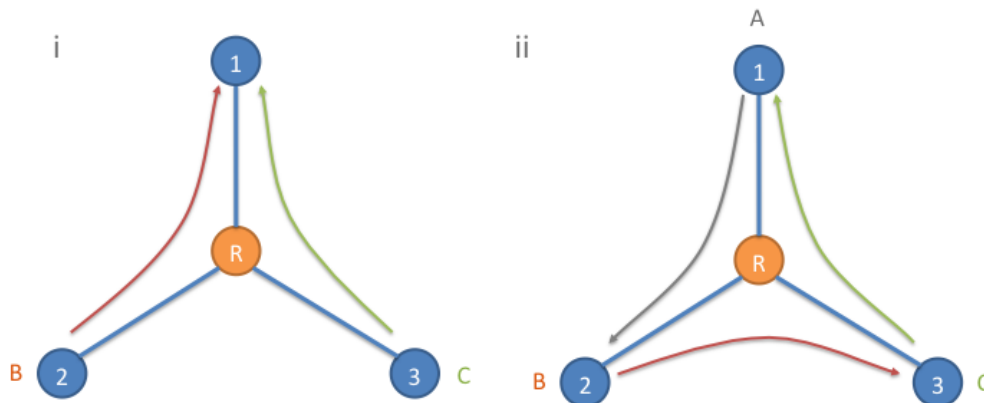
Figure 1 shows such a network. Throughout this thesis, we shall refer to this topology as the *Y-Channel-Relay problem*, and we make the following assumptions:

- *Packet A, Packet B, and Packet C are always associated with Node 1, Node 2, and Node 3 respectively.*
- *Packet A, Packet B and Packet C have the same length of  $N$  bits.*
- For the sake of simplicity, we assume that communication between the sending nodes is only possible through the use of the relay. I.e. none of the sending nodes can overhear or receive any partial side-information when others transmit to the relay. As such, the results presented in this thesis are considered worst-case analysis. Theoretically, having partial information from previous transmissions enhances the performance considerably.
- A node uses the whole bandwidth available in the channel when transmitting a packet. FDMA (Frequency Division Multiple Access) is not an option. Also the channel is assumed to be half-duplex.
- The received power at the relay from each user after path loss is equal. In other words, all users transmit at the same power, and the distance between each user and the relay is equal.



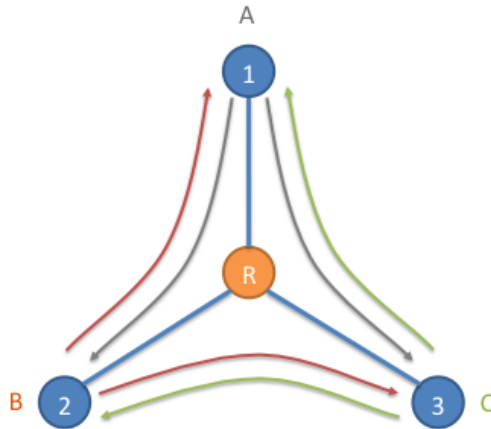
**Figure 1: The Y-Channel-Relay problem**

There are multiple transmission scenarios that might arise when two or three terminals need to send packets in a Y-Channel-Relay problem. We will touch on them briefly here, and discuss them in more detail in Section 2.2. Those situations are categorized into two groups; Two-Sender and Three-Sender Y-Channel networks (Figure 2).



**Figure 2: Two-Sender vs. Three-Sender networks**

As the name suggests, in Two-Sender scenarios, two terminals need to send to either a common recipient or multiple recipients. An example is shown in Figure 2-i where Node 2 and Node 3 want to send to the same receiver (Node 1). In Three-Sender scenarios, on the other hand, each node has a packet to be sent to either one or two receivers. The total number of packets at the end of the transmission here will vary between three in the simple case of Node 1 wants to send to Node 2, Node 2 wants to send to Node 3, and Node 3 wants to send to Node 1 (Figure 2-ii), to the extreme situation of six packets when Node 1 wants to send Packet A to Node 2 and Node 3, Node 2 wants to send Packet B to Node 1 and Node 3, and Node 3 wants to send Packet C to Node 1 and Node 2. This extreme situation is described in Figure 3.



**Figure 3: Y-Channel-Relay extreme scenario with 3 different packets sent to six receivers**

Traditionally, transmission in wireless networks relied on Medium-Access-Control procedures to prevent collisions, and prevent multiple users from accessing the wireless medium concurrently. Transmission from one user was always perceived as unwanted interference to other users. In recent years, however, there has been a lot of research around digital network coding schemes to mix packets from different nodes to reduce the number of required transmissions in relayed networks. [1] [2] Moreover, physical (analog) network coding schemes were introduced that embrace the nature of the wireless medium, and allow multiple users to send at the same time, and thus, harnessing what was classically looked at as interference and mix packets in the air rather than at the relay. [3] [4] [5] [6] This allowed for theoretically more efficient networks, and lower number of overall transmissions. For example, it takes four transmission slots to complete the communication scenario posed in a Two-Sender network using MAC procedures. However, it takes only two transmissions to send those packets when physical network coding is used.

In this thesis, we will explore existing collaborative coding schemes with physical network coding in different Y-Channel-Relay scenarios. In addition, a novel equal-rate coding scheme that can work in virtually any Y-Channel-Relay scenario is proposed. Its capabilities reduce the number of transmissions in a wireless mesh network by up to three folds.

## **1.2. Organization of the thesis**

In Chapter 2, we start with a literature review where we discuss Wireless Mesh Networks and the physical attributes of the wireless channel that can be harnessed for gains in network performance through physical network coding. Then, we explore relaying and network coding both in its digital and physical flavors. After that, we look at existing solutions for the Y-Channel-Relay with ANC problem. In Chapter 3, we propose several coding schemes to solve the general case of Y-Channel-Relay transmission. In Chapter 4, we propose practical solutions for PNC in real world deployments such as LTE eMBMS and Wireless Mesh Networks with Opportunistic routing. In Chapter 5, we take a brief look at the Wireless Mesh Network simulator developed during the development of this research. Finally, in Chapter 6, we conclude this thesis, and discuss the limitations of the collaborative coding schemes. Also, we propose future research directions. At the end of this thesis, there are Appendices that cover various physical channel coding schemes that were explored throughout the development of this research, as well as a visual tour of the Mesh Network Simulator.

## Chapter 2: Background and Literature Review

### 2.1. Wireless Mesh Networks

Mesh networks are a sub-class of wireless multi-hop networks that doesn't have a recognizable topology, and rely on a decentralized approach for management. [7] Unlike traditional wireless networks, which have a base station that virtually handles all the administrative tasks, wireless mesh networks rely on distributed measures that treat all connected wireless terminals equally. [8] Thus, each wireless node handles some control and routing responsibility. This not only makes the network expandable and scalable, but also offers redundancy and reliability since the network doesn't depend on a central unit for its survival. However, this also means that each node is required to be more complex to handle administrative tasks. Moreover, distributed routing and multiple access mechanisms are more complex than their centralized counterparts. The IEEE 802.16 standard [9] (or WiMax) defines the data link (MAC) protocol of the Wireless Metropolitan Area Network (WMAN). It provides a lower cost solution to connect home and business to the Internet wirelessly.

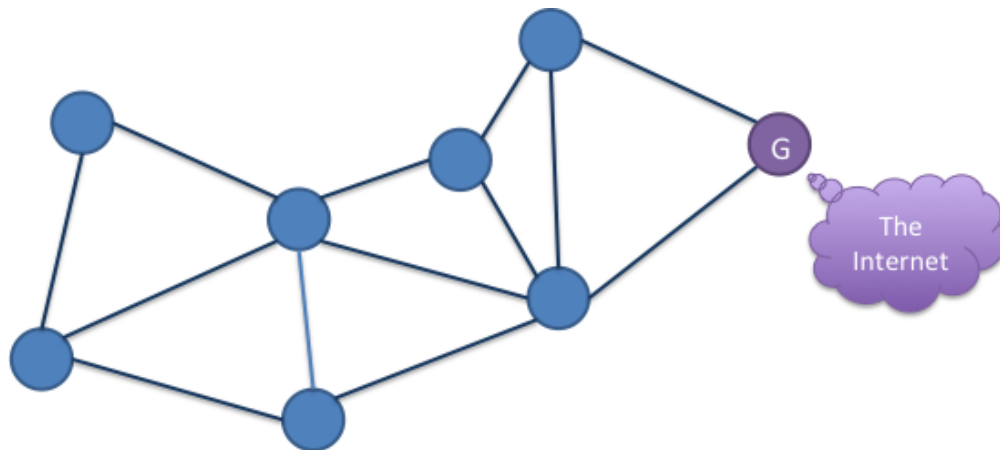


Figure 4: Mesh Network with a gateway



### 2.1.1. Properties of Wireless Mesh Networks

The following section provides some of the properties of wireless mesh networks that will be exploited further on in this thesis.

**The broadcasting nature of the wireless medium:** This means that nearby nodes can receive a packet intended for a particular neighbor receiver. While this can be regarded as wireless interference, it can be exploited, as we will see later on in this thesis, to enhance network performance. On the other hand, there is more than one path from source to destination over multiple hops, and this can be used in different ways; such as providing redundancy to the network, or dynamically shifting traffic along different paths to deal with congestion. Moreover, in contrast to wired mesh networks, there are no pre-defined paths between connected nodes. Each node is virtually connected to all the nodes within its radio range.

**Redundancy and reliability:** Decentralized mesh networks don't depend on a central unit for management. Thus, the network adapts dynamically when a node is lost, and thereby is more immune to vast network outages. I.e. if one or more nodes go offline, the network remains operational.

**Scalability and expandability:** In traditional wireless networks, a node should be within the range of a base station to connect to the network. In contrast, a wireless node simply needs to be within range of any connected wireless node.

**Inexpensive and practical in some scenarios:** Mesh networks are practical in deploying wireless networks on a large scale for VOIP and data services. The fact that a node is required to be near an access point or a base station means that simpler antennas with less power can be used to provide connectivity. The complexity of the nodes pays for the reduction in costs in static relay equipment, access points, and base stations.

The previous properties lead to some advantages as well as challenges when deploying wireless mesh networks. Although wireless mesh networks are scalable, dynamic, and reliable, there are

many challenges drawn from its distributed nature. Using the wireless medium efficiently while routing the packets and managing the network in a distributed fashion proved to be extremely challenging in practice. That is why wireless mesh networks haven't been widely used. However, many practical systems have recently evolved to address these issues. This thesis discusses those attempts and comes up with some enhancements and tweaks of its own.

### **2.1.2. Applications of Wireless Mesh Networks**

Wireless mesh networks can be used as a practical and inexpensive solution for VOIP applications due to their expandability and the fact that it is not necessary for a wireless node to be near a base station to be connected to the network. Thus, the network should easily support unicast as well as multicast transmission to allow both private calls and conferencing.

In addition, they have received much attention in recent years to provide community broadband Internet services. Such an application requires the wireless mesh network to allow and maintain several point-to-point connections simultaneously and in an efficient manner.

### **2.1.3. Opportunistic Routing in Wireless Mesh Networks**

Opportunistic routing is a routing scheme that tries to reduce the number of transmissions inside a mesh network by trying to forward the packet to the neighbor that is closest to the final destination. This requires the nodes to keep track of their neighbors at all times. [10] In contrast, other routing methods in mesh networks are similar to those of traditional wireless networks. They rely on either pre-defined static routing tables or dynamic tables that get updated regularly according to some tracked routing metric.

## 2.2. Relaying & Network Coding

Relaying is used to increase the throughput or extend the coverage of wireless networks. An intermediary node is used to forward the message received from one node to another. Network Coding, in the other hand, is the process of intelligently mixing packets at a relay before forwarding to the next hop. It is used to reduce the number of transmissions in the packet, and therefore, allows for more efficient networks. In the following sub-sections, we explore relaying, and discuss network coding in its digital and analog flavors.

### 2.2.1. Two-Way-Relay communication

When two wireless nodes are too far to communicate directly with each other, a relay can be used to facilitate the communication. This scenario is referred to in literature as the Two-Way-Relay problem (Figure 5). In this thesis, the following assumptions are made for this scenario:

- *Node 1* wants to send *Packet A* to *Node 2*, and *Node 2* wants to send *Packet B* to *Node 1*.
- *Packet A* and *Packet B* have the same length of  $N$  bits.
- For the sake of simplicity, we assume that communication between the sending nodes is only possible through the use of the relay. I.e. none of the sending nodes can overhear or receive any partial side-information when others transmit to the relay.
- Transmission in the channel is half-duplex. Transmitting terminals use the whole bandwidth available in the channel when communicating with the relay.
- The received power at the relay from each user after path loss is equal. In other words, all users transmit at the same power, and the distance between each user and the relay is equal.



Figure 5: Two-Way-Relay problem

Traditionally, communication in a Two-Way-Relay scenario goes as follows: Node 1 sends packet A to the relay in the first time slot. Then, the relay forwards packet A to Node 2 in the second time slot. The same sequence is performed on the opposite direction to transfer packet B from Node 2 to Node 1 through the relay. [11] The transmission flow for this scenario is shown in Figure 6. It is important to note here that each transmission has to happen in its own timeslot to avoid collisions. The MAC (data link) layer takes care of coordinating the transmissions in such fashion. In total, four time slots are needed to send bi-directional data through one shared wireless channel and a relay. [12]

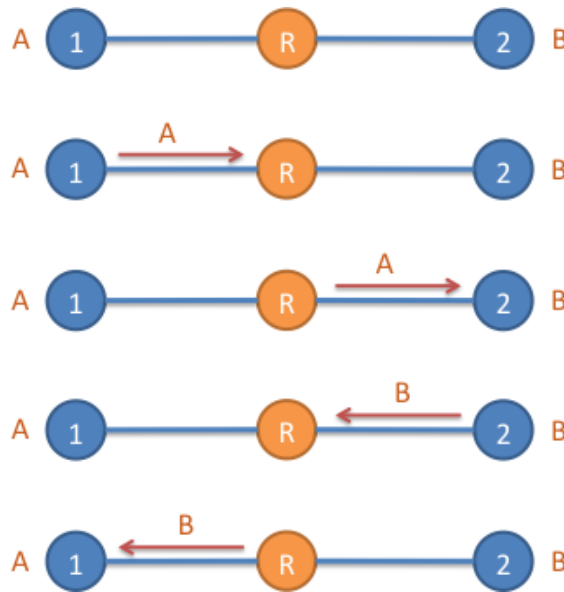


Figure 6: Two-way communication with a relay in the middle

When the relay receives a packet, it can either amplify or decode the signal before forwarding it. The following two sub-sections describe those two processes in more detail.

### 2.2.1.1. *Amplify & Forward approach*

In the Amplify and Forward approach, [13]the relay receives the signal, amplifies it and forwards it. In other words, the relay acts as an analog repeater, which means that when the signal is amplified, the noise is amplified as well.

### ***2.2.1.2. Decode & Forward approach***

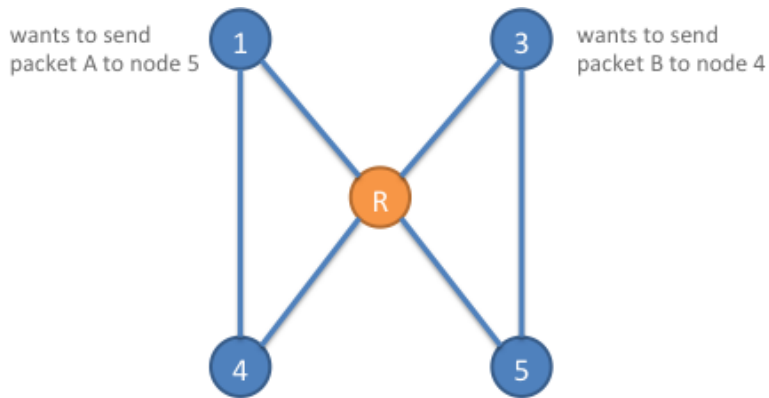
In the Decode and Forward approach, [14]the relay decodes the received digital symbols based on used MCS (Modulation & Coding Scheme). After that, it re-modulates and re-encodes the message before forwarding it through the wireless channel. The benefit of this approach is that decoding/demodulating the received signal accounts for the noise in the channel, and in most cases, produces better performance than the Amplify and Forward approach. [4] This approach is generally used in digital telecommunication networks.

Even if relay communication with analog network coding is used, the received signal at the relay can be decoded using known decision intervals before forwarding the signal to receivers.

Throughout this thesis, the Decode and Forward approach is used in all the described solutions & simulation results.

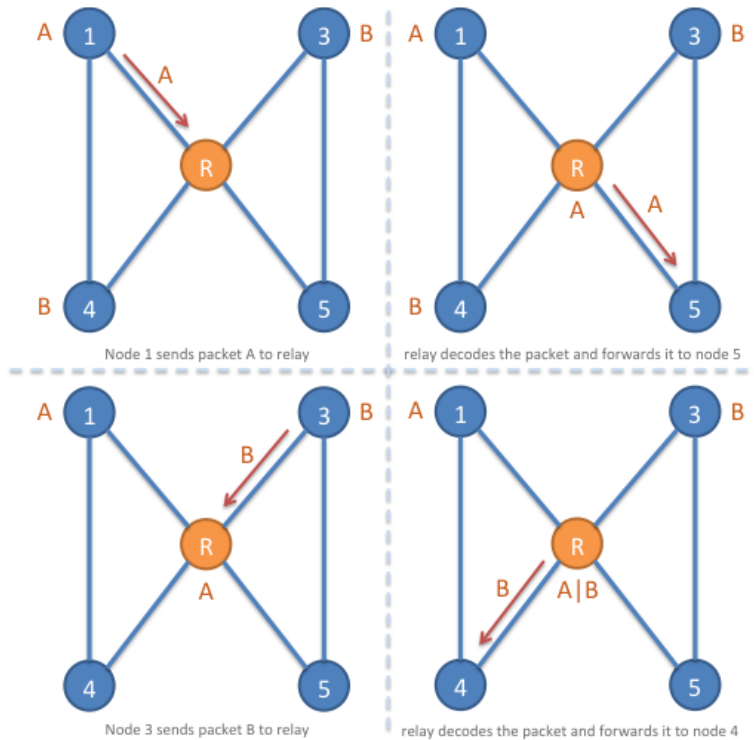
### **2.2.2. Digital Network Coding**

Network coding is the process of mixing packets in a telecom network through add-modulo operations, and hence, decreasing the number of transferred packets to achieve higher network throughput. The simplest form of network coding is explored via the butterfly topology shown in Figure 7. [15] In this scenario, four terminals communicate via a relay, with the following assumptions in mind: Node 1 wants to send packet A to Node 5, and Node 3 wants to send packet B to Node 4. Also, Node 1 and Node 3 cannot communicate directly with Node 5 and Node 4 respectively. Hence, both senders are forced to transmit through the relay R.



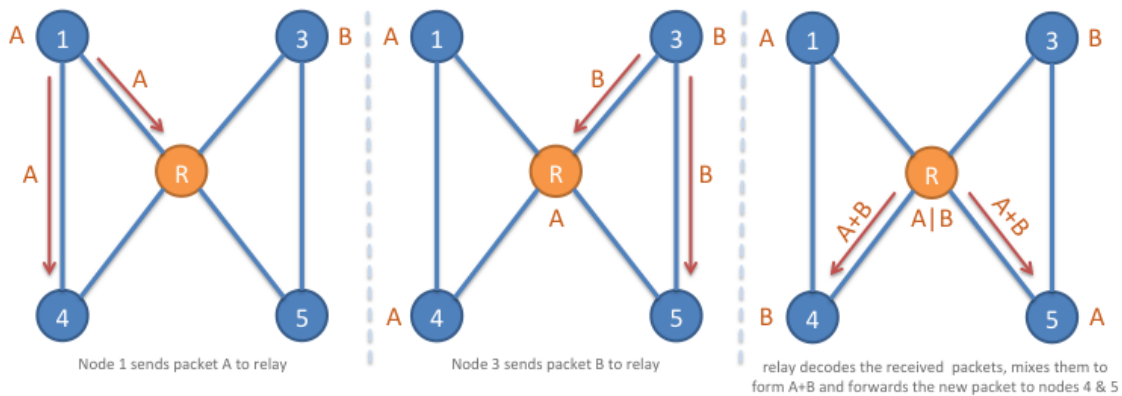
**Figure 7: Butterfly Network Topology**

If traditional routing is used and network coding is not considered, Node 1 sends packet A to the relay, then the relay decodes the received message and forwards it to Node 5. After that, Node 3 sends packet B to the relay, and the relay forwards it again to Node 4. Thus, four transmissions in total are needed. This is shown in the following figure.



**Figure 8: Transmission flow diagram in a butterfly network without network coding**

Now let's assume that network coding was used in conjunction with basic routing. Since there is a link between Node 1 and Node 4, Node 1 broadcasts packet A to Node 4 and the relay simultaneously. Similarly, Node 3 broadcasts packet B to Node 5 and the relay. The relay uses network coding to mix packets A and B together by performing an XOR operation, and transmit the new packet (A+B) to Nodes 4 and 5 concurrently. Since Node 4 already has packet B, it can extract packet A easily by using basic algebra; XORing A with A+B will give back packet B. Similarly at Node 5, packet B is extracted from packets A and A+B. The transmission flow is shown in Figure 9.



**Figure 9: Transmission flow diagram in a butterfly network with network coding**

With network coding, only three transmissions are required to accomplish the same task as traditional routing without network coding. That is a gain of  $4/3 = 1.33$ .

It can be seen from the previous example that network coding can be an extremely viable and practical solution for multi-cast applications, but it has captured some attention in the past few years as a novel solution to enhance throughput for unicast transmissions in wireless mesh networks. [16] Giving the broadcasting nature of the wireless medium, a sent packet in wireless mesh topology is often received by more than one terminal, making network coding a natural progression for enhancing wireless networks. Traditionally, if the packet wasn't received by the intended next hop in a wireless network, it is ignored and dropped from the recipient's queue. However, with network coding, a received packet is kept for a short period of time regardless of whether the receiver is the next hop or not. Based on knowledge (or prediction) of what the contents of neighbors' queues are, the stored packets can be mixed together in smart ways to

reduce the number of overall transmissions as much as possible. In other words, we can look at network coding as a form of *compression* that enhances the overall throughput and efficiency of the network. We will go through a few practical systems that take advantage of network coding to enhance network performance in the next few sections.

#### ***2.2.2.1. Practical Use of Network Coding in Modern Networks***

Network coding has been used conventionally in multi-cast, VOIP, and content distribution network applications. Microsoft Research has proposed a content distribution system that relies on network coding for delivery. Their solution transforms desktop clients/users into partial contributors to the delivery mechanism. Let's assume that an OS Software Developer wants to deliver a software update to all of its connected users. Normally, all users try to access the server and obtain the update at the same time. This causes wasted bandwidth because the same information is sent multiple times to multiple users. In addition, the servers may not handle the excessive simultaneous requests to download the update. With network coding in mind, the servers can send to a subset of users, who in turn can mix those packets in smart ways before forwarding to other users who will use the mixed packets to extract new packets, and choose a different combination of packets to mix before transmitting again. Through redundancy, and the effect of network coding, all users will end up with software update efficiently and without disrupting the download servers. [17].



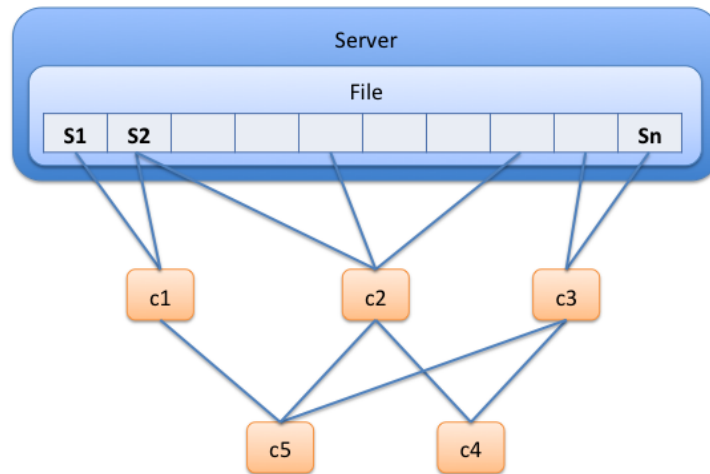


Figure 10: Using network coding for large scale content distribution

In recent years, a novel use of network coding emerged in wireless mesh networks. Much research has gone to come up with algorithms and protocols to utilize opportunistic network coding with opportunistic routing for unicast along with multicast transmission to achieve significant network throughput gains. Those proposed systems include BEND [18], COPE [19], and MIXIT [2].

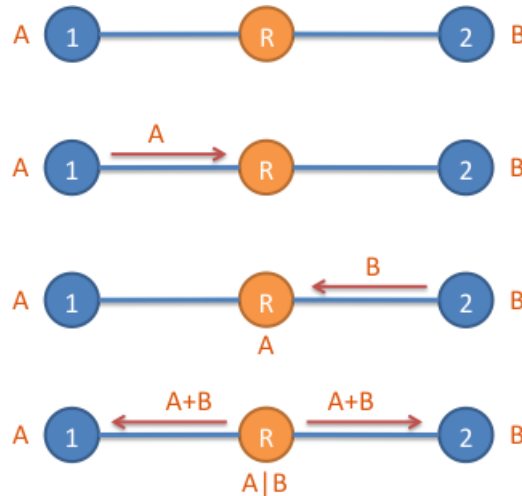
#### 2.2.2.2. Two-Way-Relay Communication with Digital Network Coding

Digital network coding can be used in the two-way communication with relay scenario. Let's assume we have a Two-Way-Relay scenario, where Nodes 1 and 2 want to transmit messages A and B respectively to each other (Figure 11). If no network coding is used, four messages need to be transmitted (Node one to relay, relay to Node two, Node two to relay, and relay to Node one). However, with digital network coding, Node 1 sends message A to relay, and Node 2 sends message B to relay. After that, the relay mixes the two messages digitally using an XOR operation and transmits  $A+B$  to both terminals, thereby exploiting the broadcasting nature of the wireless medium. Upon reception, Node 1 extracts message B from the received message by remixing the received signal with its own message using an XOR operation. The same operation is conducted at Node 2 to extract message A. The following table gives all the possibilities for A and B given that they are binary bits, along with the outcome of the relay, and the final extracted messages at each node.

**Table 1: Possible binary outcomes of the two-way communication with relay and digital network coding**

Bit @ Node 1	Bit @ Node 2	Bit transmitted by relay	Bit extracted at Node 1	Bit extracted at Node 2
A	B	$R = A+B$	$B' = R+A$	$A' = R+B$
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

The following figure shows the transmission flow when digital network coding is used in a Two-Way-Relay scenario.



**Figure 11: Two-way communication with relay and digital network coding**

With this approach, the total number of required transmissions is reduced from four to three. Many network routing schemes for wireless mesh networks emerged to exploit digital network coding and reduce the total number of transmissions in a network. The most notable is COPE, which was introduced by an MIT team in 2008. **Invalid source specified.**

### 2.2.3. Analog (Physical) Network Coding

In a multi-user multi-node wireless environment, a transmission from one node is considered unwanted noise for other nodes/users who transmit on the same channel. The MAC layer in the

OSI model is responsible of managing multiple nodes in a network, and preventing multiple users of transmitting at the same time to avoid collisions in the channel. However, noise can be exploited through collaborative concurrent transmission between multiple nodes to minimize the number of transmissions, and that is what Analog Network Coding (ANC) is about. [5]

Let's consider an example to explain the idea of ANC. Assume we have a Two-Way-Relay scenario, and that Node 1 and Node 2 are synchronized in some way to send their packets concurrently so that the  $i^{\text{th}}$  bit of Node 1 and the  $i^{\text{th}}$  bit of Node 2 are received at the relay at exactly the same time, and the received power from each user at the relay is relatively equal. Obviously, their packets will collide in air, and the relay antennas would receive an analog signal that is the result of adding both signals. [5] [20] Traditionally, this scenario would be considered non-ideal as the interfering node will induce low SINR (Signal to Interference & Noise Ratio) and reduce the receiver's fortune of decoding the packets successfully. However, with analog network coding in mind, the relay simply amplifies the signal as it is (or decodes using known decoding decision regions), and forwards it to both ends. From the mixed signal, and the knowledge of its own packet, each node can extract the other node's packet. This theoretically means that two-way communication through a relay can be achieved with only two transmissions. In other words, using ANC in this case is equivalent of putting one of the nodes in the location of the relay. [21]

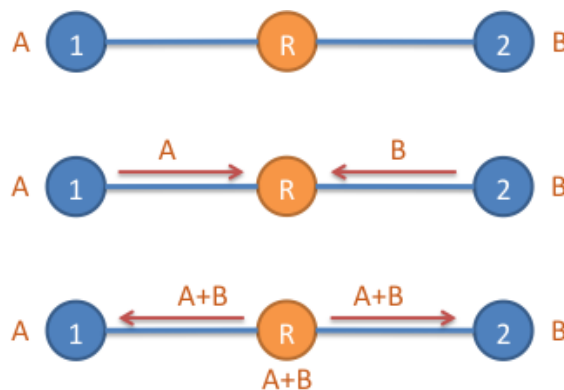


Figure 12: Two-way communication with relay and analog network coding

The problem with this approach is that if the relay doesn't use the decode-and-forward approach, and instead it amplifies then forwards the received signal, noise will also be amplified with the

signal before the second slot transmission, which will also add more noise to the final received signal. Thus, we expect the BER will be relatively larger to using Digital Network Coding. In addition, perfect synchronization between the collided signals at the relay is extremely difficult, especially when one takes into account that the nodes might be moving, and the relay might be at proximity of one of the nodes. Moreover, the received power from each terminal at the receiver should be roughly equal, in order for the decoding decision regions to be accurate. This is challenging when the users are moving. Last but not least, the length of the packets might be different, which adds to the complexity of the signal extraction algorithms at the receiving ends.

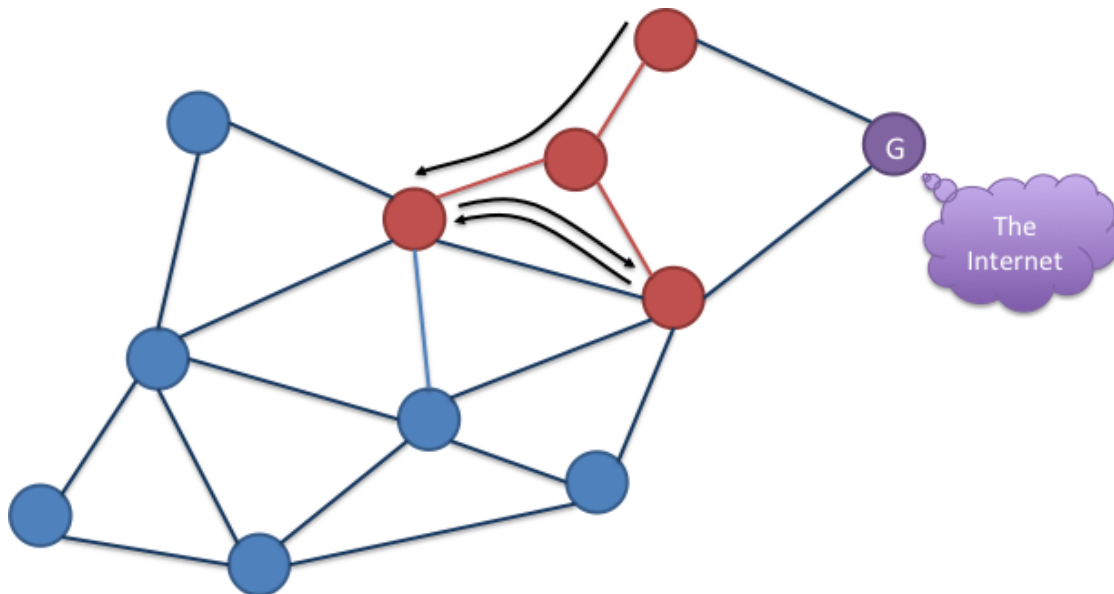
What is great about analog network coding is that it blurs the boundaries between the layers of the OSI model, by elevating the physical layer to perform activities that were once only reserved for higher layers. ANC is essentially network coding, done on the physical layer. In addition, the fact that ANC uses pure noisy signals to reduce transmission time makes this technique unique and novel.

Giving exact bounds on coding gains when using ANC in a wireless mesh network is extremely difficult, especially when you have multiple sources. That is due to the fact that the topology of wireless mesh networks is not known and constantly changing (if the wireless nodes are moving). [16]

### **2.3. Y-Channel-Relay Communication**

The model for multiuser communication channel, in which multiple users exchange information with the help of a relay terminal was first introduced by D. Gunduz in [22]. It was called the *Multiway Relay Channel*. In this model, an interfering group of users wants to communicate among themselves by multicasting messages among themselves via a relay in the middle. It is assumed that none of the users can receive messages, even partially, from other users directly.

In this section, we extend the discussion on relaying to the Y-Channel-Relay problem – where three terminals communicate via a common relay in a wireless medium. [23] This scenario is practically realized in multi-cast applications like LTE eMBMS, VoIP and tele-conferencing applications when three wireless users communicate over a relay, or in Wireless Mesh Networks with Opportunistic Routing. The following diagram shows an example Y-Channel-Relay problem in a such a network.



**Figure 13: An example of a Three-Node communication through a relay in a Wireless Mesh Network**

Figure 10 shows such a network – three terminals communicating via a common relay. Throughout this thesis, we shall refer to this topology as the *Y-Channel-Relay problem*. We assume the following:

- *Packet A, Packet B, and Packet C are always associated with Node 1, Node 2, and Node 3 respectively.*
- *Packet A, Packet B and Packet C have the same length of  $N$  bits.*

- For the sake of simplicity, we assume that communication between the sending nodes is only possible through the relay. I.e. none of the sending nodes can overhear or receive any partial side-information from the other nodes directly.
- Transmission is half-duplex. Also, a terminal uses the whole bandwidth available in the channel when transmitting a packet. FDMA (Frequency Division Multiple Access) is not an option.

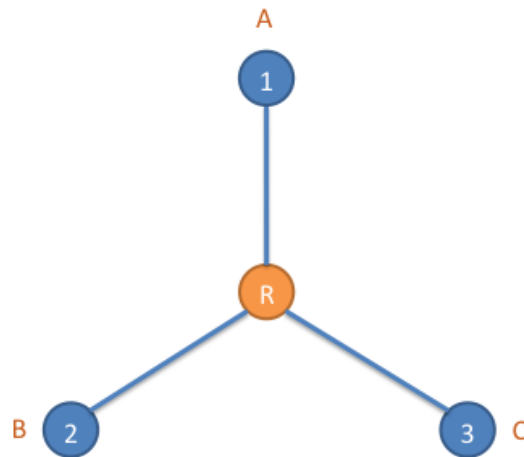
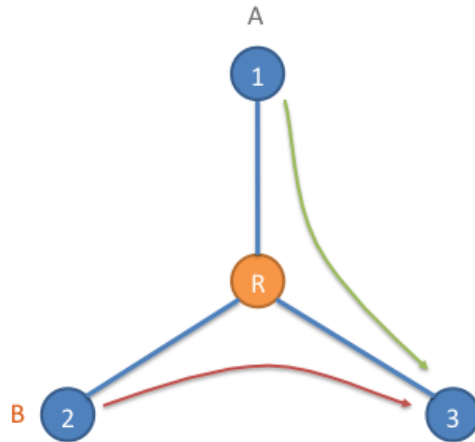


Figure 14: The Y-Channel-Relay problem

There are many transmission scenarios that arise when having three nodes communicating through a relay in a wireless medium. Table 7 lists all the possible scenarios. The ones involving three nodes can be categorized into two classes: Two-Sender, and Three-Sender topologies.

### 2.3.1. Y-Channel Two-Sender Topology

In Two-Sender topologies, two nodes have packets to be sent to a subset of the three users in the network. [4]Let's take Figure 15 as an example. In this scenario, Node 1 wants to send Packet A to Node 3, and Node 2 wants to send Packet B to Node 3. In total, we have two senders, and one receiver.



**Figure 15: A Y-Channel-Relay problem in a Two-Sender topology. Example 1.**

Without the use of network coding, Node 1 has to send packet A to the relay in the first time slot. Then, the relay forwards the packet to Node 2 in the second time slot. After that, Node 2 sends Packet B to the relay in the third time slot. Finally, the relay forwards Packet B to Node 3. In total, 4 transmissions are required to conclude the communication. Figure 16 shows the transmission flow for this scenario.

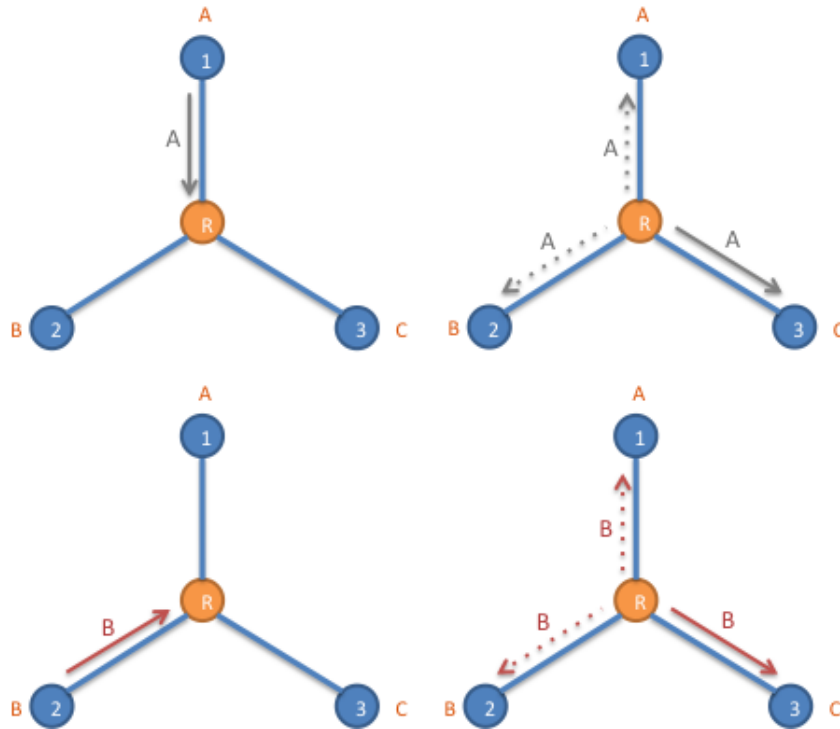


Figure 16: Transmission in Two-Sender network (example 1) without the use of network coding

Now let's assume that digital network coding is considered and the following assumptions are made:

- Node 1 and Node 2 are equi-probable binary sources over the domain  $\{0, 1\}$ .
- The modulation scheme used @ Node 1 and Node 2 is BPSK:
  - For Bit 0, the transmitted signal  $\rightarrow -\sqrt{E_b}\phi(t)$
  - For Bit 1, the transmitted signal  $\rightarrow +\sqrt{E_b}\phi(t)$
  - $\phi(t) = \sqrt{\frac{2}{T_b}} \cos(2\pi f_c t)$ , where  $f_c$  is the carrier frequency, and  $T_b$  is the period of transmitting one bit. [24]

Node 1 sends Packet A to the relay. Then, Node 2 sends Packet B to the relay. After that, the relay decodes and mixes the two packets through an XOR operation and broadcasts the mixed signal ( $A \oplus B$ ). Given the assumptions above, the broadcasted bit can be one of three possibilities detailed in Table 2.



**Table 2: Received signal at Node 3 when using digital network coding in a Two-Sender Y-Channel-Relay scenario**

Bit @ Node 1	Bit @ Node 2	Signal transmitted by relay	Outcome upon reception @ receiving user (Node 3)
A	B	$R = A \oplus B$	
0	0	$-2\sqrt{E_b}\phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 0.
0	1	0	→ Erasure
1	0	0	→ Erasure
1	1	$+2\sqrt{E_b}\phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 1.

It is interesting to note that the transmission power of the relay is equal to the sum of the transmission power of both transmitting nodes. Since Node 1 and Node are equi-probably binary sources, the bit energy and transmission power are as follows:

- $E_b$  (at Node 1) =  $0.5 E_{b \rightarrow 0} + 0.5 E_{b \rightarrow 1} = E_b$
- $E_b$  (at Node 2) =  $0.5 E_{b \rightarrow 0} + 0.5 E_{b \rightarrow 1} = E_b$
- $E_b$  (at Relay) =  $0.25 E_{b \rightarrow 0,0} + 0.25 E_{b \rightarrow 0,1} + 0.25 E_{b \rightarrow 1,0} + 0.25 E_{b \rightarrow 1,1} = E_b$
- $Transmission\ Power_{at\ Node\ 1} = \frac{1}{2} \times \frac{1}{2} E_b + \frac{1}{2} \times \frac{1}{2} E_b = \frac{1}{2} E_b$
- $Transmission\ Power_{at\ Node\ 2} = \frac{1}{2} \times \frac{1}{2} E_b + \frac{1}{2} \times \frac{1}{2} E_b = \frac{1}{2} E_b$
- $Transmission\ Power_{at\ relay} = \frac{2}{4} \times \frac{4}{2} E_b = E_b$

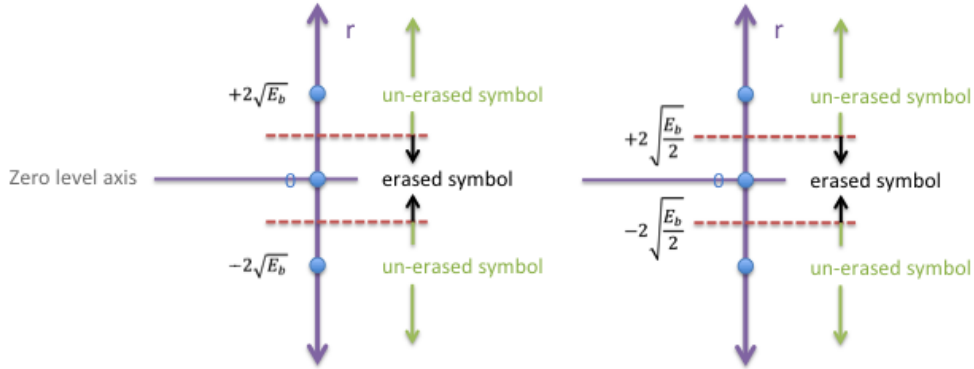
Where  $E_{b \rightarrow 0}$  is the bit energy of zero,  $E_{b \rightarrow 1}$  is the bit energy of one.

We can see that the transmission power at the relay is twice that of a sender. That being said, we can adapt the transmission power of the relay so that it is equal to that of a sender. We shall call this mode of transmission the *Adapted Transmission* mode. Table 3 details the levels that the relay should transmit at in order to have the same transmission power as any of the senders. One must bear in mind that the system will suffer a  $\sim 3$  dB loss in physical layer performance when using this mode.

**Table 3: Forwarded signal from the relay when using network coding in a Two-Sender Y-Channel-Relay scenario**

Bit @ Node 1	Bit @ Node 2	Signal transmitted by relay
A	B	$R = A \oplus B$
0	0	$-2 \sqrt{\frac{E_b}{2}} \phi(t)$
0	1	0
1	0	0
1	1	$+2 \sqrt{\frac{E_b}{2}} \phi(t)$

Upon the reception of the signal over an AWGN channel ( $r = R + n$ ), the receiver at Node 3 demodulates based on the decision regions shown in Figure 17, then forwards the mixed packet to the receivers.



**Figure 17: Decision regions at receiver in a Two-Sender Y-Channel-Relay scenario (non adaptive (left) vs. adaptive (right) power transmissions)**

When the received signal is in the decision region  $-2\sqrt{E_b} \leq r < +2\sqrt{E_b}$ , the receiver won't be able to give a decision based on modulation alone, and the bit is considered erased. This is equivalent to a Hybrid Binary Erasure Channel (See section 2.4 for more details). [25] Recent research efforts [26] [27] [28] proposed multiple collaborative coding schemes to solve this problem and provide reliable communication using network coding in a Y-Channel-Relay topology. It is important to note that these schemes/solutions add redundancy and parity, and reduce the rate of sent information packets. However, the gain introduced by reducing the number of transmissions outweighs the decrease in the rate of information in a sent codeword. Those schemes proved that using network coding in a Y-Channel-Relay may boost overall

throughput of the network. The following diagram shows the transmission flow when digital network coding is used; three transmission time slots are needed.

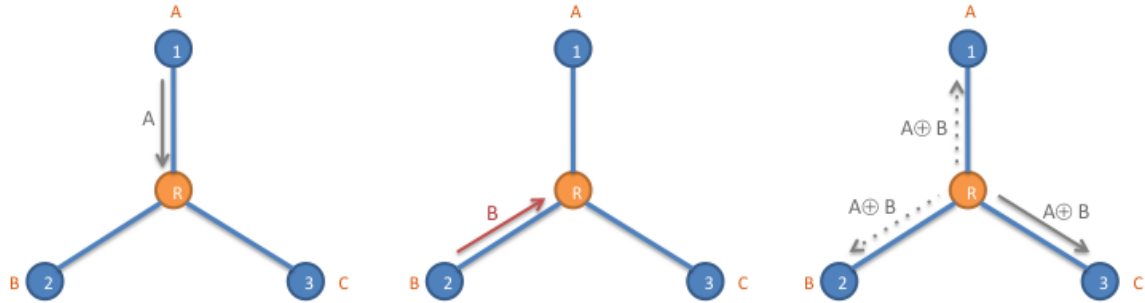


Figure 18: Transmission flow in a Two-Sender Y-Channel-Relay problem (example 1). Three transmissions are required.

If we take the previous problem, and consider the use of Analog (Physical) network coding, Node 1 and Node 2 can send Packet A and Packet B simultaneously. If we assumed that the relay receives the  $i^{\text{th}}$  bit of each at exactly the same time, then the signals will be mixed over the air as per Table 2. In this case, the number of transmissions is reduced from three to only two (Figure 19).

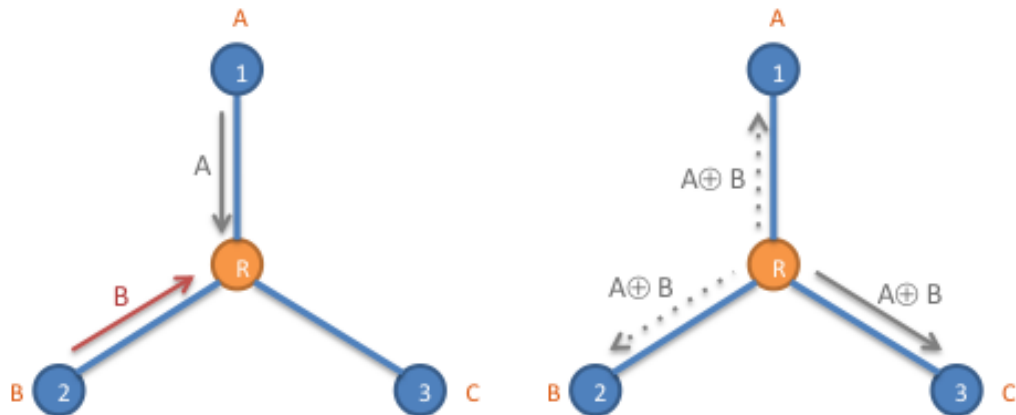


Figure 19: Transmission flow in a Two-Sender Y-Channel-Relay problem with physical network coding. Only two transmissions are required.

Let's look at another example of a Two-Sender Y-Channel-Relay topology. In this scenario, we have two senders, but more than one receiver. Node 1 wants to send Packet A to Node 3, and Node 3 wants to send Packet C to Node 2 and Node 1.

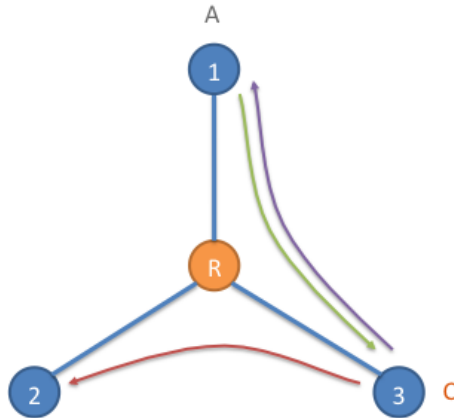


Figure 20: Two-Sender Y-Channel-Relay problem (example 2).

If we don't consider network coding and we rely on opportunistic routing and medium access control in the wireless medium, four transmissions are required to complete the communication; Node 1 will send Packet A to the relay in the first transmission instance. Then, the relay will broadcast Packet A. After that, Node 3 will send Packet C to the relay in the third transmission slot after which the relay broadcasts it.

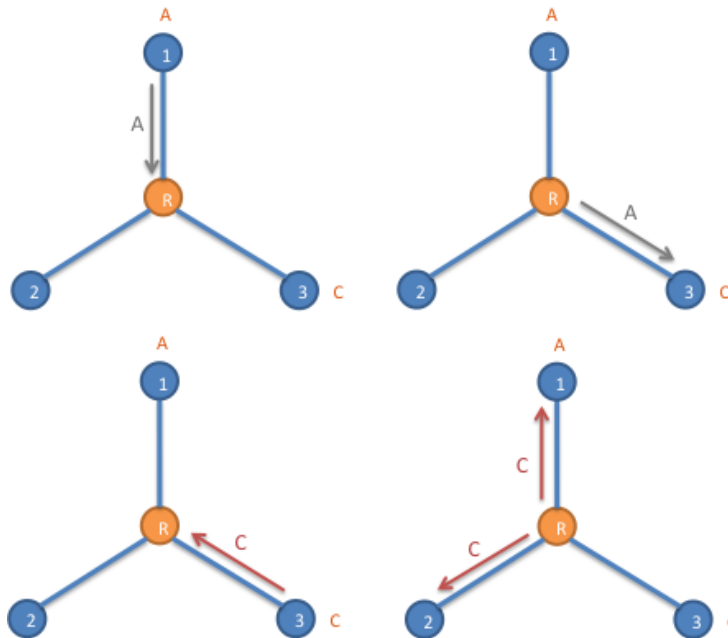
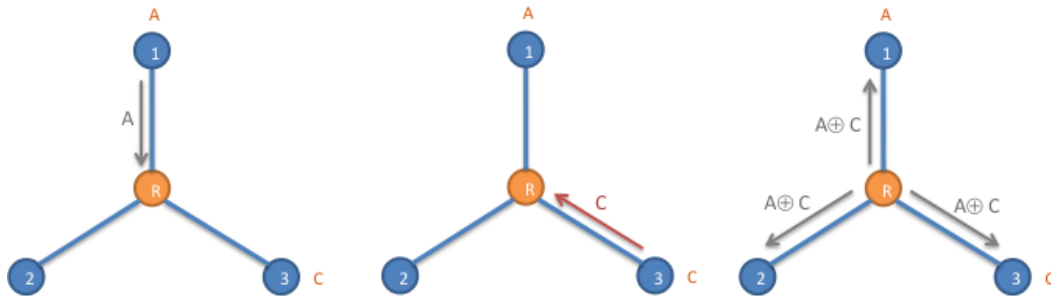


Figure 21: Transmission flow in Two-Sender Y-Channel-Relay problem (example two). Four transmission slots are required.

If we consider digital network coding in this example, Node 1 sends Packet A to the relay in the first time slot. Then, Node 3 sends Packet C to the relay in the second time slot. After that, the relay mixes the two packets and broadcasts  $(A \oplus C)$ . Upon reception:

- At Node 1: Since Node 1 has knowledge of its own information packet (Packet A), Node 1 can extract Packet C by conducting the following XOR operation:
  - *Decoded message at Node 1* =  $A \oplus (\text{Received}) = A \oplus (A \oplus C) = C$
- At Node 2: Node 2 will have to separate the two signals with the use of pre-coded collaborative coding schemes and retrieve Packet C.
- At Node 3: This node will extract its information packet (Packet C) from the received signal:
  - *Decoded message at Node 3* =  $C \oplus (\text{Received}) = C \oplus (A \oplus C) = A$

Figure 22 shows the transmission flow for this example.



**Figure 22: Transmission flow in Two-Sender Y-Channel-Relay topology with Digital Network Coding (example 2). Three transmissions are required.**

As we saw, using digital network coding reduced the number of transmissions from four to three. Now, if we consider analog (physical) network coding, we can reduce the number of transmissions to two by allowing Node 1 and Node 2 to send Packets A and C concurrently. Figure 23 shows the transmission flow when using analog network coding.

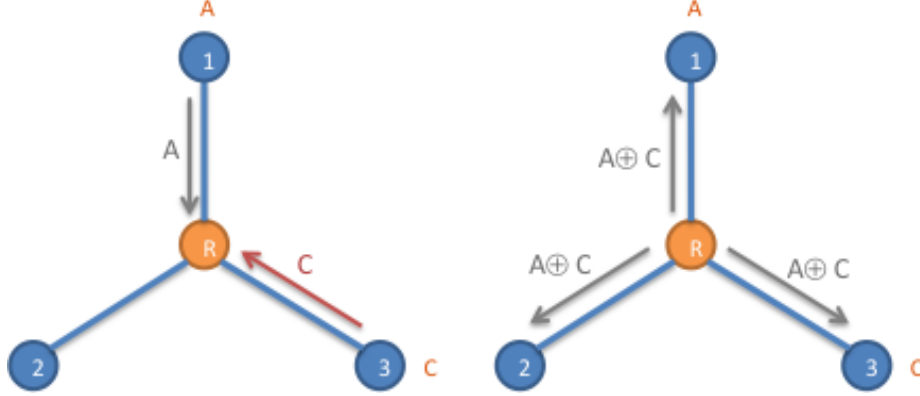


Figure 23: Transmission flow in Two-Sender Y-Channel-Relay topology with Analog (Physical Network Coding) (example 2). Only two transmissions are required.

In the next sub-section, we will discuss scenarios where all terminals in a Y-Channel-Relay topology are senders.

### 2.3.2. Y-Channel Three-Sender Topology

Three-Sender topology is a class of Y-Channel-Relay scenarios where all three users are sending to a combination of the three available nodes. The following assumptions are taken into consideration throughout this discussion:

- Node 1, Node 2 and Node 3 are equi-probable binary sources over the domain  $\{0, 1\}$ .
- The modulation scheme used @ Node 1, Node 2 and Node 3 is BPSK:
  - For Bit 0, the transmitted signal  $\rightarrow -\sqrt{E_b}\phi(t) \rightarrow$  Transmission power  $= \frac{E_b}{2}$
  - For Bit 1, the transmitted signal  $\rightarrow +\sqrt{E_b}\phi(t) \rightarrow$  Transmission power  $= \frac{E_b}{2}$
  - $\phi(t) = \sqrt{\frac{2}{T_b}} \cos(2\pi f_c t)$ , where  $f_c$  is the carrier frequency, and  $T_b$  is the period of transmitting one bit. [24]

Figure 24 shows an example of a Three-Sender Y-Channel-Relay network; Node 1 wants to send Packet A to Node 2, Node 2 wants to send Packet B to Node 3, and Node 3 wants to send packet C to Node 1. Without network coding, each node waits for its turn to send its packet to the relay,

which broadcasts the packet to the final destination. Thus, six transmissions are needed (Figure 25).

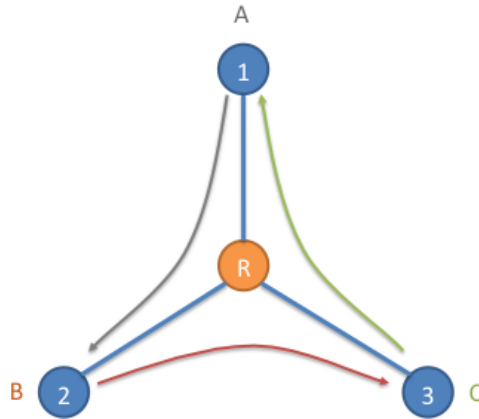


Figure 24: Three-Sender Y-Channel-Relay scenario (example 1).

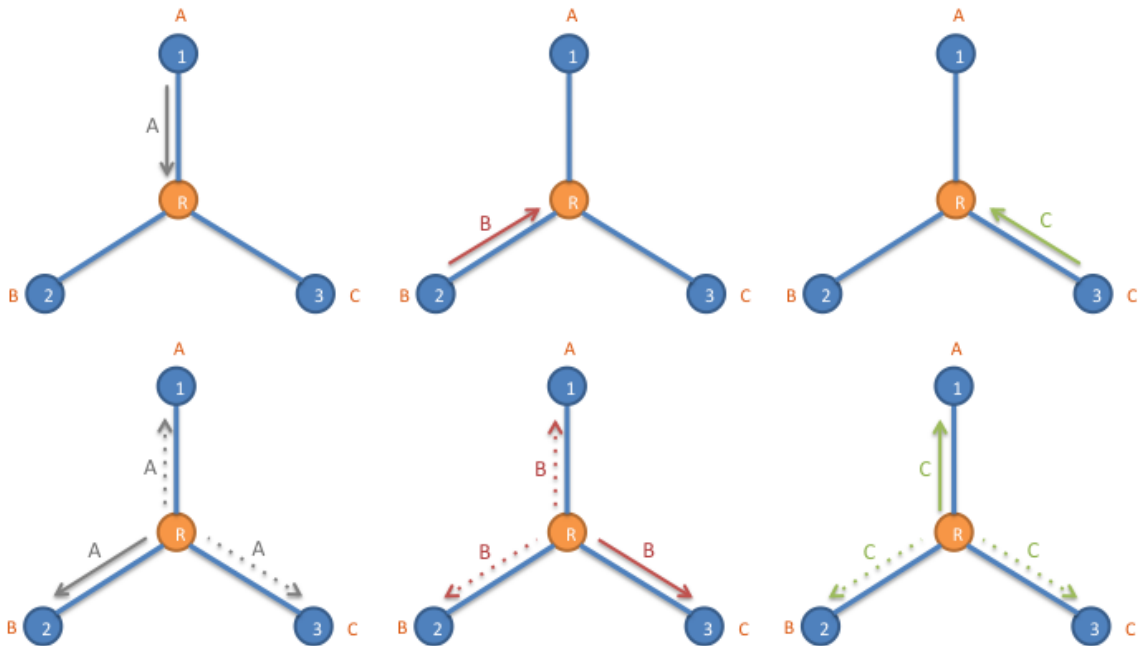


Figure 25: Transmission flow diagram for Three-Sender Y-Channel-Relay scenario (example 1).

If digital network coding is used, each node sends its packet to the relay, and the relay can either use a *Decode & Forward* or *Amplify & Forward* approach to broadcast the mixed signal ( $A \oplus B \oplus C$ ). The number of required transmissions drops to 4 transmissions as can be seen in Figure 26. Table 4 shows the possible signals that the relay broadcasts.

**Table 4: Transmitted signal of relay when using network coding in a Three-Sender Y-Channel-Relay scenario**

Bit @ Node 1	Bit @ Node 2	Bit @ Node 3	Signal transmitted by relay	Outcome upon reception @ receiving user (Node 3)
A	B	C	$R = A \oplus B \oplus C$	
0	0	0	$-3\sqrt{E_b}\phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 0.
0	0	1	$-\sqrt{E_b}\phi(t)$	→ Erasure
0	1	0	$-\sqrt{E_b}\phi(t)$	→ Erasure
0	1	1	$+\sqrt{E_b}\phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 1.
1	0	0	$-\sqrt{E_b}\phi(t)$	
1	0	1	$+\sqrt{E_b}\phi(t)$	
1	1	0	$+\sqrt{E_b}\phi(t)$	
1	1	1	$+3\sqrt{E_b}\phi(t)$	

The average transmission power at each node is calculated as follows:

- $Transmission\ Power_{at\ Node\ 1} = \frac{1}{2} \times \frac{1}{2} E_b + \frac{1}{2} \times \frac{1}{2} E_b = \frac{1}{2} E_b$
- $Transmission\ Power_{at\ Node\ 2} = \frac{1}{2} \times \frac{1}{2} E_b + \frac{1}{2} \times \frac{1}{2} E_b = \frac{1}{2} E_b$
- $Transmission\ Power_{at\ Node\ 2} = \frac{1}{2} \times \frac{1}{2} E_b + \frac{1}{2} \times \frac{1}{2} E_b = \frac{1}{2} E_b$
- $Transmission\ Power_{at\ relay} = \frac{2}{8} \times \frac{9}{2} E_b + \frac{6}{8} \times \frac{1}{2} E_b = \frac{3}{2} E_b$

We can conclude that the transmission power at the relay is three times higher than the transmission power at each sender. This  $\sim 4.77$ dB increase in power requirement can be deemed insignificant in high SNR situations since the reduction in the total number of transmission (network performance) can be much more beneficial than physical layer performance. We conclude that the power of the relay increases linearly with the number of senders in a Y-Channel-Relay network [22]:

$$P_r = LP, \text{ where } P \text{ is the power at the relay,}$$

*L* is the number of transmitting users, and *P* is the user transmission power



We can adapt the relay in very high SNR conditions to send at the same power as any of the sender nodes (this can also be used if the relay is constraint to have the same transmission power as the users). [22] Table 5 details the signals transmitted at the relay when its transmission power is equal to any of the transmitting terminals:

**Table 5: Transmitted signal of relay when using network coding in a Three-Sender Y-Channel-Relay scenario and adapted power scheme to match the power of a transmitting node**

Bit @ Node 1	Bit @ Node 2	Bit @ Node 3	Signal transmitted by relay	Outcome upon reception @ receiving user (Node 3)
A	B	C	$R = A \oplus B \oplus C$	
0	0	0	$-3 \sqrt{\frac{E_b}{3}} \phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 0.
0	0	1	$-\sqrt{\frac{E_b}{3}} \phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 0.
0	1	0	$-\sqrt{\frac{E_b}{3}} \phi(t)$	→ Erasure
0	1	1	$+\sqrt{\frac{E_b}{3}} \phi(t)$	→ Erasure
1	0	0	$-\sqrt{\frac{E_b}{3}} \phi(t)$	→ Erasure
1	0	1	$+\sqrt{\frac{E_b}{3}} \phi(t)$	→ Erasure
1	1	0	$+\sqrt{\frac{E_b}{3}} \phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 1.
1	1	1	$+3 \sqrt{\frac{E_b}{3}} \phi(t)$	→ Receiving node can conclude that both Bit A and Bit B are 1.

$$Transmission\ Power_{at\ relay\ (adapted)} = \frac{2}{8} \times \frac{9}{2 \times 3} E_b + \frac{6}{8} \times \frac{1}{2 \times 3} E_b = \frac{1}{2} E_b$$

By using an adapted transmission power at the relay, we lose ~3dB physical layer performance. This might be acceptable in high SNR conditions given that the number of transmissions will be reduced from six without network coding to only 2 with network coding. However, we might need to transmit at higher power in low SNR conditions to reduce the number of re-transmissions induced by bit error in that region.

Upon the reception of the broadcasted signal from the relay, each receiving node subtracts its packet as follows:

- At Node 1:  $A \oplus (\text{Received}) = A \oplus (A \oplus B \oplus C) = B \oplus C$
- At Node 2:  $B \oplus (\text{Received}) = B \oplus (A \oplus B \oplus C) = A \oplus C$
- At Node 3:  $C \oplus (\text{Received}) = C \oplus (A \oplus B \oplus C) = A \oplus B$

The scenario, in this case, is reduced to an erasure problem similar to the one discussed in Figure 17. In the following chapter, we will discuss solutions introduced in recent research efforts.

Figure 25 shows the transmission flow when using Digital Network Coding in a Three-Sender Y-Channel-Relay scenario.

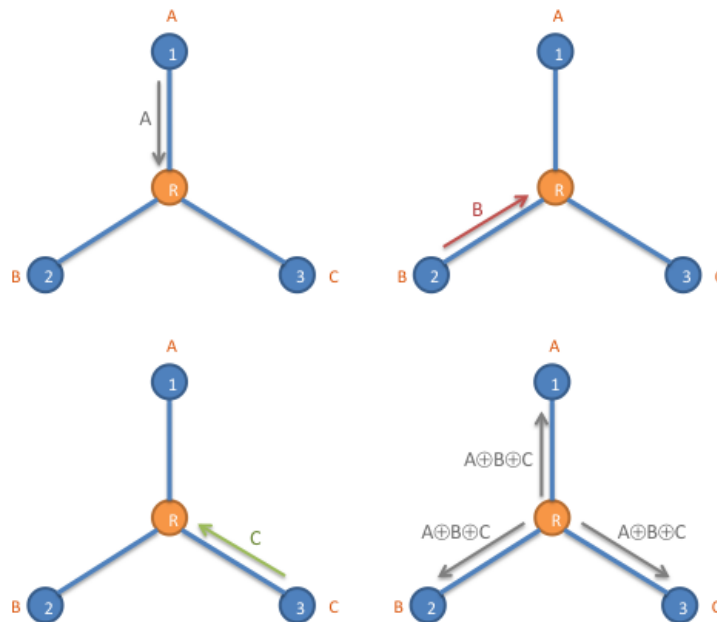


Figure 26: Transmission flow diagram Three-Sender Y-Channel-Relay Scenario with Digital Network Coding (example 1)

When Analog network coding is considered, the number of transmissions is reduced to two as all senders are allowed to send concurrently (Figure 27).

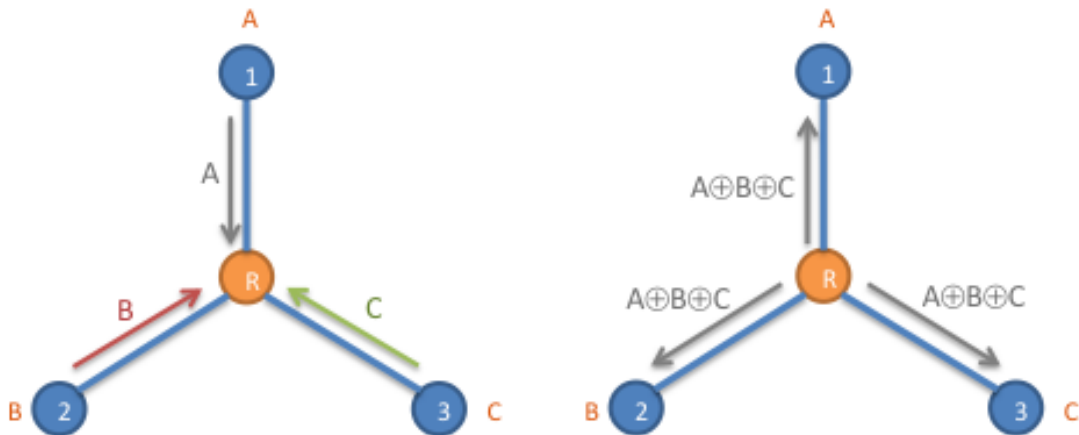


Figure 27: Transmission flow diagram for O=Shape Y-Channel-Relay Scenario with Analog (Physical) Network Coding (example 1)

Let's consider another example for Three-Sender Y-Channel-Relay networks. This time let's assume that we have the topology shown in Figure 28, and that Node 1 wants to send Packet A to Node 2 and Node 3, Node 2 wants to send packet B to Node 1 and Node 3, and finally, Node 3 wants to send packet C to Node 1 and Node 2. This scenario is practically viable if the three nodes want to do video conferencing with each other for example. This condition presents the maximum possible number of information packets exchanged in a Y-Channel-Relay problem. In total, six packets need to be sent (in contrast to only 3 in the example 1).

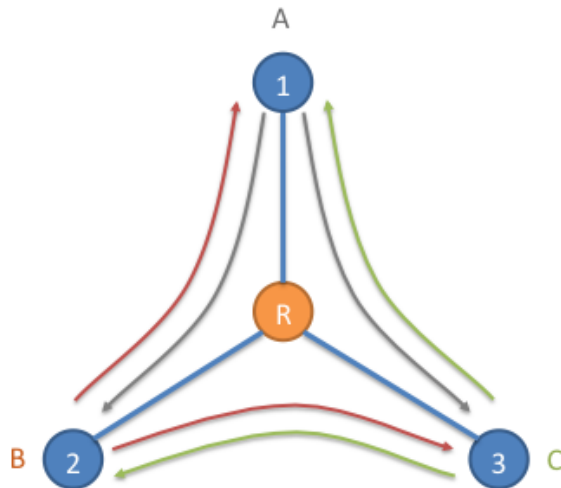


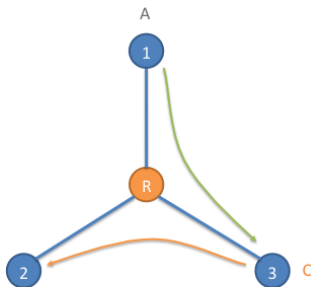
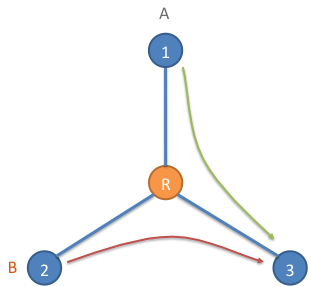
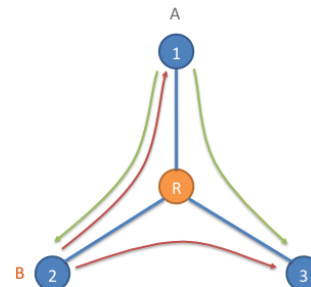
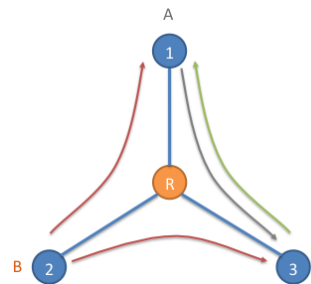
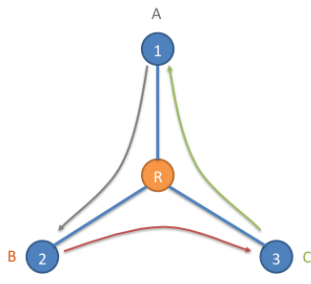
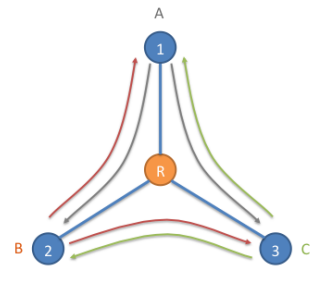
Figure 28: Three-Sender Y-Channel-Relay scenario (example 2)

Without network coding, and with Multiple Access protocol in place, the nodes have to take turns sending their packets to the relay one at a time, and then the relay forwards the packet to the other two nodes. Thus, six transmissions are required to complete the communication (Figure 25).

With digital network coding, the number of transmissions is reduced to only four as can be seen in Figure 26. With analog network coding, the number of transmissions is reduced even further to only two transmissions (Figure 27).

The following table summarizes the properties of Two-Sender and Three-Sender networks.

**Table 6: Summary of the properties and scenarios of Three-Sender and Two-Sender Y-Channel-Relay networks**

	Number of senders	Minimum number of receivers	Minimum number of received information packets	Maximum number of received information packets
Two-Sender Network	Always 2	1 Example: $1 \rightarrow 3$ & $3 \rightarrow 2$	2 Example: $1 \rightarrow 3$ & $2 \rightarrow 3$	4 Example: $1 \rightarrow 2$ & $1 \rightarrow 3$ $2 \rightarrow 1$ & $2 \rightarrow 3$
				
	Always 3	2 Example: $2 \rightarrow 1$ & $2 \rightarrow 3$ $1 \rightarrow 3$ $3 \rightarrow 1$	3 Example: $1 \rightarrow 2$ & $2 \rightarrow 3$ & $3 \rightarrow 1$	6 Example: $1 \rightarrow 2$ & $1 \rightarrow 3$ $2 \rightarrow 1$ & $2 \rightarrow 3$ $3 \rightarrow 1$ & $3 \rightarrow 2$
Three-Sender Network				

To be able to study the benefits of network coding in Y-Channel-Relay networks, we have to look at all the scenarios that involve three terminals communicating through a relay, whether that it is in a wireless mesh network, or in a multicast wireless transmission system. Table 7 goes in-depth into dissecting all these scenarios (whether they are Two-Way-Relay or Y-Channel-Relay problem) just to see or have an idea on how common Y-Channel-Relay situations are in general.

**Table 7: In depth look at all possible scenarios involving three terminals and a relay**

Node 1 sends to Node 2	Node 1 sends to Node 3	Node 2 sends to Node 1	Node 2 sends to Node 3	Node 3 sends to Node 1	Node 3 sends to Node 2	Number of transmitted info packets	Num of transmitting nodes	Number of involved nodes	Total Number Of Transmissions			Scenario Classification
									No Network Coding	Digital Network Coding	Analog (Physical) Network Coding	
No	No	No	No	No	No	0	0	0	N/A	N/A	N/A	
Yes	No	No	No	No	No	1	1	2	2	N/A	N/A	Point-to-Point / Routing
No	Yes	No	No	No	No	1	1	2	2	N/A	N/A	Point-to-Point / Routing
Yes	Yes	No	No	No	No	2	1	3	2	N/A	N/A	Point-to-Point / Routing
No	No	Yes	No	No	No	1	1	2	2	N/A	N/A	Point-to-Point / Routing
Yes	No	Yes	No	No	No	2	2	2	4	3	2	Two-way communication with relay
No	Yes	Yes	No	No	No	2	2	3	4	3	2	Two-Sender Network
Yes	Yes	Yes	No	No	No	3	2	3	4	3	2	Two-Sender Network
No	No	No	Yes	No	No	1	1	2	2	N/A	N/A	Point-to-Point / Routing
Yes	No	No	Yes	No	No	2	2	3	4	3	2	Two-Sender Network
No	Yes	No	Yes	No	No	2	2	3	4	3	2	Two-Sender Network (Special)
Yes	Yes	No	Yes	No	No	3	2	3	4	3	2	Two-Sender Network
No	No	Yes	Yes	No	No	2	1	3	2	N/A	N/A	Point-to-Point / Routing
Yes	No	Yes	Yes	No	No	3	2	3	4	3	2	Two-Sender Network
No	Yes	Yes	Yes	No	No	3	2	3	4	3	2	Two-Sender Network
Yes	Yes	Yes	Yes	No	No	4	2	3	4	3	2	Two-Sender Network
No	No	No	No	Yes	No	1	1	2	2	N/A	N/A	Point-to-Point / Routing
Yes	No	No	No	Yes	No	2	2	3	4	3	2	Two-Sender Network
No	Yes	No	No	Yes	No	2	2	2	4	3	2	Two-way communication with relay
Yes	Yes	No	No	Yes	No	3	2	3	4	3	2	Two-Sender Network
No	No	Yes	No	Yes	No	2	2	3	4	3	2	Two-Sender Network (Special)
Yes	No	Yes	No	Yes	No	3	3	3	6	4	2	Three-Sender Network
No	Yes	Yes	No	Yes	No	3	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	No	Yes	No	4	3	3	6	4	2	Three-Sender Network
No	No	No	Yes	Yes	No	2	2	3	4	3	2	Two-Sender Network
Yes	No	No	Yes	Yes	No	3	3	3	6	4	2	Three-Sender Network
No	Yes	No	Yes	Yes	No	3	3	3	6	4	2	Three-Sender Network
Yes	Yes	No	Yes	Yes	No	4	3	3	6	4	2	Three-Sender Network
No	No	Yes	Yes	Yes	No	3	2	3	4	3	2	Two-Sender Network
Yes	No	Yes	Yes	Yes	No	4	3	3	6	4	2	Three-Sender Network
No	Yes	Yes	Yes	Yes	No	4	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	Yes	Yes	No	5	3	3	6	4	2	Three-Sender Network
No	No	No	No	No	Yes	1	1	2	2	N/A	N/A	Point-to-Point / Routing
Yes	No	No	No	No	Yes	2	2	3	4	3	2	Two-Sender Network (Special)
No	Yes	No	No	No	Yes	2	2	3	4	3	2	Two-Sender Network
Yes	Yes	No	No	No	Yes	3	2	3	4	3	2	Two-Sender Network
No	No	Yes	No	No	Yes	2	2	3	4	3	2	Two-Sender Network
Yes	No	Yes	No	No	Yes	3	3	3	6	4	2	Three-Sender Network

No	Yes	Yes	No	No	Yes	3	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	No	No	Yes	4	3	3	6	4	2	Three-Sender Network
No	No	No	Yes	No	Yes	2	2	2	4	3	2	Two-way communication with relay
Yes	No	No	Yes	No	Yes	3	3	3	6	4	2	Three-Sender Network
No	Yes	No	Yes	No	Yes	3	3	3	6	4	2	Three-Sender Network
Yes	Yes	No	Yes	No	Yes	4	3	3	6	4	2	Three-Sender Network
No	No	Yes	Yes	No	Yes	3	2	3	4	3	2	Two-Sender Network
Yes	No	Yes	Yes	No	Yes	4	3	3	6	4	2	Three-Sender Network
No	Yes	Yes	Yes	No	Yes	4	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	Yes	No	Yes	5	3	3	6	4	2	Three-Sender Network
No	No	No	No	Yes	Yes	2	1	3	2	N/A	N/A	Point-to-Point / Routing
Yes	No	No	No	Yes	Yes	3	2	3	4	3	2	Two-Sender Network
No	Yes	No	No	Yes	Yes	3	2	3	4	3	2	Two-Sender Network
Yes	Yes	No	No	Yes	Yes	4	2	3	4	3	2	Two-Sender Network
No	No	Yes	No	Yes	Yes	3	2	3	4	3	2	Two-Sender Network
Yes	No	Yes	No	Yes	Yes	4	3	3	6	4	2	Three-Sender Network
No	Yes	Yes	No	Yes	Yes	4	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	No	Yes	Yes	5	3	3	6	4	2	Three-Sender Network
No	No	No	Yes	Yes	Yes	3	2	3	4	3	2	Two-Sender Network
Yes	No	No	Yes	Yes	Yes	4	3	3	6	4	2	Three-Sender Network
No	Yes	No	Yes	Yes	Yes	4	3	3	6	4	2	Three-Sender Network
Yes	Yes	No	Yes	Yes	Yes	5	3	3	6	4	2	Three-Sender Network
No	No	Yes	Yes	Yes	Yes	4	2	3	4	3	2	Two-Sender Network
Yes	No	Yes	Yes	Yes	Yes	5	3	3	6	4	2	Three-Sender Network
No	Yes	Yes	Yes	Yes	Yes	5	3	3	6	4	2	Three-Sender Network
Yes	Yes	Yes	Yes	Yes	Yes	6	3	3	6	4	2	Three-Sender Network

From the latter discussion, we can conclude that in order to complete communication for *Three-Sender Y-Channel-Relay* situation, we need:

- Six transmissions with Multiple Access Control and routing.
- Four transmissions with Digital Network Coding.
- Two transmissions with Analog (Physical) Network Coding.

For *Two-Sender Y-Channel-Relay* & *Two-Way-Relay* scenarios, we need:

- Four transmissions with Multiple Access Control and routing.
- Three transmissions with Digital Network Coding.
- Two transmissions with Analog (Physical) Network Coding.

It can be clearly seen that with the use of network coding, we can improve network performance by up to three folds. More realistically, however, if we assume that each of the possible scenarios described in Table 7 has equal chances of it happening, then the average number of transmissions required for communication goes down from 4.57 transmissions with routing only to 3.29

transmissions with digital network coding, and only 2 with the addition of analog network capabilities. If we limit the discussion to Two-Sender and Three-Sender scenarios, the average number of transmissions goes down from 4.76 transmissions without network coding to 3.51 and 2 when using digital and analog (physical) network coding respectively. While practical systems proposed in [5] [19] harness network coding, their benefits are limited to increasing network performance by a maximum of two folds. However, by expanding those systems to include Y-Channel-Relay scenarios, we can reduce the number of required transmissions by 2.4 times.

With that said, a better indication of how much network coding increases performance is the rate of real information bits that can be sent through the system in one second (i.e. Capacity). In the following section, we will derive the capacity limits for Two-Sender and Three-Sender networks with and without network coding.

Most researchers focused on analyzing the Two-Way-Relay communication problem. However, a few proposals include a joint network coding and superposition coding for three-user relay channels, in which two-stage operations are required for encoding and decoding [29]. We will be looking at this approach and others in section 2.5. It is interesting to note that the analysis of the Y-Channel-Relay communication problem can be extended to N-way relay problem – where N terminals communicate through a common relay.

## **2.4. The Hybrid Error-Erasure Binary Channel**

The hybrid error –erasure binary channel is one where an input bit is susceptible to both erasures and errors. [25] Figure 29 shows such a channel:

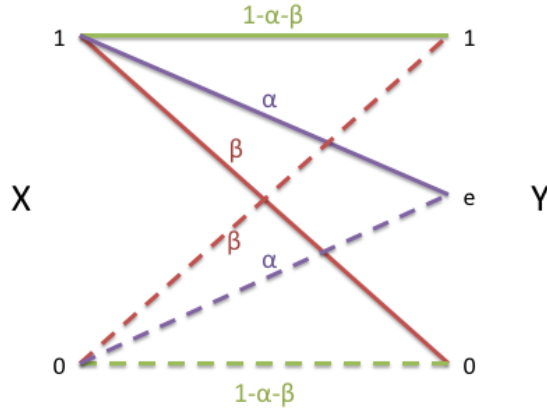


Figure 29: The hybrid error-erasure binary channel

This particular channel has two discrete inputs  $X \in \{0, 1\}$  and three probable discrete outputs  $Y \in \{0, e, 1\}$ , where  $e$  represents an erasure, and assuming the following:

$$P(Y = e|X = 0) = P(Y = e|X = 1) = \alpha = P(\text{bit is erased}) \quad (2.1)$$

$$P(Y = 1|X = 0) = P(Y = 0|X = 1) = \beta = P(\text{bit in error}) \quad (2.2)$$

$$P(Y = 1|X = 1) = P(Y = 0|X = 0) = 1 - \alpha - \beta = P(\text{bit received correctly}) \quad (2.3)$$

$$P(\text{bit is received with error} | \text{no erasure}) = p = \frac{\beta}{1 - \alpha} \quad (2.4)$$

Equation (2.4) his can also be called the conditional bit error probability.

The capacity for such channel is equal to [16]:

$$C = (1 - \alpha)(1 - h_b(p)), \quad (2.5)$$

Where  $h_b(p) = -p \log p - (1 - p) \log(1 - p)$  is the binary entropy function. Thus, the hybrid binary symmetric channel can be represented with erasure probability  $\alpha$  and conditional error probability  $p$  as  $HBSC(\alpha, p)$ . In a noiseless channel and 0.5 probability of erasure. The capacity would be:

$$C = 0.5(1 - h_b(0)) = 0.5 \quad (2.6)$$



In the following sub-sections, we will derive the capacity of different scenarios related to the Y-Channel-Relay problem with and without network coding. The following assumptions are made in this discussion:

- For simplicity, we assume we have a noiseless channel and that bit error rate  $BER \approx 0$ . I.e.  $P_b = 0$ .
- The used modulation scheme is BPSK, and the system is capable of sending one transmission every second (1 bit/channel use). I.e. the bit rate at which each node generates and transmits a packet is 1 *bit/sec*.
- Each node can only receive from the relay. No side information is available in any of the receiving nodes.
- The relay and the receiving nodes have full knowledge of fading and channel coefficients, and account for that.

#### **2.4.1. Capacity in Two-Way-Relay Scenario**

We discussed the Two-Way-Relay problem in Section 2.2.1, and we saw that without network coding four transmissions are needed to complete two-way communication. Since there are no erasures, and with the assumption of a noiseless channel, each node can send at a maximum rate of 1. The following summarizes the derivation of capacity of this scenario:

- The capacity of Node 1 (User 1) is 1 bit per channel use. [30]
- The capacity of Node 2 (User 2) is 1 bit per channel use.
- Without routing, four transmissions are needed. If each transmission takes 1 second, we have a total of two bits sent over 4 seconds. That is 0.5 information bits/sec.

With Digital network coding, the number of required transmissions is reduced to three:

- The maximum capacity of Node 1 (User 1) is 1 bit per channel use.
- The maximum capacity of Node 2 (User 2) is 1 bit per channel use.

- The relay transmits a mixed signal that is the result of a modulo-2 operation of the packets received from Node 1 and Node 2. Upon the reception of this signal, each node can subtract its information bit, and figure out the other user's information bit. Hence, we have two information bits sent over 3 seconds. That is 0.67 information bits/sec.

With Analog (Physical) network coding, the number of required transmissions is reduced to two:

- The maximum capacity of Node 1 (User 1) is 1 bit per channel use.
- The maximum capacity of Node 2 (User 2) is 1 bit per channel use.
- We have two information bits sent over 2 seconds. That is 1 information bit/sec.

In this case, the use of physical network coding doubles the capacity of the system when compared to classical wireless routing schemes. [5] [4]

#### **2.4.2. Capacity in Two-Sender Y-Channel-Relay Scenario with One Common Receiver**

The scenario that is going to be discussed here was looked at in Section 2.3.1. Without network coding, we saw that we need four transmissions to complete communication. The maximum capacity in this case is as follows:

- The capacity of Node 1 (User 1) is 1 bit per channel use.
- The capacity of Node 2 (User 2) is 1 bit per channel use.
- We have two information bits sent over 4 seconds. That is 0.5 information bits/sec.

Now let's consider the case where network coding is used. The system can be described by Figure 30. Since the common receiver (Node 3 in this case) doesn't have any prior knowledge of either  $X_1$  or  $X_2$ , the problem of decoding the received signal at decoder 3 can be considered a binary erasure channel problem with 50% chance of having an erased bit.

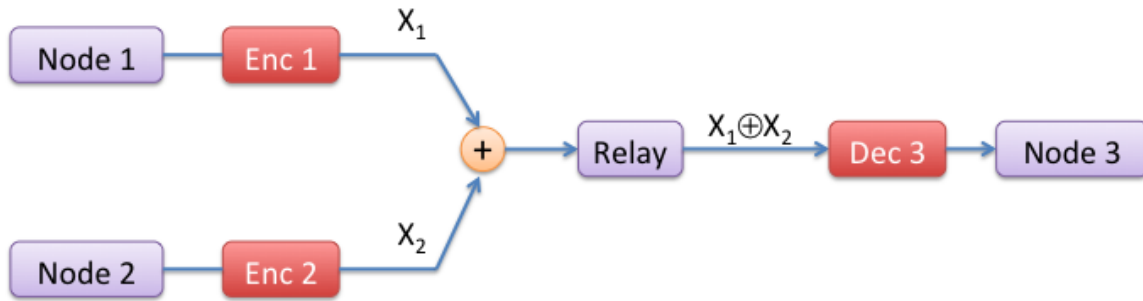


Figure 30: Transmission and decoding in a Two-Sender Y-Channel-Relay scenario with one common receiver

In a noiseless channel, the capacity of an erasure binary channel is 0.5 bits/channel use. This, either  $X_1$  or  $X_2$  need to be encoded of at least a code of rate 0.5 to account for the erased bits. Let's consider for example that  $X_1$  was encoded with a code of rate 0.5 that will guarantee successful decoding of  $X_1$  at Node 3. In this case, Node 3 will decode  $X_1$  first, then subtract the decoded signal from  $X_1 \oplus X_2$  to get  $X_2$ . Thus, in the case of digital network coding, we will have:

- The maximum capacity of Node 1 (User 1) is 0.5 information bits/sec.
- The maximum capacity of Node 2 (User 2) is 1 information bits/sec.
- We have 1.5 information bits sent over 3 seconds. That is 0.5 information bits/sec.

When analog network coding is used:

- The maximum capacity of Node 1 (User 1) is 0.5 information bits/sec.
- The maximum capacity of Node 2 (User 2) is 1 information bits/sec.
- We have 1.5 information bits sent over 2 seconds. That is 0.75 information bits/sec.

We can see that analog network coding in this case increases the capacity of the system by 50%.

Now, let's consider the maximum theoretical possible capacity scenario for a Two-Sender network where we have two senders and three receivers.

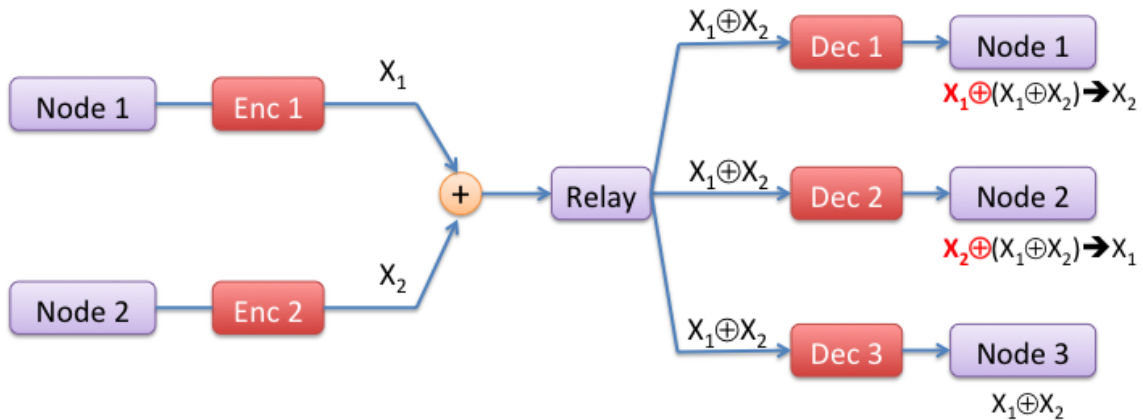


Figure 31: Transmission and decoding in the maximum configuration of a Two-Sender network with two sender and three receivers

In this case, with digital network coding:

- The maximum capacity of Node 1 (User 1) is 0.5 information bits/sec.
- The maximum capacity of Node 2 (User 2) is 1 information bits/sec.
- After three seconds, Node 1 decodes the message of Node 2 and has 1 bit, Node 2 decodes the message of Node 1 and has 0.5 bits, and Node 3 decodes both Node 1 and Node 2 packets and has 1.5 bits. In total, 3 information bits were sent over 3 seconds. That is 1 information bits/sec.

And with analog network coding:

- The maximum capacity of Node 1 (User 1) is 0.5 information bits/sec.
- The maximum capacity of Node 2 (User 2) is 1 information bits/sec.
- After two seconds, Node 1 decodes the message of Node 2 and has 1 bit, Node 2 decodes the message of Node 1 and has 0.5 bits, and Node 3 decodes both Node 1 and Node 2 packets and has 1.5 bits. In total, 3 information bits were sent over 2 seconds. That is 1.5 information bits/sec.

### 2.4.3. Capacity in Three-Sender Y-Channel-Relay Scenarios

Let's now consider the maximum Three-Sender Y-Channel-Relay scenario where each sender wants to send a packet to the other two nodes. Without network coding, and with routing and multiple-access scheme in place, capacity is as follows:

- The maximum capacity of Node 1 (User 1) is 1 information bit/sec.
- The maximum capacity of Node 2 (User 2) is 1 information bit/sec.
- The maximum capacity of Node 3 (User 3) is 1 information bit/sec.
- Six transmissions are needed to complete communication. The total number of information bits received at all nodes is six. That is 1 information bit/sec.

Now let's consider capacity of the system when using network coding. The transmission and decoding flow is represented in Figure 32. Each receiver receives  $X_1 \oplus X_2 \oplus X_3$ , and upon reception, subtracts its message to have a modulo-2 mixed signal that requires a rate of at least 1.5 to be decoded successfully. If we assume that the rate at Node 1 and Node 2 is 1 and 0.5 respectively. Then:

- At Node 3,  $X_1 \oplus X_2$  will be decoded successfully.
- At Node 2,  $X_1 \oplus X_3$  must have a rate that is at least 1.5. Hence, the maximum rate for Node 3 should be at least 0.5 to guarantee successful decoding.
- At Node 1,  $X_2 \oplus X_3$  will be decoded successfully since each has a rate 0.5.  $X_2$  is decoded first using the rate 0.5 code. After that,  $X_3$  is decoded.

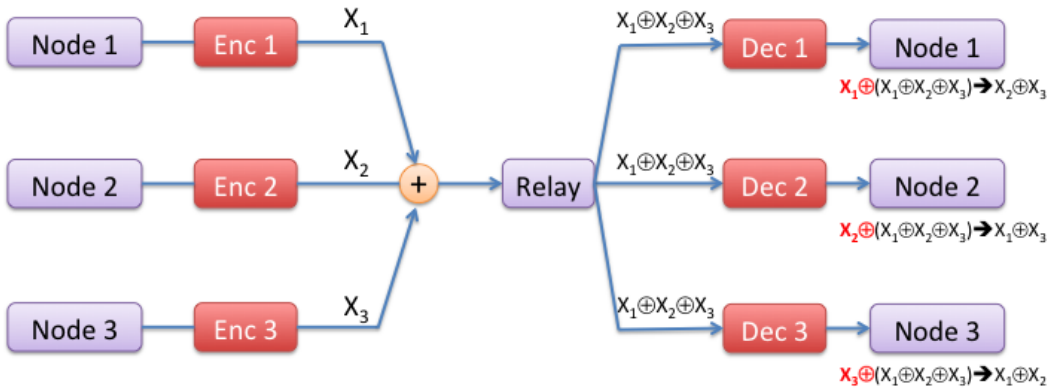


Figure 32: Transmission and decoding in Three-Sender Y-Channel-Relay scenario

Hence, when digital network coding is used:

- The maximum capacity of Node 1 (User 1) is 1 information bit/sec.
- The maximum capacity of Node 2 (User 2) is 0.5 information bits/sec.
- The maximum capacity of Node 3 (User 3) is 0.5 information bits/sec.
- Four transmissions are required to complete communication. Thus, we have 4/4 information bits = 1 information bit/sec

When analog (physical) network coding is used, the number of transmissions is reduced to only 2. Thus, the capacity of the system increases to 2 information bits/sec.

Table 8 summarizes the capacity for Two-Way-Relay scenarios.

Scenario	Capacity & Rate		
	Routing and multiple access	Digital Network Coding	Analog (Physical) Network Coding
<b>Example:</b> $1 \rightarrow 2$ & $2 \rightarrow 1$	0.5 Info Bits/Sec	0.667 Info Bits/Sec	1 Info Bit/Sec

Table 9 summarizes the capacity for different Two-Sender Y-Channel-Relay scenarios.

**Table 9: Capacity of Two-Sender Y-Channel-Relay Scenarios**

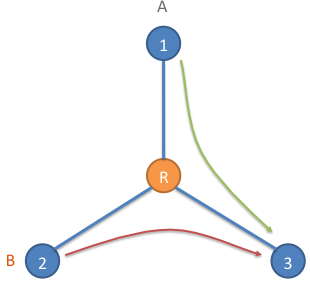
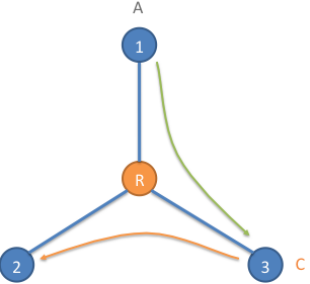
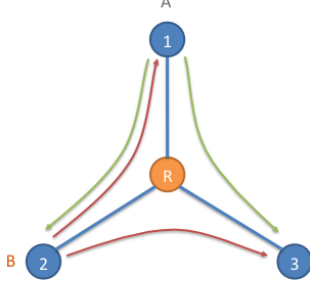
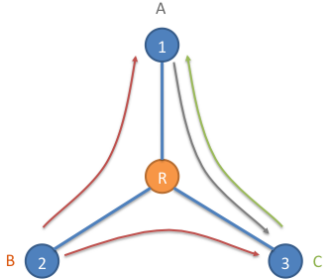
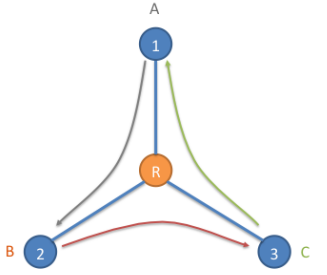
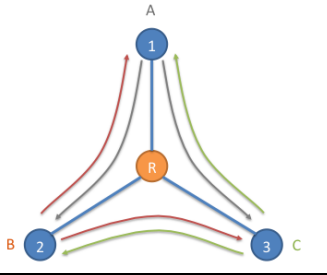
Scenario	Capacity & Rate		
	Routing and multiple access	Digital Network Coding	Analog (Physical) Network Coding
<p><b>Example:</b>  <math>1 \rightarrow 3</math> &amp; <math>2 \rightarrow 3</math></p> 	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=1$ Bit/Sec  0.5 Info Bits/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=0.5$ Bit/Sec  0.5 Info Bits/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=0.5$ Bit/Sec  0.75 Info Bits/Sec
<p><b>Example:</b>  <math>1 \rightarrow 3</math> &amp; <math>3 \rightarrow 2</math></p> 	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 3}}=1$ Bit/Sec  0.5 Info Bits/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 3}}=0.5$ Bit/Sec  0.5 Info Bits/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 3}}=0.5$ Bit/Sec  0.75 Info Bits/Sec
<p><b>Example:</b>  <math>1 \rightarrow 2</math> &amp; <math>1 \rightarrow 3</math>  <math>2 \rightarrow 1</math> &amp; <math>2 \rightarrow 3</math></p> 	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=1$ Bit/Sec  1 Info Bit/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=0.5$ Bit/Sec  1 Info Bit/Sec	$R_{\text{Node 1}}=1$ Bit/Sec $R_{\text{Node 2}}=0.5$ Bit/Sec  1.5 Info Bit/Sec

Table 10 summarizes the capacity of different Three-Sender Y-Channel-Relay scenarios.

**Table 10: Capacity of Three-Sender Y-Channel-Relay Scenarios**

	Capacity & Rate		
	Routing and multiple access	Digital Network Coding	Analog (Physical) Network Coding
<p><b>Example:</b>  <math>2 \rightarrow 1</math> &amp; <math>2 \rightarrow 3</math>  <math>1 \rightarrow 3</math>  <math>3 \rightarrow 1</math></p> 	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=1$ Bit/Sec $R_{Node\ 3}=1$ Bit/Sec  0.667 Info Bit/Sec	$R_{Node\ 1}=0.5$ Bit/Sec $R_{Node\ 2}=1$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  0.75 Info Bit/Sec	$R_{Node\ 1}=0.5$ Bit/Sec $R_{Node\ 2}=1$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  1.5 Info Bit/Sec
<p><b>Example:</b>  <math>1 \rightarrow 2</math> &amp; <math>2 \rightarrow 3</math> &amp; <math>3 \rightarrow 1</math></p> 	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=1$ Bit/Sec $R_{Node\ 3}=1$ Bit/Sec  0.5 Info Bit/Sec	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=0.5$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  0.5 Info Bit/Sec	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=0.5$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  1 Info Bit/Sec
<p><b>Example:</b>  <math>1 \rightarrow 2</math> &amp; <math>1 \rightarrow 3</math>  <math>2 \rightarrow 1</math> &amp; <math>2 \rightarrow 3</math>  <math>3 \rightarrow 1</math> &amp; <math>3 \rightarrow 2</math></p> 	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=1$ Bit/Sec $R_{Node\ 3}=1$ Bit/Sec  1 Info Bit/Sec	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=0.5$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  1 Info Bit/Sec	$R_{Node\ 1}=1$ Bit/Sec $R_{Node\ 2}=0.5$ Bit/Sec $R_{Node\ 3}=0.5$ Bit/Sec  2 Info Bit/Sec

In section 2.4.4, we will analyze the Y-Channel bound on performance without any type of coding.



#### 2.4.4. Y-Channel-Relay Analysis without Coding

In the Y-Channel-Relay problem, in order to separate the signal at the receiver without using any coding schemes, each sender has to send with a certain energy level in such a way that the relay can separate the signal sent by each terminal using eight decision regions. Consequently, each node in this system is viewed as a binary source where Node 1, Node 2, and Node 3 transmit using a BPSK modulation scheme at energy levels  $\alpha$ ,  $\beta$ , and  $\gamma$  respectively:

- Node 1 can be looked at as a binary source;  $\mathbf{X}_1 = \{-\alpha, +\alpha\}, \alpha > 0$
- Node 2 can be looked at as a binary source;  $\mathbf{X}_2 = \{-\beta, +\beta\}, \beta > 0$
- Node 3 can be looked at as a binary source;  $\mathbf{X}_3 = \{-\gamma, +\gamma\}, \gamma > 0$

The relay will receive a combined analog signal over the air that is equal to the following:

$$R = \mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 \quad (2.7)$$

The energy levels are selected in a certain way so that any combination of  $X_1$ ,  $X_2$  and  $X_3$  produces a unique level at the relay (and subsequently at the receiver) so that the signal can be separated based on a decision region. The following diagram shows such energy level scheme along with the decision regions at the receiver.

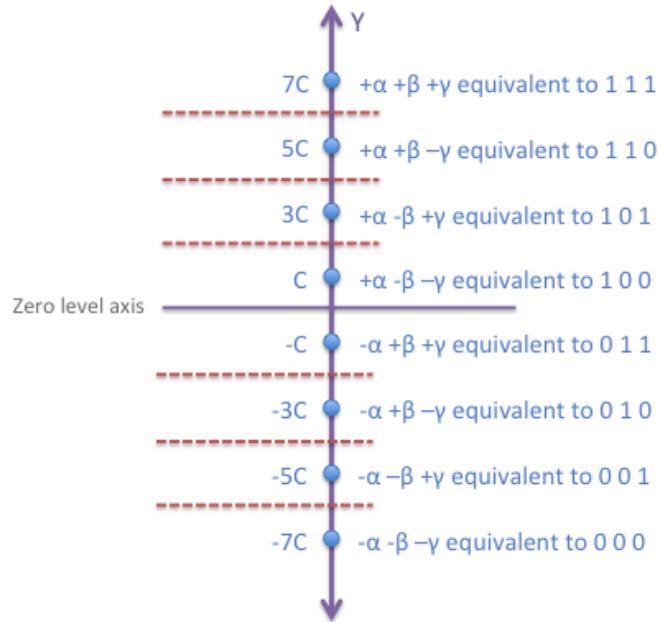


Figure 33: Decision regions on the receiver side

The latter un-coded example is very similar to a three-bit point-to-point amplitude modulation scheme. While amplitude modulation schemes usually use grey coded mappings to maximize the possible performance of the system, such mapping is not possible in the Y-Channel-Relay with analog network coding problem. The reason for that is because  $\alpha, \beta, & \gamma$  cannot be chosen in a way that allow equi-probable mapping of decision regions for the mixed signal.

As an example, let's assume that Node 1, Node 2 and Node 3 want to send [01010101, 11100001 and 11000110] respectively. The transmitted levels are shown below:

Table 11: Message of node 1

Message bits	0	1	0	1	0	1	0	1
Transmitted level	$-\alpha$	$+\alpha$	$-\alpha$	$+\alpha$	$-\alpha$	$+\alpha$	$-\alpha$	$+\alpha$

Table 12: Message of node 2

Message bits	1	1	1	0	0	0	0	1
Transmitted level	$+\beta$	$+\beta$	$+\beta$	$-\beta$	$-\beta$	$-\beta$	$-\beta$	$+\beta$

Table 13: Message of node 3

Message bits	1	1	0	0	0	1	1	0
Transmitted level	$+\gamma$	$+\gamma$	$-\gamma$	$-\gamma$	$-\gamma$	$+\gamma$	$+\gamma$	$-\gamma$

Based on that, at the end of timeslot 1 of the transmission, the relay will have received a combined signal that is equal to the following:

**Table 14: Received signal at relay**

Bit	1	2	3	4	5	6	7	8
Received level	$-\alpha + \beta$ $+\gamma$	$+\gamma + \beta$ $+\gamma$	$-\gamma + \beta$ $+\gamma$	$-\gamma - \beta$ $+\gamma$	$-\gamma - \beta$ $+\gamma$	$+\gamma - \beta$ $+\gamma$	$+\gamma - \beta$ $+\gamma$	$-\gamma + \beta$ $+\gamma$

Once the relay receives the signal, it can either decode it based on the eight decision regions shown in Figure 33, or simply amplify it and transmit it back to all nodes in the second transmission time slot. Each receiver, in turn, decodes the signal based on the received signal and the eight decision regions.

#### ***2.4.4.1. Calculating The Upper Bound of BER For The Un-Coded Solution***

In this section, the upper bound of BER for the un-coded solution will be derived. We will use the example in the previous section as a basis for the derivation. The following assumptions are taken into consideration:

- We have three nodes and a relay in the middle. Each node has a continuous stream of bits that needs to be multi-casted to the other two nodes.
- The nodes are distant from each other, and direct communication between the nodes is not possible. Communication is only possible through the relay. The relay receives, amplifies, and forwards an analog mixed signal without trying to perform any kind of decoding or demodulation.
- Reception is synchronized. That means the relay receives bits from the three nodes at exactly the same time.
- Noise in phase 1 of the transmission (from the nodes to the relay) is assumed to be white Gaussian noise  $\mathcal{N}_1 \sim (0, \sigma_1)$ . Noise in phase 2 is assumed to be white Gaussian noise  $\mathcal{N}_2 \sim (0, \sigma_2)$ .

- It is assumed that an amplify-and-forward approach is used by the relay to forward the mixed signal. This means that noise can be treated as  $\mathcal{N} = \mathcal{N}_1 + \mathcal{N}_2 \sim (0, \sigma = \sigma_1 + \sigma_1) \rightarrow \mathcal{N} \sim (0, \sigma)$ .
- Fading is not taken into consideration during the derivation of the upper bound. It is assumed that the destination knows the fading of the channel, and accounts for that.
- The relay doesn't do any decoding. Noise is taken end to end.
- The nodes transmit with the following level:
  - Node 1 transmits with levels  $\{+\alpha, -\alpha\}$  where  $\{+\sqrt{E_{b_1}}, -\sqrt{E_{b_1}}\} = 4C$
  - Node 2 transmits with levels  $\{+\beta, -\beta\}$  where  $\{+\sqrt{E_{b_2}}, -\sqrt{E_{b_2}}\} = 2C$
  - Node 3 transmits with levels  $\{+\gamma, -\gamma\}$  where  $\{+\sqrt{E_{b_3}}, -\sqrt{E_{b_3}}\} = C$

When decoding the received message at node 1, we have the following bit error probabilities for node 1's message in the eight decision regions:

$$P(Y_1 = 0 | X_1 = 1, X_2 = 1, X_3 = 1) \leq Q\left(\frac{7C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.8)$$

$$P(Y_1 = 0 | X_1 = 1, X_2 = 1, X_3 = 0) \leq Q\left(\frac{5C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.9)$$

$$P(Y_1 = 0 | X_1 = 1, X_2 = 0, X_3 = 1) \leq Q\left(\frac{3C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.10)$$

$$P(Y_1 = 0 | X_1 = 1, X_2 = 0, X_3 = 0) \leq Q\left(\frac{C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.11)$$

$$P(Y_1 = 1 | X_1 = 0, X_2 = 0, X_3 = 0) \leq Q \left( \frac{7C}{\sqrt{\frac{N_o}{2}}} \right) \quad (2.12)$$

$$P(Y_1 = 1 | X_1 = 0, X_2 = 0, X_3 = 1) \leq Q \left( \frac{5C}{\sqrt{\frac{N_o}{2}}} \right) \quad (2.13)$$

$$P(Y_1 = 1 | X_1 = 0, X_2 = 1, X_3 = 0) \leq Q \left( \frac{3C}{\sqrt{\frac{N_o}{2}}} \right) \quad (2.14)$$

$$P(Y_1 = 1 | X_1 = 0, X_2 = 1, X_3 = 1) \leq Q \left( \frac{C}{\sqrt{\frac{N_o}{2}}} \right) \quad (2.15)$$

From the previous equations, the probability of bit error rate for Node 1's packet will be:

$$\begin{aligned} &P(\text{error in Node 1's bit}) \\ &\leq \left(\frac{1}{4}\right) Q \left( \frac{7C}{\sqrt{\frac{N_o}{2}}} \right) + \left(\frac{1}{4}\right) Q \left( \frac{5C}{\sqrt{\frac{N_o}{2}}} \right) + \left(\frac{1}{4}\right) Q \left( \frac{3C}{\sqrt{\frac{N_o}{2}}} \right) \\ &+ \left(\frac{1}{4}\right) Q \left( \frac{C}{\sqrt{\frac{N_o}{2}}} \right) \end{aligned} \quad (2.16)$$

Similarly, the probability of error for Node 2's bit is:

$$P(\text{error in Node 2's bit}) \leq \left(\frac{1}{4}\right) Q \left( \frac{3C}{\sqrt{\frac{N_o}{2}}} \right) + \left(\frac{3}{4}\right) Q \left( \frac{C}{\sqrt{\frac{N_o}{2}}} \right) \quad (2.17)$$

And for Node 3:

$$P(\text{error in Node 3's bit}) \leq Q\left(\frac{C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.18)$$

The overall average bit error rate will be:

$$BER = \frac{1}{3}(P(\text{error in node 1's bit}) + P(\text{error in node 2's bit}) + P(\text{error in node 2's bit})) \quad (2.19)$$

$$BER \leq \left(\frac{1}{12}\right)Q\left(\frac{7C}{\sqrt{\frac{N_o}{2}}}\right) + \left(\frac{1}{12}\right)Q\left(\frac{5C}{\sqrt{\frac{N_o}{2}}}\right) + \left(\frac{1}{6}\right)Q\left(\frac{3C}{\sqrt{\frac{N_o}{2}}}\right) + \frac{2}{3}Q\left(\frac{C}{\sqrt{\frac{N_o}{2}}}\right) \quad (2.20)$$

The theoretical BER performance curve is shown below:

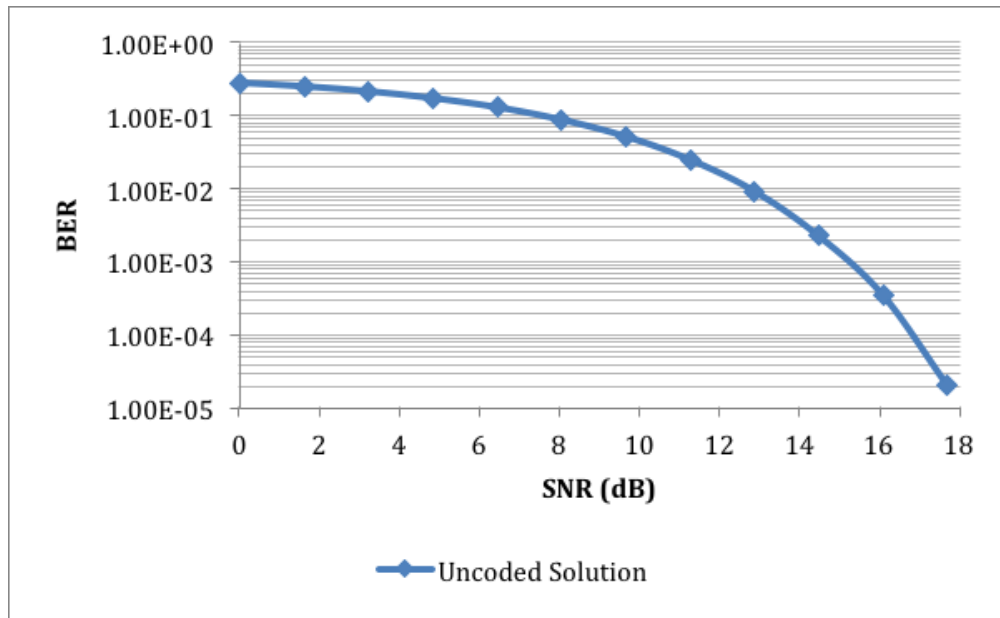


Figure 34: Theoretical BER performance curve of the uncoded solution

The previous equations don't take into consideration that the receiver already knows its own message, and thus, it can deduct its message before decoding the signal itself. This can be utilized to get a tighter bound.

#### 2.4.4.2. *Simulation*

The uncoded solution to the three-node with relay communication problem was simulated using a decode and forward approach. Three equi-probable binary sources generate an endless stream of bits. Each node is assigned a BPSK modulator with certain energy level( $E_b$ ). The initial energy level is chosen to give an SINR of 0 dB. Once a predetermined confidence value for  $P_b$  is earned from the simulating at that SINR level,  $E_b$  is changed to obtain a higher SINR, and the process is repeated to get the  $P_b$  curve.

For each SINR level, the following is done to obtain the  $P_b$ :

- Produce a continuous stream of bits from the three binary sources.
- Modulate the bits as per each node's modulator.
- Combine the relative output from each node, and add AWGN to it based on the chosen  $E_b$  and  $N_0$  to get the desired SINR. This simulates mixing the signals in the air at the relay.
- The received signal is then demodulated at the receiver.

Here is the curve for the  $P_b$  when simulating the uncoded solution at node 3:

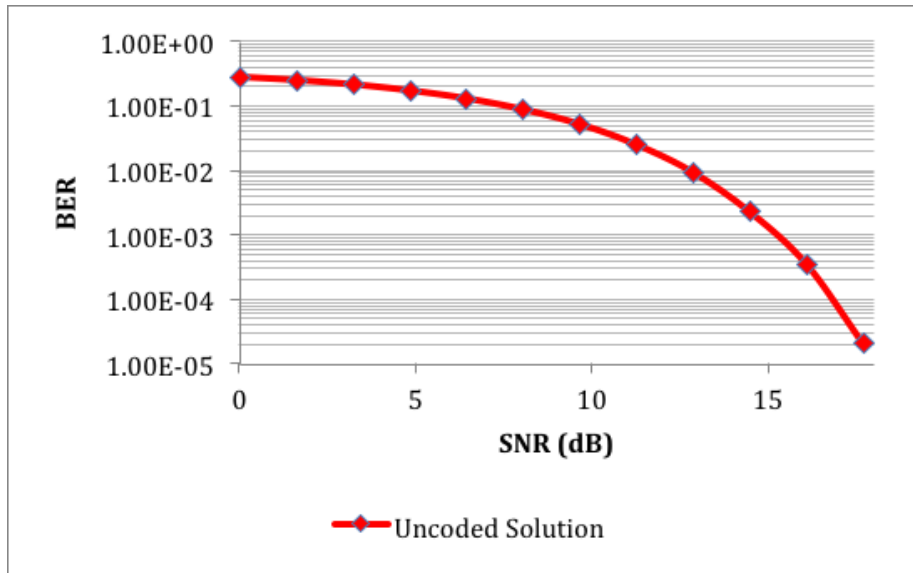


Figure 35: The simulated BER curve for the uncoded solution

The simulated curve is identical to the calculated theoretical curve.

At first glance, one might observe that the uncoded solution for the Y-Channel-Relay problem with ANC has worse performance than point-to-point BPSK. However, when looking from a higher layer perspective, using ANC reduces the number of packet transfers in a wireless mesh network from six transmissions down to two.

## 2.5. Existing Coding Techniques for The Y-Channel-Relay Problem

Physical network coding has been a hot topic in recent research. In this section, we will look at existing solutions to the Y-Channel-Relay problem in the literature.

### 2.5.1. Nested Codes

Nested codes were first proposed in [1] as a new approach to channel code design in networks where Analog (Physical) network coding is considered. In these codes, packets can be decoded in multiple ways at the receiver depending on how much prior information it knows. If a receiver has information for one or more of the received XORed packets, then those known packets can be subtracted by regarding them as “scrambling” patterns. Thus, the receiver decodes the



unknown packets at a lower effective rate. On the other hand, if the receiver has no prior information of any of the received XORed packets, it considers those codewords to be produced by a higher-rate “nested” code. With this in mind, each receiver can decode the received packets differently based on his prior knowledge of one or more of the XORed codewords. In this thesis, the worst case scenario is considered, where none of the recipients has a priori knowledge of the received message from the relay, except for its information sequence.

### 2.5.1.1. Encoding

If we assume we have  $N$  users that want to communicate through a relay using Physical (Analog) Network coding, each sender encodes its information vector using a separate low rate  $\frac{R}{N} = k/(nN)$  that would be XORed in the air. I.e. when excluding noise and fading, codeword received at the relay would be:

$$\begin{aligned}
 c &= i_1 G_1 \oplus i_2 G_2 \oplus \dots \oplus i_N G_N \\
 &= [i_1, i_2, \dots, i_N] \begin{bmatrix} G_1 \\ G_2 \\ \dots \\ G_N \end{bmatrix}, \tag{2.21}
 \end{aligned}$$

where  $G_1, G_2, \dots, G_N$  are generator matrices of rate  $k/(nN)$ , and  $i_1, i_2, \dots, i_N$  are information vectors for Nodes 1, 2,  $\dots$ ,  $N$ .

It is worth noting here that using different linearly independent generator matrices at each sender is crucial for getting the maximum performance out of the system. A consequence of having two or more identical generator matrices will cause the coding to be noninvertible at the decoder. This is called matrix rank deficiency.

### 2.5.1.2. Decoding

$$c = \underbrace{\bigoplus_{l \notin \mathcal{K}_j} i_l G_l}_{c_u} \oplus \underbrace{\bigoplus_{l' \in \mathcal{K}_j} i_{l'} G_{l'}}_{c_c}, \quad (2.22)$$

where  $c_u$  is a collection of unknown codewords,  $c_c$  is a collection of known codewords at the  $j$ -th receiver,  $G_l$  is the generator matrix of the  $l$ -th user, and  $\mathcal{K}_j$  represents the indices of known information packets to the receiver. Since the receiver already knows  $c_c$ , it can subtract that signal and decode  $c_u$ . The Log-Likelihood-Ratio of the  $i$ -th bit is calculated as follows:

$$L_{c_u(i)} \triangleq \log \frac{\Pr[c_u(i) = 0]}{\Pr[c_u(i) = 1]} \quad (2.23)$$

$$= \begin{cases} L_{c(i)} = \log \frac{\Pr[c_u \oplus c_c(i) = 0]}{\Pr[c_u \oplus c_c(i) = 1]} & \text{if } c_c(i) = 0 \\ -L_{c(i)} = \log \frac{\Pr[c_u \oplus c_c(i) = 1]}{\Pr[c_u \oplus c_c(i) = 0]} & \text{if } c_c(i) = 1 \end{cases}$$

The previous LLR operation is referred to as the “flipping” operation.

The “Broadcast Infrastructure Aided Multicasting” example described in [1] is identical to the Two-Sender Y-Channel-Relay scenario where two senders are sending to a common receiver Figure 15. In it, Node 1 uses a 64-state rate 1/3 convolutional code with  $G_1 = [06 \ 13 \ 13]_8$  and Node 2 uses another 64-state 1/3 convolutional code with  $G_2 = [13 \ 06 \ 17]_8$  [31]. The overall received codeword at the receiver can be considered as a “stacked” 2/3 convolutional code with  $G_{overall} = \begin{bmatrix} 06 & 13 & 13 \\ 13 & 06 & 17 \end{bmatrix}_8$ . The decoder uses the flipping operation to decode the messages simultaneously and flips the LLR for each erased bit. The following is simulation results for an AWGN channel. It is assumed that the receiver doesn’t have any a priori information of the received packets.

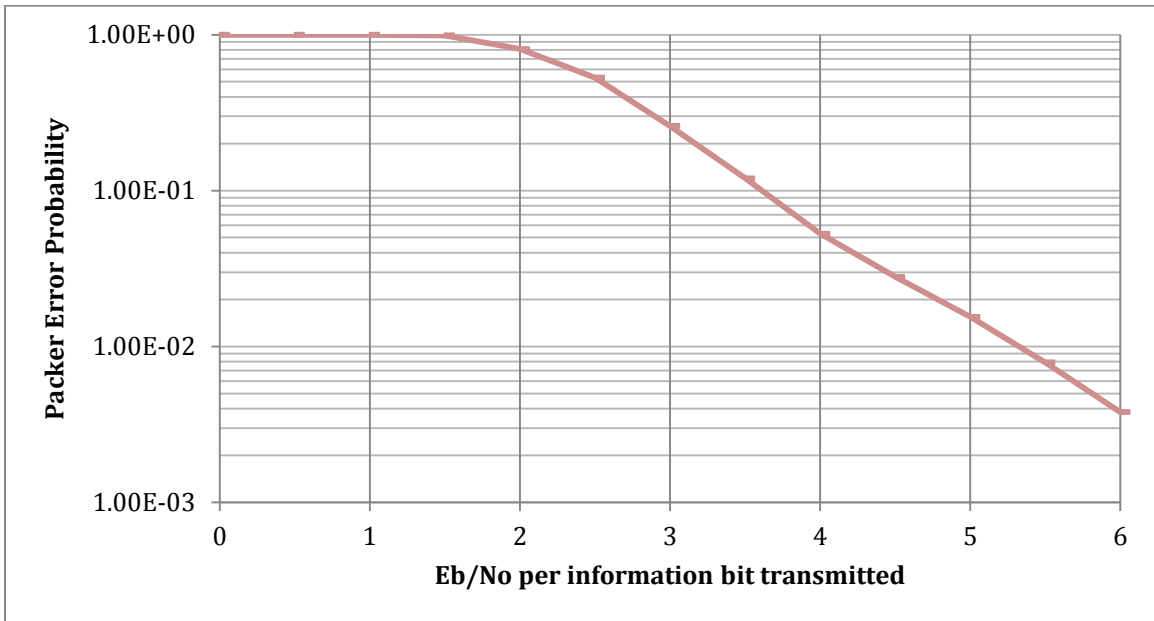


Figure 36: Packet Error Rate when using nested codes in Two-Sender Network with two senders and a common receiver

Since each sender sends with rate 1/3, the maximum throughput of the system in information bits/sec would be 0.667 information bits in two seconds (0.333 information bits/sec) as seen in Figure 37.

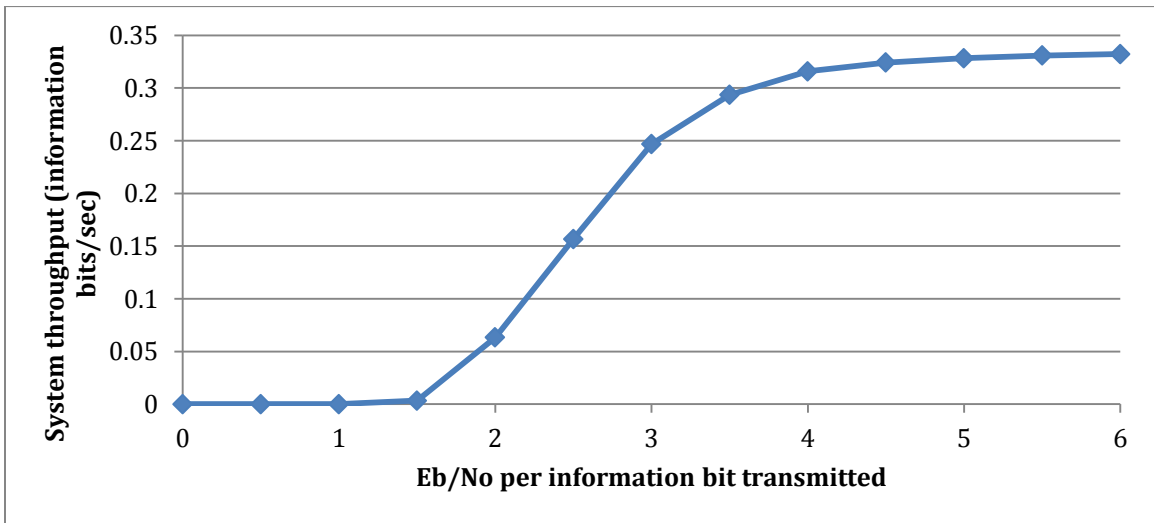


Figure 37: Throughput (information bits/sec) when using Nested Codes in Two-Sender Y-Channel-Relay problem with two senders and one common receiver.

We can adapt the upper solution to work with a Three-Sender Y-Channel-Relay networks and analog network coding by adding a third generator polynomial to the mix  $G_3 = [17 \ 17 \ 06]_8$ . Each node still sends using a rate 1/3 convolutional encoder and the effective rate will still be 2/3 at each receiver after omitting its own message. The overall system throughput will increase to 1 info bit/sec when each node wants to send the same packet to the other two nodes.

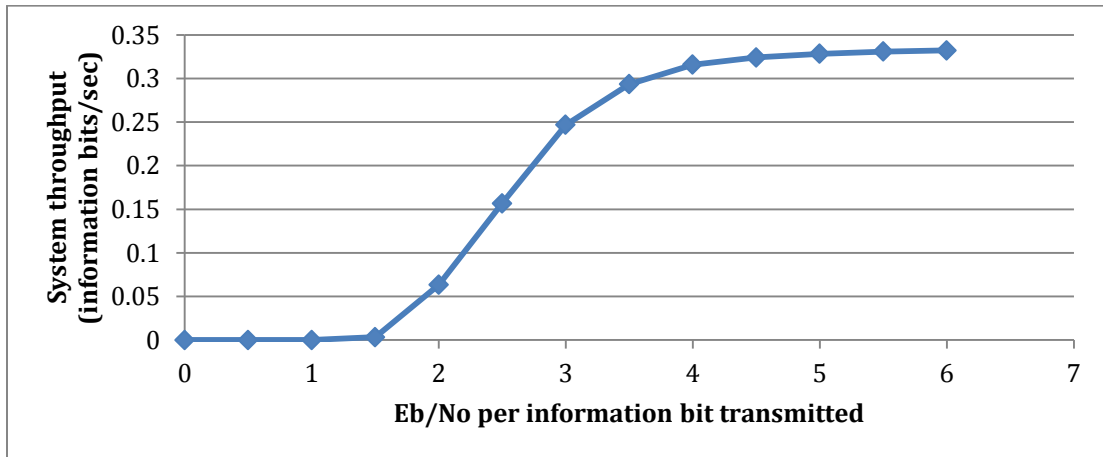
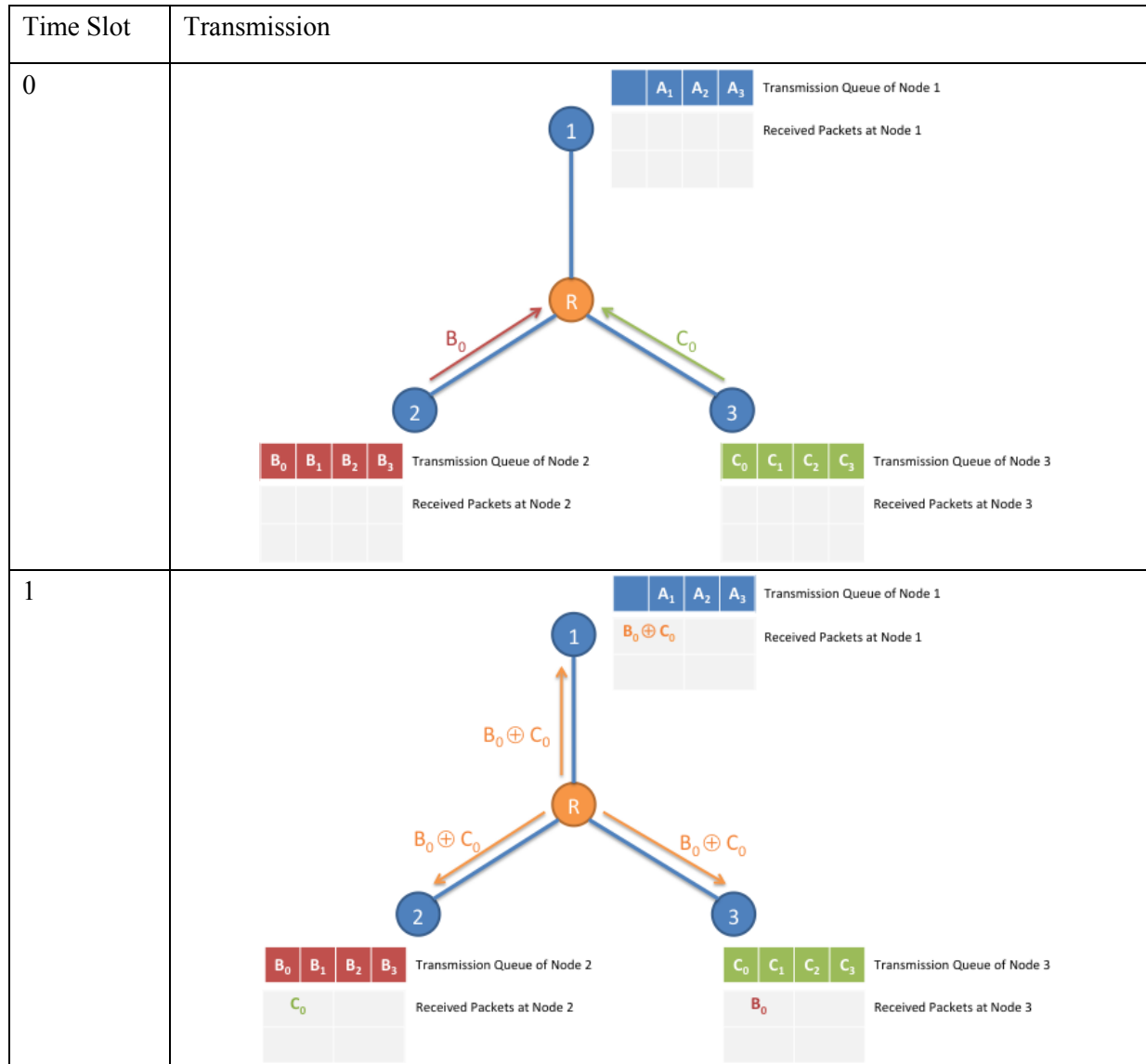


Figure 38: Throughput (information bits/sec) when using Nested Codes in Three-Sender Y-Channel-Relay problem and each sender needs to send the same packet to the other two users.

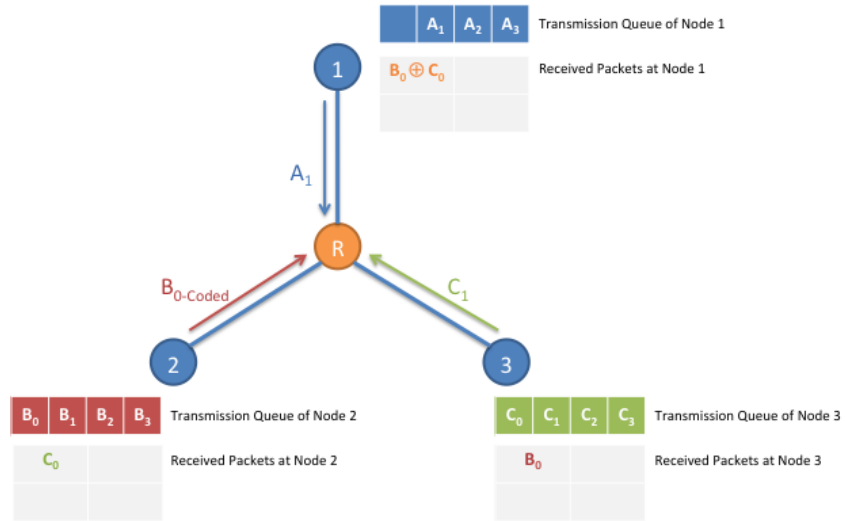
### 2.5.2. Combined Network Channel (CNC) Coding

B. Khoueir, H. Khoshnevis and M. R. Soleymani proposed in [6] a coding scheme and a communication protocol [28] for Physical Network Coding in Y-Channel-Relay topologies. In this approach, each sender transmits with the same power, and only two users are allowed to transmit new messages at full rate (rate one) while the third user at a rate that is equal to half that of the erased bits from a previous transmission. To harness the full potential of this scheme, the buffers of the each sender need to be full so that the transmission is continuous. Table 15 shows the transmission cycle of this scheme.

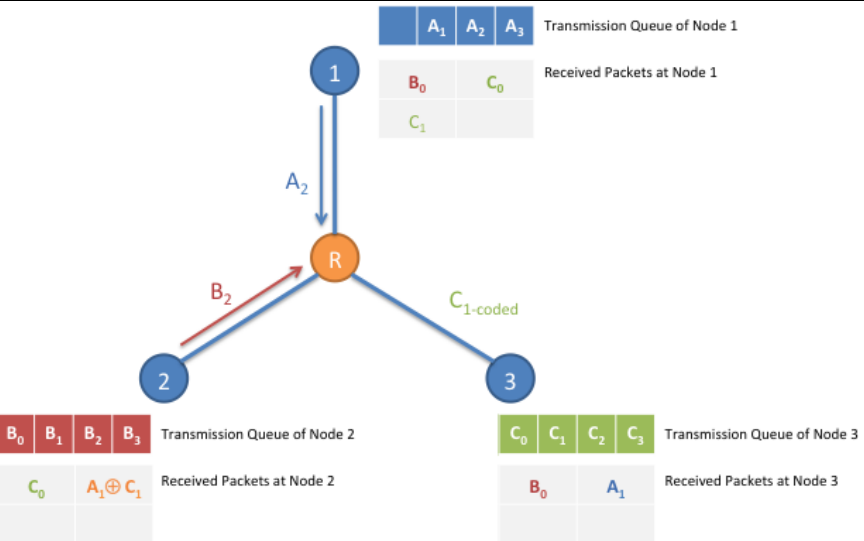
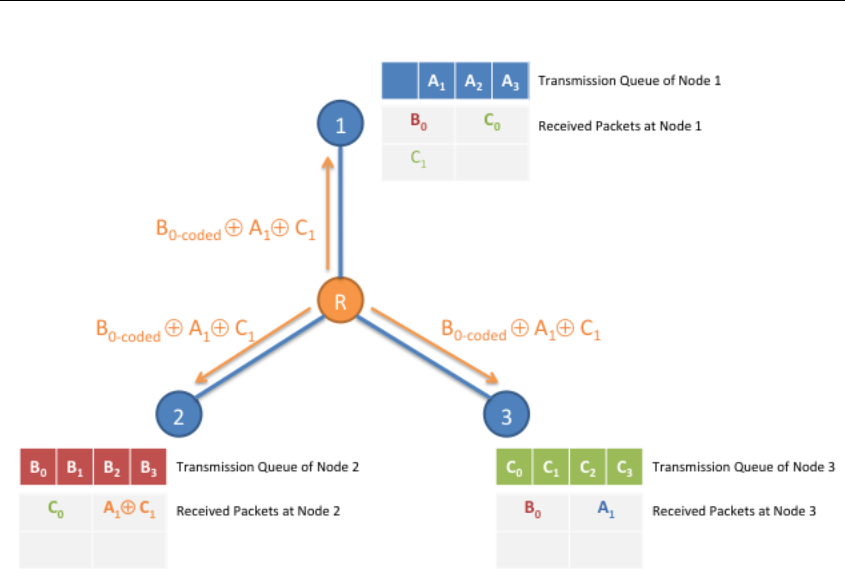
**Table 15: Transmission Flow Diagram in an Three-Sender Y-Channel-Relay network using Combined Network Channel (CNC) coding**

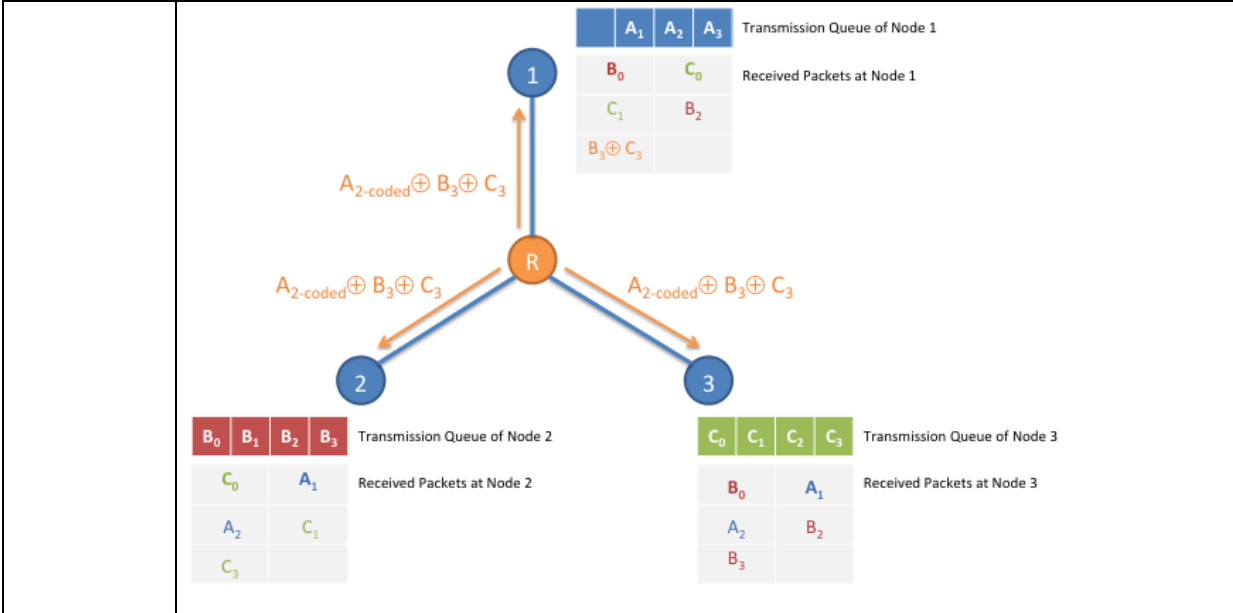
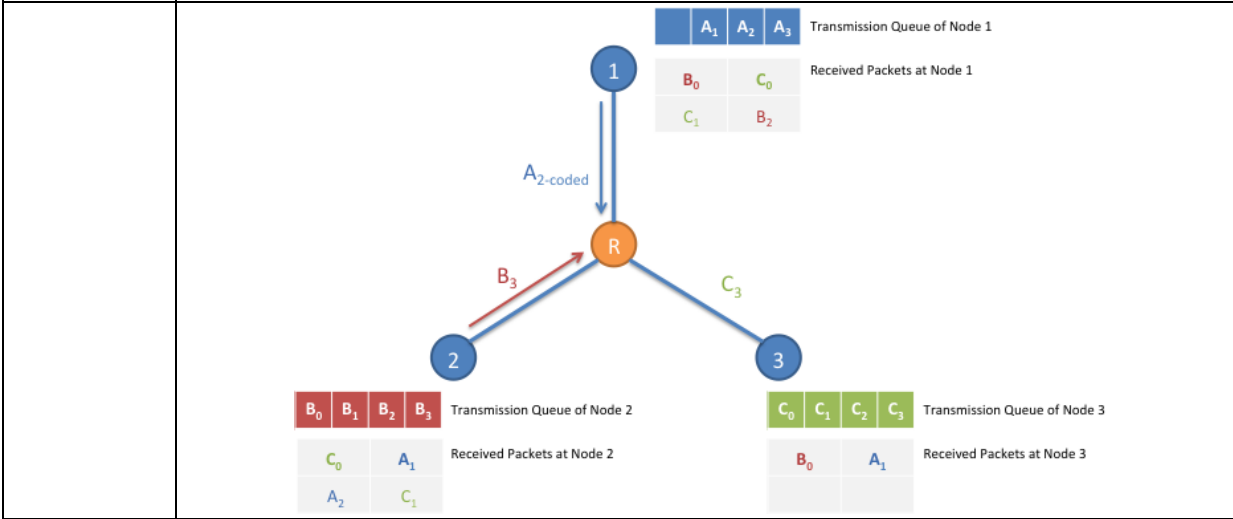
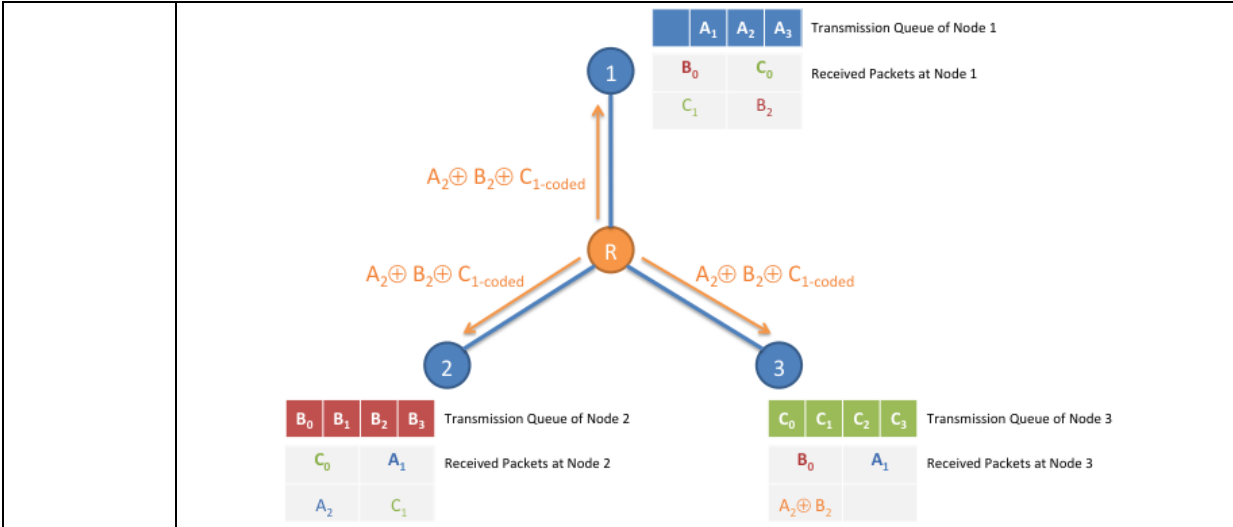


2



3





$A_i, B_i, C_i$  denote Node 1, Node 2, and Node 3 packets respectively that are coded at full rate (rate 1).  $A_{i-coded}, B_{i-coded}, C_{i-coded}$  contains erased bits from previous transmission and is transmitted at rate  $\frac{1}{2}$ . Upon reception, each user subtracts its own message, then attempts to decode. There are two possibilities for the codeword that remains after omitting one's own message; either an XORed signal that consists of two rate one codewords, and in this case, the messages are not yet decodable and the user waits for a rate  $\frac{1}{2}$  word to decode. Or, an XORed signal that consists of one rate  $\frac{1}{2}$  codeword and one full rate codeword. In this case, the decoder decodes the lower rate message first, then decodes the full rate message using successive decoding (SD).

The system above was simulated using a raptor code. A block length  $k = 65536$  of information bits was used. The block is pre-coded with LDPC code rate 0.98 before generating  $n$  encoded symbols to the optimized distribution in [32]. Figure 39 shows the simulation results. The system throughput maxes out at 1.94 bits/sec.

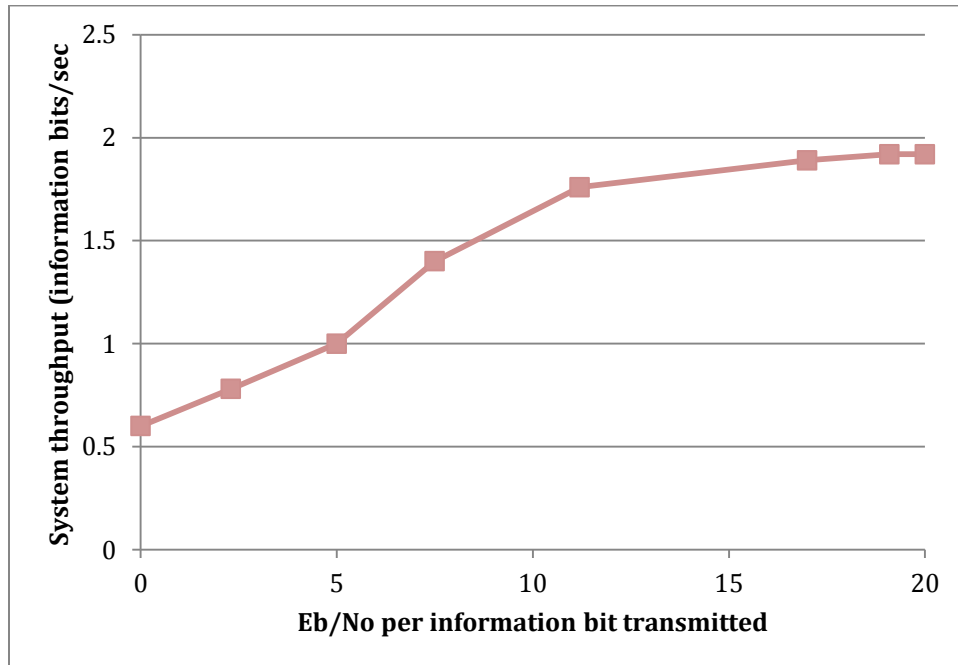


Figure 39: System throughput (info bits/sec) for full transmission scheme in an Three-Sender Network using CNC



The problem with this transmission protocol is that it is tailored to a network that has three terminals communicating constantly through a relay. I.e. the system is less efficient when each node transmit sporadically. To demonstrate this, let's take the simple example when each node needs to send one and only one packet to the other two nodes. In this case, four transmissions rather than two are required; Node 1 and Node 2 transmit at full rate in the first time slot. The relay broadcasts the received signal to the users in the second time slot. In the third time slot, Node 3 sends at full rate and Node 1 re-transmits the erased bits using rate half code. In the fourth time slot the relay broadcasts the received signal. In this case, we have transferred six packets in four transmissions (DoF=1.5 and not 2). In addition to this, the system is practically more complex since each node transmits at different rates in each transmission slot.

# Chapter 3: Proposed Coding Techniques for the Y-Channel-Relay Network

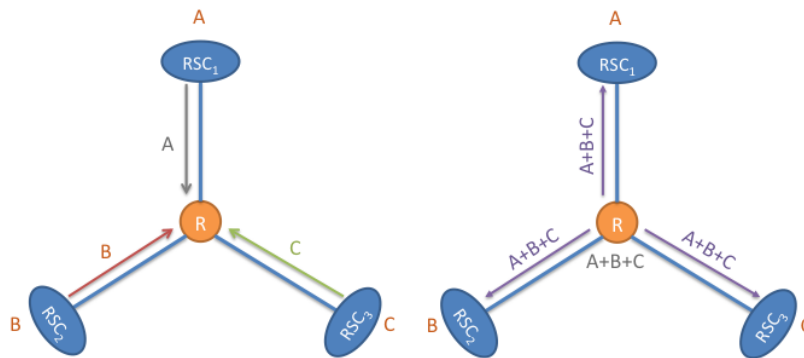
In this chapter, various collaborative coding schemes are proposed to solve the Y-Channel-Relay problem. First, a solution using long RSC Nested Codes is presented. Then, a Nested Codes solution based on Turbo Codes is proposed. After that, an intuitive solution based on Algebraic Linear Block Codes is explored in detail. At the end of the chapter, a comparison is made between all the different solutions discussed in this thesis.

## 3.1. Nested Recursive Convolutional Codes Solution

In this section, we will look at a solution to the Y-Channel-Relay problem using nested codes based on long RSC codes. This section describes, in detail the encoding and decoding sequence, and sets the fundamentals for nesting Turbo Codes, which will be discussed in section 3.2.

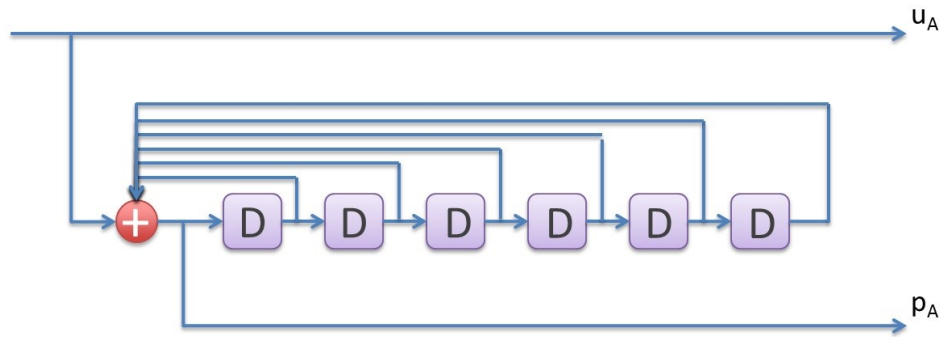
### 3.1.1. Encoding

In this solution, each node uses an RSC encoder with a unique and independent transfer function. The performance here is governed by the constraint length, rate, and minimum distance of the chosen RSC codes, as well as how different the state transition function used at each sender is. Figure 40 shows the transmission flow of this solution. Each terminal encodes its message bits using its RSC encoder, and transmits to the relay. All terminals transmit at relatively the same time, and it is assumed that the relay receives combined bits that are in sync. In other words, if the received message in relay is  $R = \{R_1, R_2, R_3 \dots\}$ , then  $R_i = A_i + B_i + C_i$ , where  $A_i$ ,  $B_i$ , and  $C_i$  are the  $i^{\text{th}}$  codeword of Node A, Node B, and Node C respectively. After that, the relay amplifies the received signal and broadcasts it to the receiving nodes. Each recipient subtracts its message from the received message, then, attempts to decode the remainder.

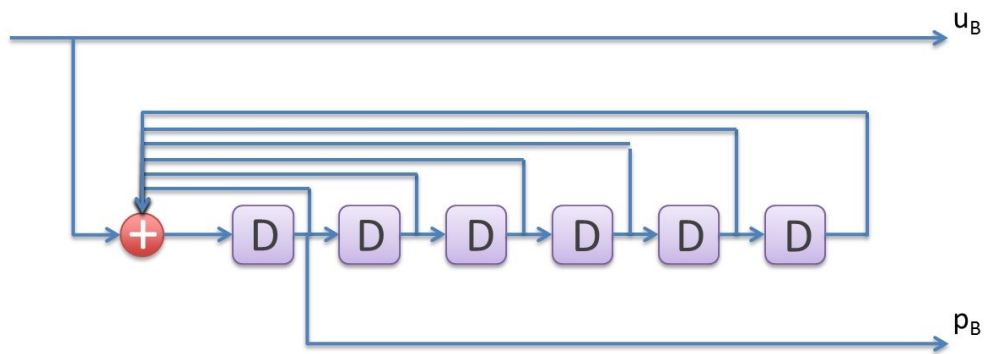


**Figure 40: Recursive Convolutional Codes based solution to the Y-Channel-Relay with ANC problem**

Let's consider an example using rate 1/3 RSC codes with constraint length equal to seven. Figure 41, Figure 42, and Figure 43 show the codes used at Node 1, Node 2, and Node 3 respectively. For the sake of simplicity, the codes used here have registers with unity length ( $k = 1$ )



**Figure 41: Recursive Convolutional Encoder of Node 1**



**Figure 42: Recursive Convolutional Encoder of Node 2**

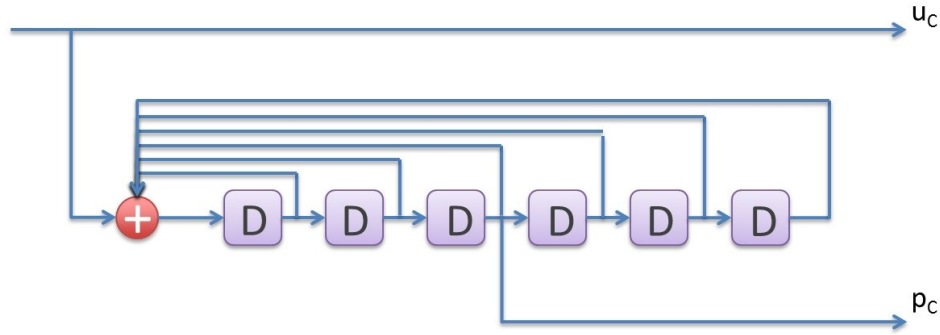


Figure 43: Recursive Convolutional Encoder of Node 3

The transform domain generator matrices for the encoders above are as follows:

$$\mathbf{G}_{RSC_1}(D) = \left[ 1 \quad \frac{1}{1 + D + D^2 + D^3 + D^4 + D^5 + D^6} \right] \quad (3.1)$$

$$\mathbf{G}_{RSC_2}(D) = \left[ 1 \quad \frac{D}{1 + D + D^2 + D^3 + D^4 + D^5 + D^6} \right] \quad (3.2)$$

$$\mathbf{G}_{RSC_3}(D) = \left[ 1 \quad \frac{D^3}{1 + D + D^2 + D^3 + D^4 + D^5 + D^6} \right] \quad (3.3)$$

### 3.1.2. Decoding

The decoder used to separate the combined signal is an evolution of the iterative modified BCJR-decoder of turbo codes. The soft decoder of the RSC codes solution relies on the simple fact that if the received bit is erased, then that means the relative bits from the first and second senders are not equal (one is equal to 1, and the other is equal to 0). On the other hand, if the received bit is not erased, then the relative bits from the first and second senders are equal (both are 1, or both are 0). Assuming we are decoding at node C and we have a noiseless channel with equal bit energy levels, where  $\sqrt{E_b} = \gamma$ , we will have one of the three following scenarios:

- $A_i + B_i = 0 \rightarrow$  erasure
  - Then,  $A_i = -\gamma$ ,  $B_i = +\gamma$  or  $A_i = +\gamma$ ,  $B_i = -\gamma$
- $A_i + B_i = +2\gamma \rightarrow$  high-bits
  - Then,  $A_i = +\gamma$ ,  $B_i = +\gamma$

- $A_i + B_i = -2\gamma \rightarrow$  low-bits
  - Then,  $A_i = -\gamma, B_i = -\gamma$

When the channel is noisy, the info bits can be determined using the decision regions described in Figure 45.

As with turbo decoders, the RSC based decoder for Y-Channel-Relay not only solves for erasures, but also corrects bit errors caused by white noise. The derived decoder is shown in the figure below.

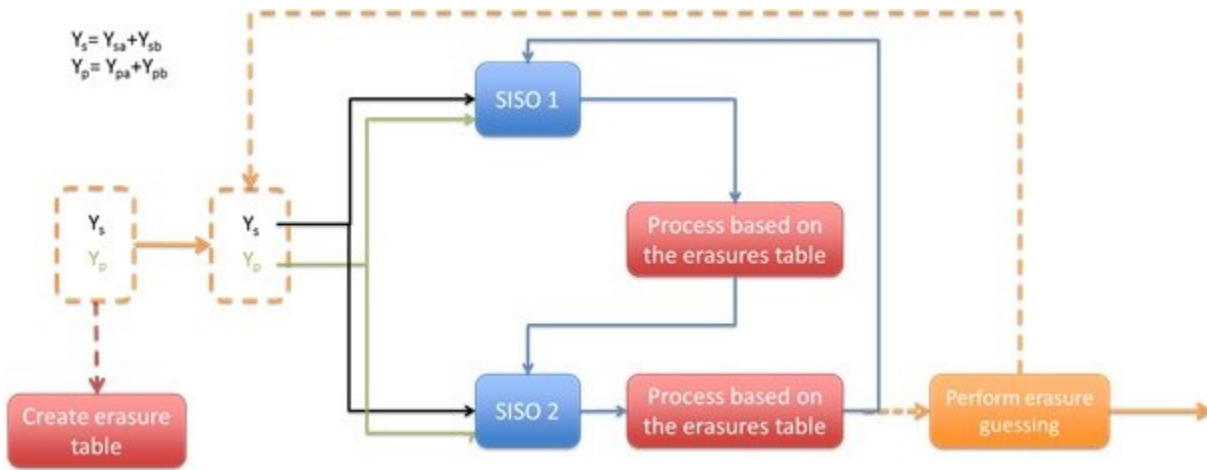


Figure 44: Iterative Decoder for the Recursive Convolutional Codes based solution

$Y_s$  and  $Y_p$  represent the combined noisy systematic and parity signal sequences found at the receiver after deducting its message. At node C, this would be:

$$Y_s = A_s + B_s \quad (3.4)$$

$$Y_p = A_p + B_p \quad (3.5)$$

Similar to turbo decoders, the decoder represented here relies on iterative use of Log-APP or max-Log-APP decoding algorithms. The following subsections describe the decoding process in details.

### 3.1.2.1. *Creating the erasure table*

After subtracting the recipient's message from the received combined signal, the decoder starts creating an erasure table. The erasure table is a binary table of a length equal to the input sequence length. For each received symbol, the erasure table holds a value of either 1 when the decoder initially thinks that the symbol was erased, or 0 for when it thinks the symbol was not erased, based on the combined received signal after deducting the receiver's own symbol.

Initially, the erasure table is filled based only on the received analog signal. Assuming that all sources transmit at the same power level, the decision regions will be as follows:

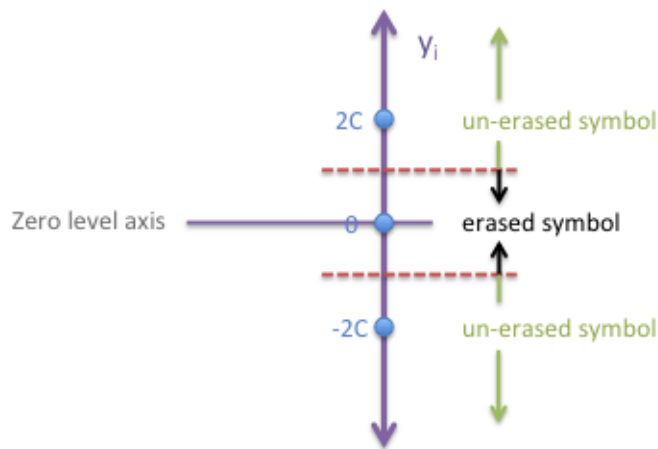


Figure 45: The Decision Regions for estimating erasures

The erasure table can be updated after each decoding iteration for better erasure estimation based on the latest soft outputs of the decoder. In addition, one can use a soft erasure table with soft values representing the likelihood of a bit being erased, rather than having hard binary erasure table.

### 3.1.2.2. *Soft-Input Soft-Output Decoder*

There are two SISO decoders used in the overall iterative design. Each SISO decoder is a BCJR soft output decoder tailored for each sender. At node C, for example, SISO-1 decoder is based on Node A's encoder while SISO-2 decoder is based on Node B's encoder.

In every iteration, the SISO decoder accepts the input signals  $(\mathbf{Y}_s, \mathbf{Y}_p)$  as well as the a posteriori soft values taken from previous iterations. The a posteriori  $L$  values can be evaluated by the following sum:

$$L(u_i) = L_c y_i^s + L^{(a)}(u_i) + L^{(e)}(u_i) \quad (3.6)$$

where;

$$L_c y_i^s = \frac{4\sqrt{\mathcal{E}_c} y_i^s}{N_0} \quad (3.7)$$

$$L^{(a)}(u_i) = \ln \frac{P(u_i = 1)}{P(u_i = 0)} \quad (3.8)$$

$$L^{(e)}(u_i) \approx \max_{(\sigma_{i-1}, \sigma_i) \in \mathcal{S}_1} \left\{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \tilde{\beta}_i(\sigma_i) \right\} - \max_{(\sigma_{i-1}, \sigma_i) \in \mathcal{S}_1} \left\{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \tilde{\beta}_i(\sigma_i) \right\} \quad (3.9)$$

$L_c$  is defined as  $L_c = \frac{4}{N_0} \sqrt{\mathcal{E}_c}$ .

There are three terms of interest in the previous equations:

- $L_c y_i^s$  indicates the effects of the channel output (decoder input) corresponding to the systematic symbols.
- $L^{(a)}(u_i)$  is the a priori probabilities of the sequence.

- $L^{(e)}(u_i)$  is the extrinsic information. It is the part of the a posteriori value that doesn't depend on the channel output of the calculated a priori values.

The latter equations give a sub optimal method for MAP calculations. It is called the Max-Log-APP algorithm.

### 3.1.2.3. *Iterative Decoding*

As discussed before, the overall decoder is comprised of two SISO decoders. Once the soft output sequence of SISO 1 is evaluated, it is passed to a process that converts it to A priori values that are used as inputs to the second SISO. This process relies on the fact that if we have an erasure on a certain bit, its soft a priori input for SISO 2 will be of opposite sign from the output of SISO 1. This is called the “flipping” operation. On the contrary, if we don't have an erasure for a bit, its A priori value in SISO 2 will be equal to its soft output value of SISO 1.

SISO 2 will calculate soft outputs that are going to be used as inputs for the first SISO. This iterative approach can be repeated several times until we reach high confidence for decoding. It is important to note here that in the first iteration, the decoder assumes that the a priori values for the sequence are zeros (since we have equiprobable inputs). The soft output passed between the decoders is the new extrinsic information that we get from subtracting the channel values from the calculated posteriori values. Calculating newer extrinsic information by relying on previously calculated extrinsic information and the difference/orthogonality in the state functions of the two SISO decoders is what make this a powerful iterative decoding solution.

After a few decoding iterations, the soft output values can be used to revise the erasure table. Since the erasure table was initially built using raw input, it is probable that we might have symbol errors building it. Using the soft outputs after every iteration to re-build the erasure table enhances the performance of the decoder.



### 3.1.3. Simulation

The example discussed in the previous section was simulated using a decode and forward approach. Three equi-probable binary sources generate an endless stream of bits, and the recursive convolutional encoders of constraint length  $K=7$  shown in Section 3.1.1 were used at Nodes 1, 2 and 3.

After encoding, the bits are passed into a BPSK modulator with levels  $\{-\sqrt{E_b}, +\sqrt{E_b}\}$ . The initial energy level is chosen to give an SINR of 0 dB. Once a predetermined confidence value for  $P_b$  is earned from the simulating at that SINR level,  $E_b$  is changed to obtain a higher SINR, and the process is repeated to get the  $P_b$  curve.

For each SINR level, the following is done to obtain the  $P_b$ :

- Produce a continuous stream of bits from the three binary sources.
- Encode the bits and modulate them.
- Combine the relative output from each node, and add AWGN to it based on the chosen  $E_b$  and  $N_0$  to get the desired SINR. This simulates mixing the signals in the air at the relay.
- The received signal is then decoded in the received node.

Here is the curve for the  $P_b$  when simulating the collaborative recursive convolutional codes based solution at node 3 when compared to the uncoded solution:

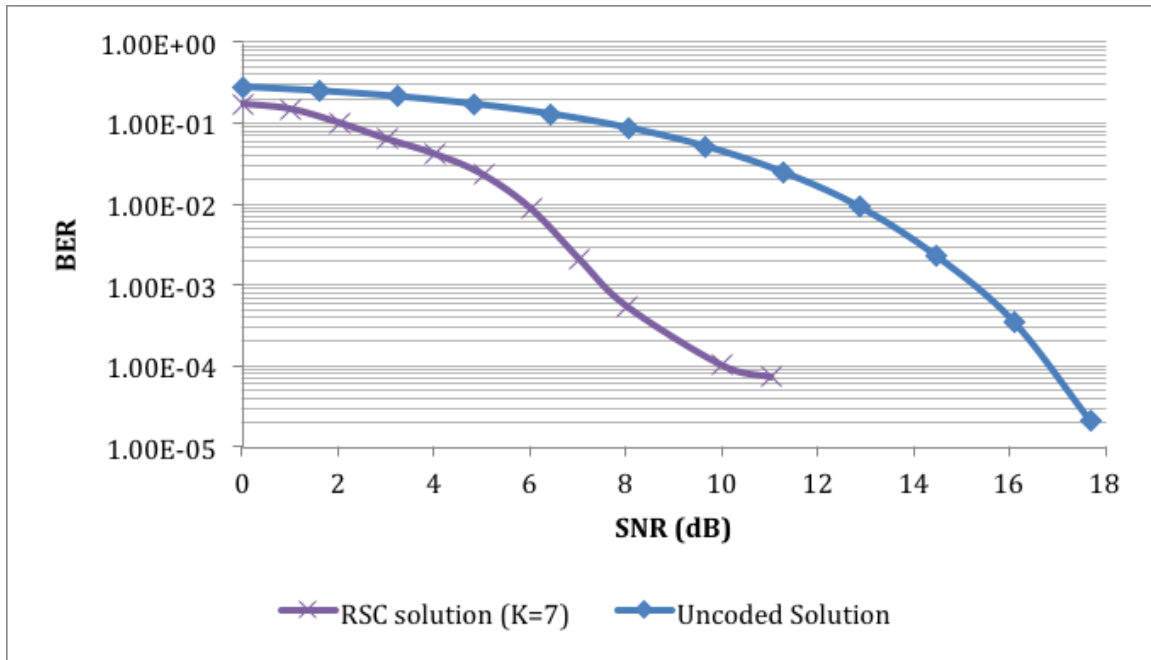


Figure 46: Simulated BER performance of the RSC based solution example with K=7

The curve shows that using there is a maximum of approximately 8 dB better performance than the uncoded solution. It also shows that the error floor is below  $10^{-4}$ . In the following subsection, the effects of the constraint length on performance and error floor are studied

### 3.1.3.1. *Effect of the constraint length*

In order to study the effect of the used constraint length on the performance of the recursive convolutional codes solution, another solution with constraint length of K=5 was simulated, and the results were compared with the previous example. Figure 47, Figure 48, and Figure 49 show the encoders used at Node1, Node 2, and Node 3 respectively.

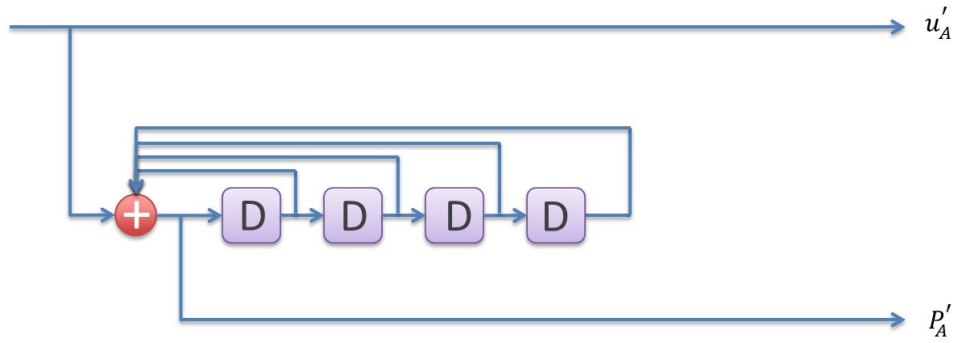


Figure 47: RSC encoder of Node 1 with K=5

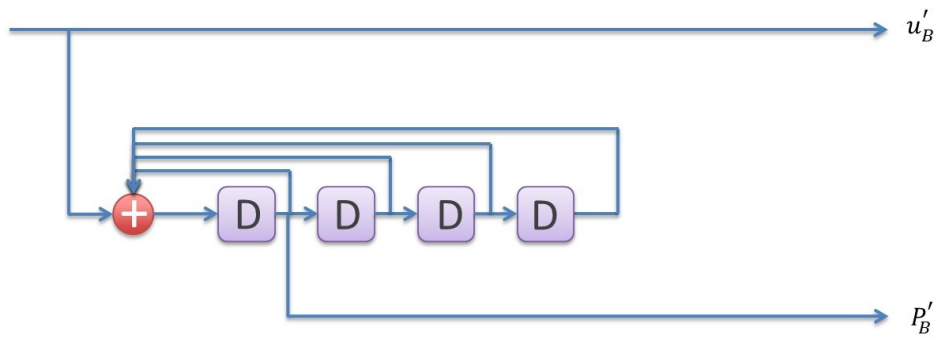


Figure 48: RSC encoder of Node 2 with K=5

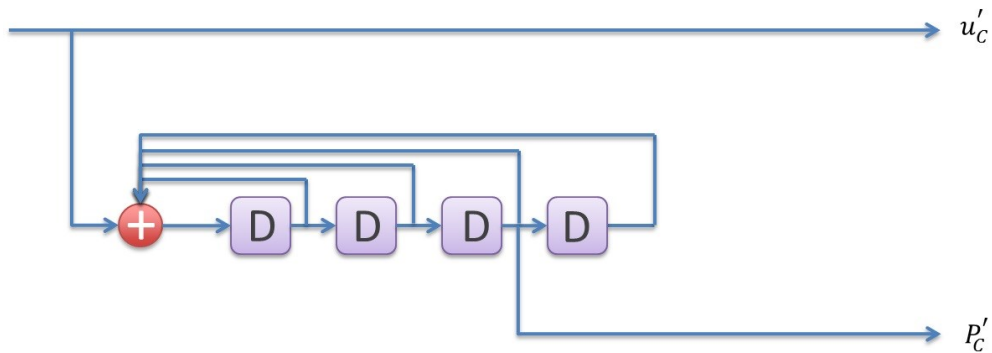


Figure 49: RSC encoder of Node 3 with K=5

The following plot shows the BER results when using the K=5 encoders described above compared to the K=7 encoder combination shown in the previous sub-section:

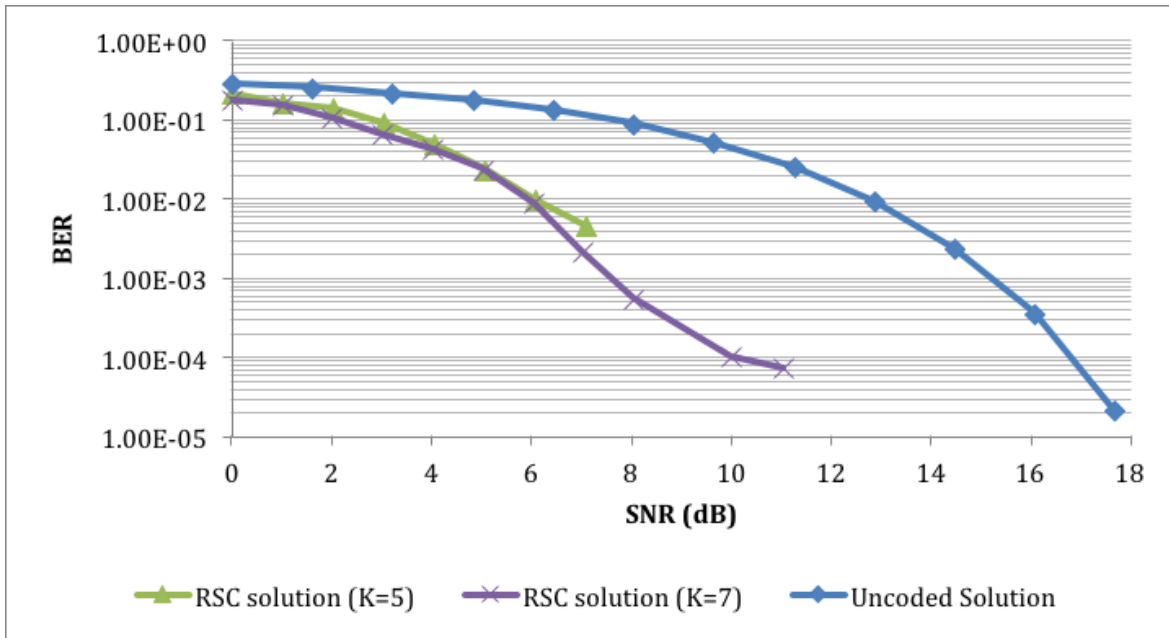


Figure 50: Effect of used RSC constraint length on BER performance

From the performance plot, it can be noted the performance is slightly better when using a combination of codes with larger constraint length. This makes sense due to the fact that larger constraint lengths give longer codes. In addition, the error floor drops significantly when using higher constraint lengths. In theory, increasing the constraint length can virtually eliminate the error floor problem. That being said, increasing the constraint length increases the decoding complexity exponentially.

It is good to mention that not only the constraint length affects the performance, but also the combination of the used encoder polynomials. Variations of encoders with  $K=5$  and  $K=7$  were picked by trial and error and used to obtain good results. More research can go into choosing the optimum combinations for each constraint length to maximize the distance between each node's encoder.

In the following section, the idea presented for recursive convolutional codes is expanded to accommodate the use of turbo codes.

## **3.2. Nested Turbo Codes Based Solution**

The collaborative RSC Nested Codes based solution for the Y-Channel-Relay with ANC problem can be extended to use turbo codes. Turbo codes have performance that is near-Shannon bound. Using Turbo-codes allows for more code diversity due to the use of interleaving. This leads to lower error floors and better BER in the low SNR conditions.

### **3.2.1. Encoding**

In this solution, each terminal has its own unique turbo encoder that consists of two different recursive systematic convolutional encoders and a unique interleaver. The encoder can use a puncturing process to get a higher code rate. Having different interleavers and different RSC encoders in each turbo encoder creates more linear relationships that may be harnessed for better decoding at the receiver.

To describe this solution, an example is considered using a unique Turbo Code of constraint length of four ( $K=4$ ). Figure 51, Figure 52, and Figure 53 show the turbo encoder used at Nodes 1, 2 and 3 respectively.

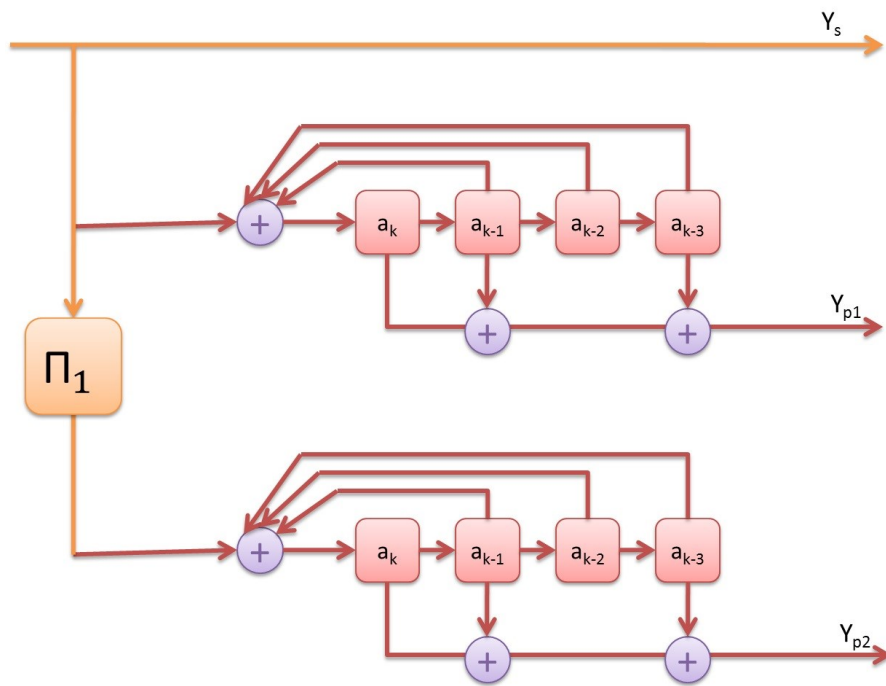


Figure 51: Turbo encoder of node 1

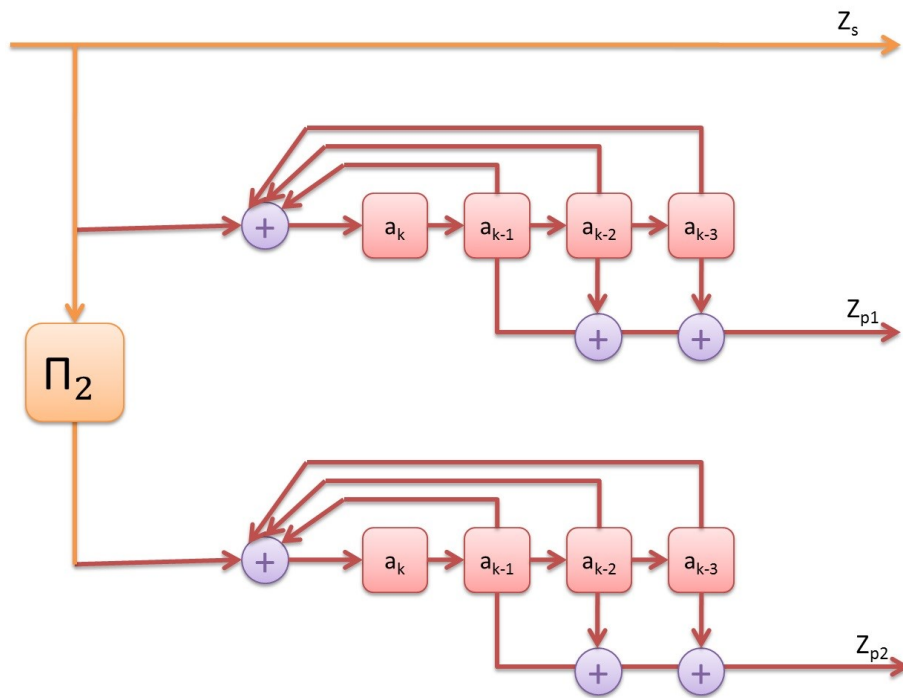


Figure 52: Turbo encoder of node 2

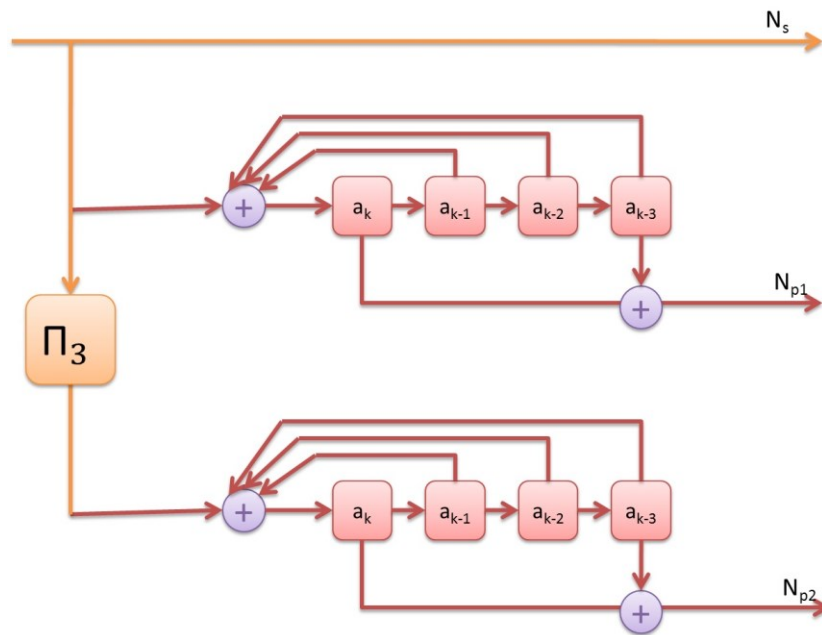


Figure 53: Turbo encoder of node 3

### 3.2.2. Decoding

The decoder for the turbo-codes solution can be realized in two ways. On a high level, the decoder can be built with two turbo decoders that iteratively pass information to each other using modified BCJR algorithms until a certain confidence in the decoding decision is reached. Figure 54 shows a high level block diagram of the decoder.

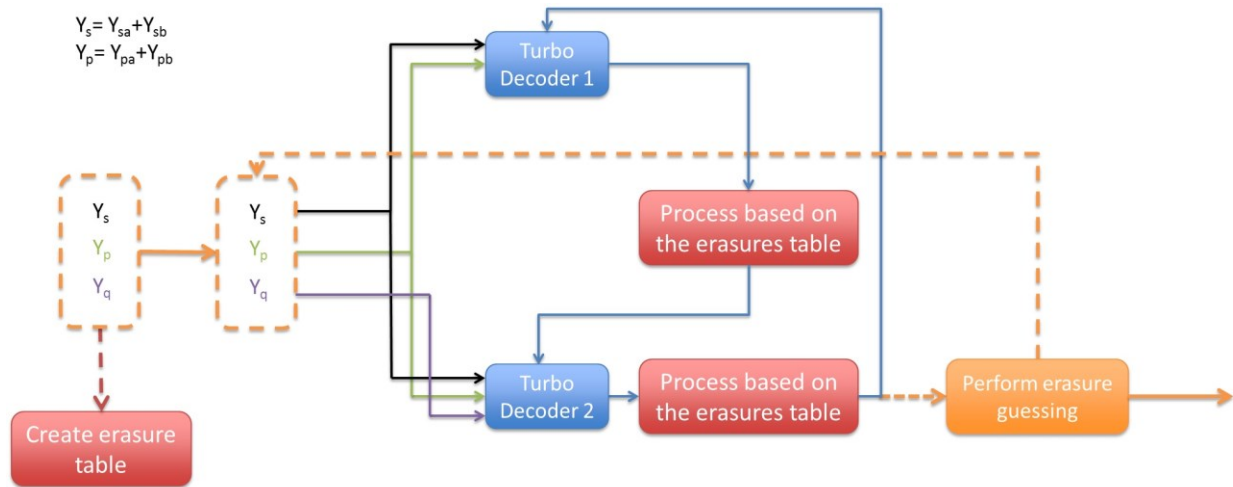


Figure 54: High-level Turbo based solution decoder

We can dissect the decoder further. It can be implemented using four soft-input soft-output decoders. In this case, each SISO decoder passes soft-output values that result from modified BCJR algorithm to the following SISO decoder. The process is repeated iteratively until a decision with high confidence is made. It can be noticed here that the turbo-based solution is a natural extension of the convolutional codes based solution. Figure 55 shows the low-level block diagram of the decoder.

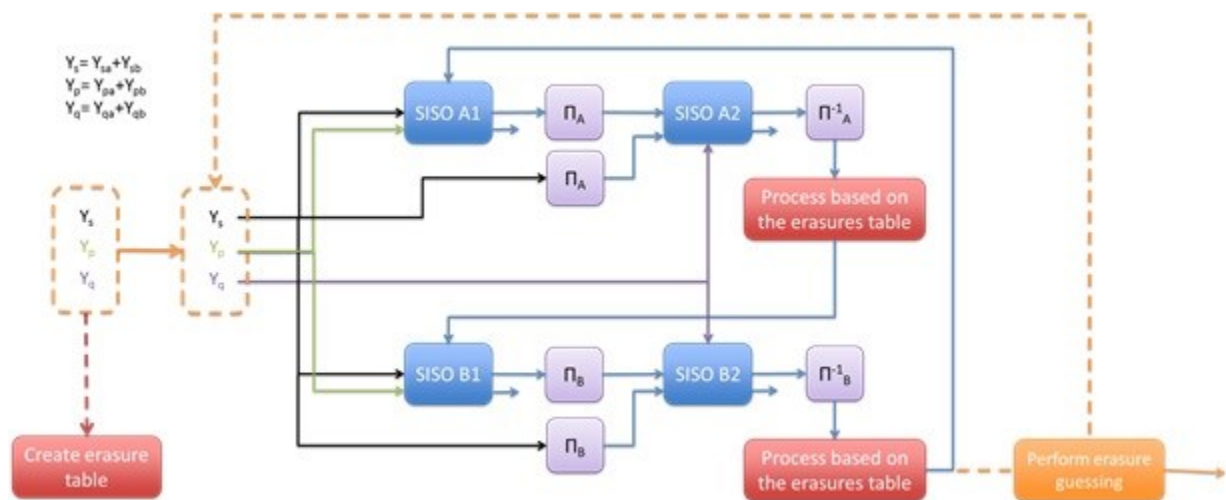


Figure 55: Low level turbo-codes based solution decoder

$Y_s$ ,  $Y_p$  and  $Y_q$  represent the combined noisy systematic and parity signal sequences found at the receiver after deducting its message. At node C, this would be:



$$Y_s = A_s + B_s \quad (3.10)$$

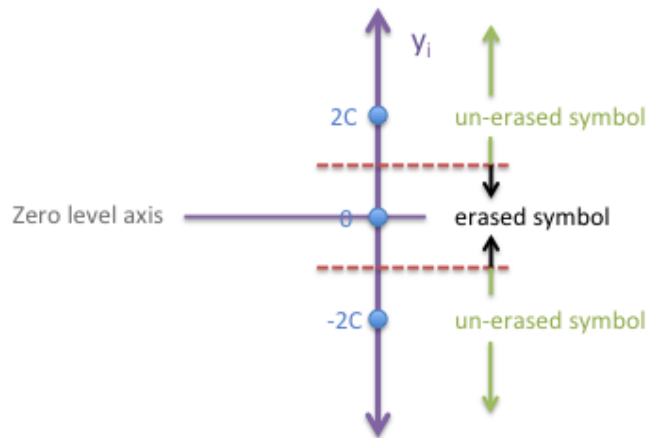
$$Y_p = A_p + B_p \quad (3.11)$$

$$Y_q = A_q + B_q \quad (3.12)$$

Similar to regular turbo decoders, this decoder relies on iterative use of Log-APP or max-Log-APP decoding algorithms. The following subsections describe the decoding process in details.

### 3.2.2.1. *Creating the erasure table*

The process for creating the erasure table for the turbo-based nested codes solution is identical to the convolutional codes based solution. After subtracting its message from the received signal, the recipient starts creating an erasure table. The erasure table is a binary table of a length equal to the input sequence length. For each received symbol, the erasure table holds a value of either 1 when the decoder initially thinks that the symbol was erased, or 0 for when it thinks the symbol was not erased, based on the combined received signal after deducting the receiver's own symbol. Initially, the erasure table is filled based only on the received analog signal. Assuming that all sources transmit at the same power level, the decision regions are as shown in Figure 56.



**Figure 56: Decision Region used to estimate erasures in Turbo based Nested Codes solution**

The erasure table can be updated after each decoding iteration for better erasure estimation based on the latest soft outputs of the decoder.

### 3.2.2.2. Iterative Decoding

There are four SISO decoders used in the overall iterative design. Each SISO decoder is a BCJR soft output decoder tailored for each encoder in a sender. At node C, for example, SISO-A1 decoder is based on node A's first RSC encoder, SISO-A2 decoder is based on node A's second RSC encoder, while SISO-B1 decoder is based on node B's first RSC encoder and SISO-B2 decoder is based on node B's second RSC encoder.

In each iteration, the SISO decoder accepts the input signals  $(\mathbf{Y}_s, \mathbf{Y}_p)$  or  $(\mathbf{Y}_s, \mathbf{Y}_q)$  as well as the a posteriori soft values taken from previous iterations. The a posteriori  $L$  values can be evaluated by the following sum:

$$L(u_i) = L_c y_i^s + L^{(a)}(u_i) + L^{(e)}(u_i) \quad (3.13)$$

where;

$$L_c y_i^s = \frac{4\sqrt{\mathcal{E}_c} y_i^s}{N_0} \quad (3.14)$$

$$L^{(a)}(u_i) = \ln \frac{P(u_i = 1)}{P(u_i = 0)} \quad (3.15)$$

$$L^{(e)}(u_i) \approx \max_{(\sigma_{i-1}, \sigma_i) \in \mathcal{S}_1} \left\{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \tilde{\beta}_i(\sigma_i) \right\} - \max_{(\sigma_{i-1}, \sigma_i) \in \mathcal{S}_1} \left\{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \tilde{\beta}_i(\sigma_i) \right\} \quad (3.16)$$

$L_c$  is defined as  $L_c = \frac{4}{N_0} \sqrt{\mathcal{E}_c}$ .

There are three terms of interest in the previous equations;

- $L_c y_i^s$  indicates the effects of the channel output (decoder input) corresponding to the systematic symbols.
- $L^{(a)}(u_i)$  is the a priori probabilities of the sequence.
- $L^{(e)}(u_i)$  is the extrinsic information. It is the part of the a posteriori value that doesn't depend on the channel output of the calculated a priori values.

The latter equations give a sub optimal method for MAP calculations. It is called the Max-Log-APP algorithm. Once the soft values are calculated based on the first turbo decoder (SISO 1), the soft output values are treated via a flipping operation based on whether a bit is erased or not. If the bit is erased, the sign of the soft output value is flipped, if it is not, the soft output value is kept as is. After that, the a posteriori values are passed to the second turbo decoder. SISO 2 calculates soft outputs that are going to be used as inputs for the first SISO. This iterative process is repeated several times until we reach high confidence for decoding.

It is important to note here that in the first iteration, the decoder assumes that the a priori values for the sequence are zeros (since we have equiprobable inputs). Calculating newer extrinsic information by relying on previously calculated values in previous iterations and the difference/orthogonality in the state functions of the two SISO decoders is what make this a powerful iterative decoding solution.

After a few decoding iterations, the soft output values can be used to revise the erasure table. Since the erasure table was initially built using raw input, it is probable that we might have symbol errors building it. The soft outputs from each iteration can be used to re-build the erasure table and, thereby, enhance the performance of the decoder.

### 3.2.3. Simulation Results

The turbo based collaborative scheme was simulated using a decode and forward approach. Three equi-probable binary sources generate an endless stream of bits. The encoders for Node 1, Node 2, and Node 3 are shown in Figure 51, Figure 52, and Figure 53 respectively. After

encoding, the bits are passed into a BPSK modulator with levels  $\{-\sqrt{E_b}, +\sqrt{E_b}\}$ . The initial energy level is chosen to give an SINR of 0 dB. Once a predetermined confidence value for  $P_b$  is earned from the simulating at that SINR level,  $E_b$  is changed to obtain a higher SINR, and the process is repeated to get the  $P_b$  curve.

For each SINR level, the following is done to obtain the  $P_b$ :

- Produce a continuous stream of bits from the three binary sources.
- Encode the bits and modulate them.
- Combine the relative output from each node, and add AWGN to it based on the chosen  $E_b$  and  $N_0$  to get the desired SINR. This simulates mixing the signals in the air at the relay.
- The received signal is then decoded in the received node. A decoding length of 900 bits is used which means that a batch of 900 bits is decoded at a time. The iterative decoding algorithm is run 20 times before making a decoding decision.

Figure 57 shows the curve for the  $P_b$  when simulating the collaborative turbo based solution at node 3 when compared to the uncoded solution:

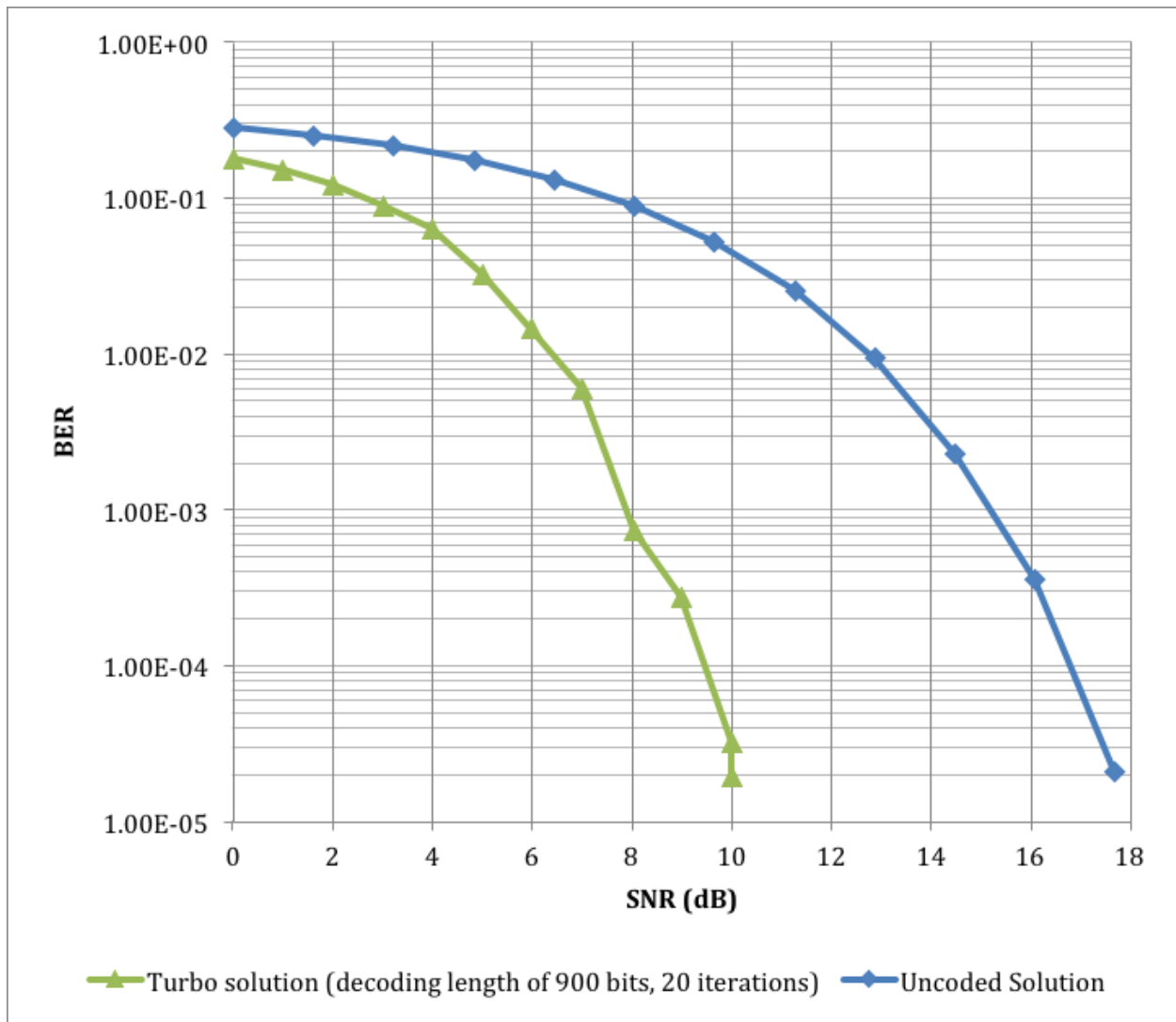


Figure 57: Simulated BER performance of Turbo based solution

The error floor obtained are lower than  $10^{-5}$ , which is significantly lower than that of the RSC based solution explored in the previous section. That is even more impressive considering the fact that the turbo solution uses codes of constraint length of 4, which is lower than that used in the simulation of the RSC code. Figure 58 compares the performance of the RSC code with constraint length 7, and the turbo code with constraint length 4.

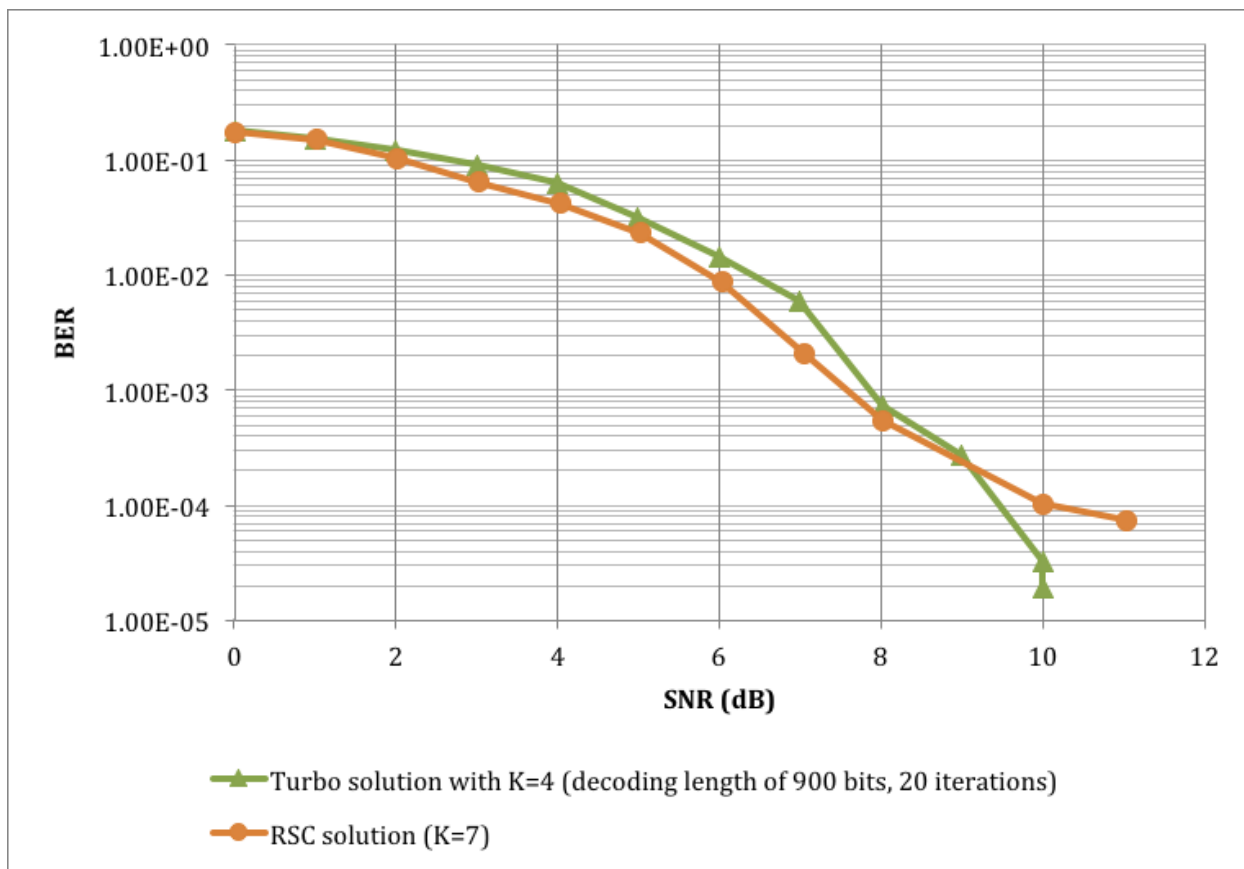


Figure 58: Comparison in BER performance between Turbo and RSC based solutions

From the plot above, it is observed that the performance of the RSC solution is ever slightly better at low SNR due to the fact that it is using a significantly higher constraint length. However, the turbo solution with constraint length of only 4 is far superior at high SNR with its significantly lower error floor. It is worth mentioning here that the exact error floor of the turbo solution couldn't be shown in the plots due to a limitation of the simulation itself.

In conclusion, nesting turbo codes to solve the Y-Channel-Relay with ANC is superior than using RSC codes. A hardware decoder would be able to handle lengthier turbo encoders with larger constraint lengths, and puncturing to obtain a practical solution that virtually eliminates the error floor problem, and at the same time provides very good performance at low SNR.

### 3.3. Collaborative Algebraic Linear Block Codes Solution

While the proposed solutions based on nested codes in sections 3.1 and 3.2 are practical, they use low rate codes. Thus, making them inferior to solutions based on raptor codes [6] and lattice codes [4].

The Y-Channel-Relay problem can be looked at algebraically using linear codes. Linear block codes such as Hamming Codes, BCH codes, and LDPC codes encode a block of information bits by adding parity bits that can be looked at as simple linear algebraic relationships. On the other hand, erasures introduced from network coding can also be looked at as linear correlation between the  $i^{th}$  bit of the first and second senders. If we assume we have two binary sources  $X_1^i \in \{-1, +1\}$  and  $X_2^i \in \{-1, +1\}$ , and that  $Y^i = X_1^i + X_2^i$ . The outcome of  $Y^i$  will be one of the following:

- $Y^i = -2 \quad \rightarrow X_1^i = X_2^i = -1$
- $Y^i = 0 \quad \rightarrow X_1^i = -X_2^i$
- $Y^i = +2 \quad \rightarrow X_1^i = X_2^i = +1$

In this section, we will introduce a novel algebraic solution to the Y-Channel-Relay problem that harness the power of linear correlation by using linearly independent block codes at each sender. To explain this idea, let's consider a simple example. Assume that:

- Each sender in the Y-Channel-Relay problem system shown in Figure 14 uses a unique linearly independent (7,4) Hamming code (Figure 59).
- Node 1's message bits are denoted with  $a_1, a_2, \dots, a_7$ , where  $a_1, a_2, a_3, a_4$  are Node 1's information bits, and  $a_5, a_6, a_7$  are Node 1's parity bits. Similarly, Node 2's bits are denoted with  $b_i$ , and Node 3's bits are denoted with  $c_i$ .
- Node 1, Node 2, and Node 3 transmit at the same power, and are equi-probable binary sources. They transmit at the following levels respectively:
  - $X_1^i \in \{-\rho, +\rho\}$

- $X_2^i \in \{-\rho, +\rho\}$
- $X_3^i \in \{-\rho, +\rho\}$
- The signal received at the relay when fading is not taken into consideration and assuming we have an AWGN channel would be:
  - $r^i = X_1^i + X_2^i + X_3^i + n$

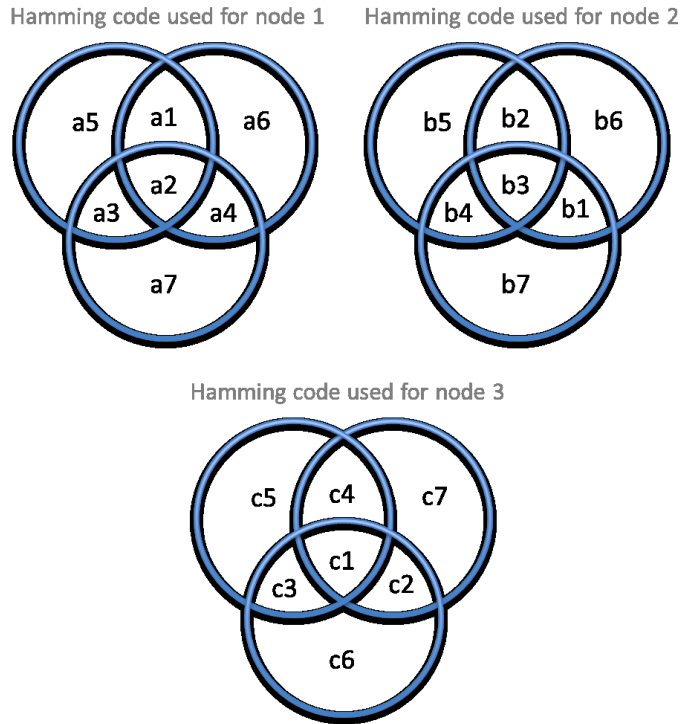


Figure 59: Bubble notation for the Collaborative Hamming Code encoder used in each terminal

Figure 59 shows the bubble notation [24] for the encoders used at each terminal. They translate to the following linear relations:

$$a_5 = a_1 + a_2 + a_3 \tag{3.17}$$

$$a_6 = a_1 + a_2 + a_4 \tag{3.18}$$

$$a_7 = a_2 + a_3 + a_4 \tag{3.19}$$

$$b_5 = b_2 + b_3 + b_4 \tag{3.20}$$

$$b_6 = b_1 + b_2 + b_3 \tag{3.21}$$

$$b_7 = b_1 + b_3 + b_4 \tag{3.22}$$



$$c_5 = c_1 + c_3 + c_4 \quad (3.23)$$

$$c_6 = c_1 + c_2 + c_3 \quad (3.24)$$

$$c_7 = c_1 + c_2 + c_4 \quad (3.25)$$

The table below shows the mapping between the 16 possible inputs at each sender, and their corresponding codeword.

**Table 16: Encoding table for collaborative linear block code solution example**

Info bits (k=4)	Node 1 message	Node 2 message	Node 3 message
$m_1 m_2 m_3 m_4$	$a_1 a_2 a_3 a_4 \ a_5 a_6 a_7$	$b_1 b_2 b_3 b_4 \ b_5 b_6 b_7$	$c_1 c_2 c_3 c_4 \ c_5 c_6 c_7$
0000	0000 000	0000 000	0000 000
0001	0001 011	0001 101	0001 101
0010	0010 101	0010 111	0010 110
0011	0011 110	0011 010	0011 011
0100	0100 111	0100 110	0100 011
0101	0101 100	0101 011	0101 110
0110	0110 010	0110 001	0110 101
0111	0111 001	0111 100	0111 000
1000	1000 110	1000 011	1000 111
1001	1001 101	1001 110	1001 010
1010	1010 011	1010 100	1010 001
1011	1011 000	1011 001	1011 100
1100	1100 001	1100 101	1100 100
1101	1101 010	1101 000	1101 001
1110	1110 100	1110 010	1110 010
1111	1111 111	1111 111	1111 111

The following can be concluded from the latter codebook; the used block code, without taking into consideration noise in the channel, can separate the combined signal successfully given that the number of erasures in the received codeword is less or equal to 6 erasures. That is because the number of linearly independent parity bits, and hence the number of orthogonal linear relationships, is equal to six at each receiving node, the code can solve for up to six variables (six erasures).

### 3.3.1. The Decoding Process

Having linearly independent BCH/linear block codes when using ANC in Y-Channel-Relay problems lead to more redundancy and better error correction capability since the code won't

only identify erasure bit positions and guess the original bit values, but also detect and correct bit errors.

To give a better understanding of how this coding scheme helps in signal separation, a concrete example is given without taking noise into consideration. Assume that each node sends the following:

**Table 17: Node 1 message**

	Info bits				Parity bits		
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$
<b>Node 1 msg (binary)</b>	0	0	1	1	1	1	0
<b>Node 1 msg (sent signal)</b>	$-\rho$	$-\rho$	$+\rho$	$+\rho$	$+\rho$	$+\rho$	$-\rho$

**Table 18: Node 2 message**

	Info bits				Parity bits		
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$
<b>Node 2 msg (binary)</b>	1	1	0	0	1	0	1
<b>Node 2 msg (sent signal)</b>	$+\rho$	$+\rho$	$-\rho$	$-\rho$	$+\rho$	$-\rho$	$+\rho$

**Table 19: Node 3 message**

	Info bits				Parity bits		
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
<b>Node 3 msg (binary)</b>	1	0	0	0	1	1	1
<b>Node 3 msg (sent signal)</b>	$+\rho$	$-\rho$	$-\rho$	$-\rho$	$+\rho$	$+\rho$	$+\rho$

The relay will receive the following mixed signal:

**Table 20: Received mixed signal at relay**

	Info bits				Parity bits		
	$m_1$	$m_2$	$m_3$	$m_4$	$m_5$	$m_6$	$m_7$
<b>Received signal at relay</b>	$+\rho$	$-\rho$	$-\rho$	$-\rho$	$+3\rho$	$+\rho$	$+\rho$

Since fading and noise are not taken into consideration in this example, each node will receive the same mixed signal when the relay forwards the signal. In the following sub-sections, the decoding and signal separation process for erasures.

### 3.3.1.1. *Signal Separation at Node 1*

After receiving the mixed signal, Node 1 will deduct its message and end up with a combined signal that is equal to  $i_i = m_i - a_i = b_i + c_i$  with the following:

**Table 21: Received mixed signal at Node 1 after deducting its message**

	Info bits				Parity bits		
	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$	$i_7$
<b>Combined signal (<math>b_i + c_i</math>)</b>	$+2\rho$	0	$-2\rho$	$-2\rho$	$+2\rho$	0	$+2\rho$

We can simply show that:

When  $(b_i + c_i) = 0 \rightarrow$  we have an erasure

When  $(b_i + c_i) = +2\rho \rightarrow b_i = +\rho$  &  $c_i = +\rho$

When  $(b_i + c_i) = -2\rho \rightarrow b_i = -\rho$  &  $c_i = -\rho$

Thus,  $b_i$  &  $c_i$  can be reduced as follows:

**Table 22: Initial estimation of Node 2 message at Node 1**

	Info bits				Parity bits		
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$
<b><math>b_i</math> (signal)</b>	$+2\rho$	0	$-2\rho$	$-2\rho$	$+2\rho$	0	$+2\rho$
<b><math>b_i</math> (binary)</b>	1	e	0	0	1	e	1

**Table 23: Initial estimation of Node 3 message at Node 1**

	Info bits				Parity bits		
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
<b><math>c_i</math> (signal)</b>	$+2\rho$	0	$-2\rho$	$-2\rho$	$+2\rho$	0	$+2\rho$
<b><math>c_i</math> (binary)</b>	1	e	0	0	1	e	1

Where e stands for erasure.

We can note that the total number of erased bits is four bits (two bits per node). Thus, we need 4 orthogonal/independent linear equations to find the original values of the erased bits. The following are binary linear equations that we can conclude from the combined signal:

$$b_5 = b_2 + b_3 + b_4 \tag{3.26}$$

$$b_6 = b_1 + b_2 + b_3 \quad (3.27)$$

$$b_7 = b_1 + b_3 + b_4 \quad (3.28)$$

$$c_5 = c_1 + c_3 + c_4 \quad (3.29)$$

$$c_6 = c_1 + c_2 + c_3 \quad (3.30)$$

$$c_7 = c_1 + c_2 + c_4 \quad (3.31)$$

$$b_2 = c_2 + 1 \quad (3.32)$$

$$b_6 = c_6 + 1 \quad (3.33)$$

$$b_1 = c_1 = 1 \quad (3.34)$$

$$b_3 = c_3 = 0 \quad (3.35)$$

$$b_4 = c_4 = 0 \quad (3.36)$$

$$b_5 = c_5 = 1 \quad (3.37)$$

$$b_7 = c_7 = 1 \quad (3.38)$$

It is good to mention that the first six equations are taken from the code itself, the next two equations are based on the fact that an erased bit means that the original combined binary bits are different. The equations can be reduced further to the following:

$$1 = b_2 + 0 + 0 \quad (3.39)$$

$$b_6 = 1 + b_2 + 0 \quad (3.40)$$

$$b_2 = c_2 + 1 \quad (3.41)$$

$$b_6 = c_6 + 1 \quad (3.42)$$

And thus:

$$b_2 = 1 \quad (3.43)$$

$$b_6 = 0 \quad (3.44)$$

$$c_2 = 0 \quad (3.45)$$

$$c_6 = 1 \quad (3.46)$$

And the decoded messages at Node 1 are:

**Table 24: Node 2 decoded message at Node 1**

	Info bits				Parity bits		
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$
$b_i$ (decoded-binary)	1	1	0	0	1	0	1

**Table 25: Node 3 decoded message at Node 1**

	Info bits				Parity bits		
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
$c_i$ (decoded-binary)	1	0	0	0	1	1	1

### 3.3.1.2. Signal Separation at Node 2

After receiving the mixed signal, Node 2 will deduct its message and end up with a combined signal that is equal to  $j_i = m_i - b_i = a_i + c_i$  with the following:

**Table 26: Received mixed signal at Node 2 after deducting its message**

	Info bits				Parity bits		
	$j_1$	$j_2$	$j_3$	$j_4$	$j_5$	$j_6$	$j_7$
Combined signal ( $a_i + c_i$ )	0	$-2\rho$	0	0	$+2\rho$	$+2\rho$	0

We can simply show that:

When  $(a_i + c_i) = 0 \rightarrow$  we have an erasure

When  $(a_i + c_i) = +2\rho \rightarrow a_i = +\rho$  &  $c_i = +\rho$

When  $(a_i + c_i) = -2\rho \rightarrow a_i = -\rho$  &  $c_i = -\rho$

Thus,  $a_i$  &  $c_i$  can be reduced as follows:

**Table 27: Initial estimation of Node 1 message at Node 2**

	Info bits				Parity bits		
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$
$a_i$ (signal)	0	$-2\rho$	0	0	$+2\rho$	$+2\rho$	0
$a_i$ (binary)	e	0	e	e	1	1	e

**Table 28: Initial estimation of Node 3 message at Node 2**

	Info bits				Parity bits		
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
$c_i$ (signal)	0	$-2\rho$	0	0	$+2\rho$	$+2\rho$	0
$c_i$ (binary)	e	0	e	e	1	1	e

Where e stands for erasure.

We can note that the total number of erased bits is eight bits (four bits per node). Thus, we need 8 orthogonal/independent linear equations to find the original values of the erased bits. The following are binary linear equations that we can conclude from the combined signal:

$$a_5 = a_1 + a_2 + a_3 \quad (3.47)$$

$$a_6 = a_1 + a_2 + a_4 \quad (3.48)$$

$$a_7 = a_2 + a_3 + a_4 \quad (3.49)$$

$$c_5 = c_1 + c_3 + c_4 \quad (3.50)$$

$$c_6 = c_1 + c_2 + c_3 \quad (3.51)$$

$$c_7 = c_1 + c_2 + c_4 \quad (3.52)$$

$$a_1 = c_1 + 1 \quad (3.53)$$

$$a_3 = c_3 + 1 \quad (3.54)$$

$$a_4 = c_4 + 1 \quad (3.55)$$

$$a_7 = c_7 + 1 \quad (3.56)$$

$$a_2 = c_2 = 0 \quad (3.57)$$

$$a_5 = c_5 = 1 \quad (3.58)$$

$$a_6 = c_6 = 1 \quad (3.59)$$

It is good to mention that the first six equations are taken from the code itself, the next four equations are based on the fact that an erased bit means that the original combined binary bits are different. The equations can be reduced further to the following:

$$1 = a_1 + a_3 \quad (3.60)$$

$$1 = a_1 + a_4 \quad (3.61)$$

$$a_7 = a_3 + a_4 \quad (3.62)$$

$$1 = c_1 + c_3 + c_4 \quad (3.63)$$

$$1 = c_1 + c_3 \quad (3.64)$$

$$c_7 = c_1 + c_4 \quad (3.65)$$

$$a_1 = c_1 + 1 \quad (3.66)$$

$$a_3 = c_3 + 1 \quad (3.67)$$

$$a_4 = c_4 + 1 \quad (3.68)$$

$$a_7 = c_7 + 1 \quad (3.69)$$

And by substituting (7-10) in (4-6), we will end up with:

$$1 = a_1 + a_3 \quad (3.70)$$

$$1 = a_1 + a_4 \quad (3.71)$$

$$a_7 = a_3 + a_4 \quad (3.72)$$

$$1 = a_1 + a_3 + a_4 + 1 \quad (3.73)$$

$$1 = a_1 + a_3 \quad (3.74)$$

$$a_7 + 1 = a_1 + a_4 \quad (3.75)$$

The latter equations can be easily solved to reach the following result:

$$a_1 = 0 \quad (3.76)$$

$$a_4 = 1 \quad (3.77)$$

$$a_7 = 0 \quad (3.78)$$

$$a_3 = 1 \quad (3.79)$$

$$c_1 = 1 \quad (3.80)$$

$$c_4 = 0 \quad (3.81)$$

$$c_7 = 1 \quad (3.82)$$

$$c_3 = 0 \quad (3.83)$$

And the decoded messages at Node 2 are:

**Table 29: Node 1 decoded message at Node 2**

	Info bits				Parity bits		
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$
$a_i$ (decoded-binary)	0	0	1	1	1	1	0

**Table 30: Node 3 decoded message at Node 2**

	Info bits				Parity bits		
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
$c_i$ (decoded-binary)	1	0	0	0	1	1	1

### 3.3.1.3. Signal Separation at Node 3

After receiving the mixed signal, Node 3 will deduct its message and end up with a combined signal that is equal to  $k_i = m_i - c_i = a_i + b_i$  with the following:

**Table 31: The recieved mixed signal at node 3 after deducting its message**

	Info bits				Parity bits		
	$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$
Combined signal ( $a_i + b_i$ )	0	0	0	0	$+2\rho$	0	0

We can simply show that:

When  $(a_i + b_i) = 0 \rightarrow$  we have an erasure

When  $(a_i + b_i) = +2\rho \rightarrow a_i = +\rho$  &  $b_i = +\rho$

When  $(a_i + b_i) = -2\rho \rightarrow a_i = -\rho$  &  $b_i = -\rho$

Thus,  $a_i$  &  $b_i$  can be reduced as follows:

**Table 32: Initial estimation of Node 1 message at Node 3**

	Info bits				Parity bits		
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$
$a_i$ (signal)	0	0	0	0	$+2\rho$	0	0
$a_i$ (binary)	e	e	e	e	1	e	e



**Table 33: Initial estimation of Node 2 message at Node 3**

	Info bits				Parity bits		
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$
$b_i$ (signal)	0	0	0	0	$+2\rho$	0	0
$b_i$ (binary)	e	e	e	e	1	e	e

Where e stands for erasure.

We can note that the total number of erased bits is twelve bits (six bits per node). Thus, we need 12 orthogonal/independent linear equations to find the original values of the erased bits. The following are binary linear equations that we can conclude from the combined signal:

$$a_5 = a_1 + a_2 + a_3 \quad (3.84)$$

$$a_6 = a_1 + a_2 + a_4 \quad (3.85)$$

$$a_7 = a_2 + a_3 + a_4 \quad (3.86)$$

$$b_5 = b_2 + b_3 + b_4 \quad (3.87)$$

$$b_6 = b_1 + b_2 + b_3 \quad (3.88)$$

$$b_7 = b_1 + b_3 + b_4 \quad (3.89)$$

$$a_1 = b_1 + 1 \quad (3.90)$$

$$a_2 = b_2 + 1 \quad (3.91)$$

$$a_3 = b_3 + 1 \quad (3.92)$$

$$a_4 = b_4 + 1 \quad (3.93)$$

$$a_6 = b_6 + 1 \quad (3.94)$$

$$a_7 = b_7 + 1 \quad (3.95)$$

$$a_5 = b_5 = 1 \quad (3.96)$$

It is good to mention that the first six equations are taken from the code itself, the next seven equations are based on the fact that an erased bit means that the original combined binary bits are different. The equations can be reduced further to the following:

$$1 = a_1 + a_2 + a_3 \quad (3.97)$$

$$a_6 = a_1 + a_2 + a_4 \quad (3.98)$$

$$a_7 = a_2 + a_3 + a_4 \quad (3.99)$$

$$1 = b_2 + b_3 + b_4 \quad (3.100)$$

$$b_6 = b_1 + b_2 + b_3 \quad (3.101)$$

$$b_7 = b_1 + b_3 + b_4 \quad (3.102)$$

$$a_1 = b_1 + 1 \quad (3.103)$$

$$a_2 = b_2 + 1 \quad (3.104)$$

$$a_3 = b_3 + 1 \quad (3.105)$$

$$a_4 = b_4 + 1 \quad (3.106)$$

$$a_6 = b_6 + 1 \quad (3.107)$$

$$a_7 = b_7 + 1 \quad (3.108)$$

And by substituting (7-12) in (4-6), we will end up with:

$$1 = a_1 + a_2 + a_3 \quad (3.109)$$

$$a_6 = a_1 + a_2 + a_4 \quad (3.110)$$

$$a_7 = a_2 + a_3 + a_4 \quad (3.111)$$

$$0 = a_2 + a_3 + a_4 \quad (3.112)$$

$$a_6 = a_1 + a_2 + a_3 \quad (3.113)$$

$$a_7 = a_1 + a_3 + a_4 \quad (3.114)$$

The latter equations can be easily solved to reach the following result:

$$a_1 = 0 \quad (3.115)$$

$$a_2 = 0 \quad (3.116)$$

$$a_3 = 1 \quad (3.117)$$

$$a_4 = 1 \quad (3.118)$$

$$a_6 = 1 \quad (3.119)$$

$$a_7 = 0 \quad (3.120)$$

$$b_1 = 1 \tag{3.121}$$

$$b_2 = 1 \tag{3.122}$$

$$b_3 = 0 \tag{3.123}$$

$$b_4 = 0 \tag{3.124}$$

$$b_6 = 0 \tag{3.125}$$

$$b_7 = 1 \tag{3.126}$$

And the decoded messages at Node 3 are:

**Table 34: Node 1 decoded message at Node 3**

	Info bits				Parity bits		
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$
<b><math>a_i</math> (decoded-binary)</b>	0	0	1	1	1	1	0

**Table 35: Node 2 decoded message at Node 3**

	Info bits				Parity bits		
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$
<b><math>b_i</math> (decoded-binary)</b>	1	1	0	0	1	0	1

As seen from this example, we successfully retrieved the messages from the other two senders by solving the linear system of equations that consists of parity relationships that correlate the bits of one sender and erasure relations that correlate bits of one sender with the others. The performance at decoder 1 is slightly lower than that of decoder 2 and 3, and is only able to decode up to 5 erasures. The encoders used in this example were chosen for the simplicity of their visual representation using bubble notation. In the following section, we will use optimal encoder/decoder design that guarantees optimal performance for solving erasures all nodes.

### 3.3.2. Algorithm for Decoding Using Array Manipulation

In this section, we will look at how we can decode using simple matrix manipulation and algebraic method to decoder at each receiver. Computer simulation results presented later on in

this chapter use this algorithm. Let's first look at the generator matrices used at each sender used in this example:

$$G_{Node\ 1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (3.127)$$

$$G_{Node\ 2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (3.128)$$

$$G_{Node\ 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (3.129)$$

To encode, each sender uses its generator matrix to produce a codeword as follows:

$$c_{Node\ i} = \vec{X}_i \times G_{Node\ i} \quad (3.130)$$

Let's consider the same example discussed in the previous section where that the same codewords were generated. I.e.  $c_{Node\ 1} = \mathbf{a} = [0011110]$ ,  $c_{Node\ 2} = \mathbf{b} = [1000001]$  and  $c_{Node\ 3} = \mathbf{c} = [0101110]$ .

### 3.3.2.1. Decoding at Node 1

From the generator matrices in the previous section, the decoder knows the following:

$$b_2 + b_3 + b_4 + b_5 = 0 \quad (3.131)$$

$$b_2 + b_3 + b_4 + b_6 = 0 \quad (3.132)$$

$$b_1 + b_3 + b_4 + b_7 = 0 \quad (3.133)$$

This is translated to the following matrix notation:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.134)$$

And,

$$c_1 + c_3 + c_4 + c_5 = 0 \quad (3.135)$$

$$c_1 + c_2 + c_3 + c_6 = 0 \quad (3.136)$$

$$c_1 + c_2 + c_4 + c_6 = 0 \quad (3.137)$$

Which translates to the following matrix notation:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.138)$$

Since  $r_i^{at Node 1} = b_i \oplus c_i$ , after demodulating the signal, and subtracting its own message, Node 1 ends up with the following erasure pattern:

$$\mathbf{r}^{at Node 1} = [ee0eeee] \quad (3.139)$$

From this, the decoder knows that:

$$b_1 = 1 \oplus c_1 \quad (3.140)$$

$$b_2 = 1 \oplus c_2 \quad (3.141)$$

$$b_3 = 0 \quad (3.142)$$

$$b_4 = 1 \oplus c_4 \quad (3.143)$$

$$b_5 = 1 \oplus c_5 \quad (3.144)$$

$$b_6 = 1 \oplus c_6 \quad (3.145)$$

$$b_7 = 1 \oplus c_7 \quad (3.146)$$

And thereby, we can rewrite the matrix relation in equation (3.138) can be rewritten as follows:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.147)$$

Then the decoder merges the matrices in (3.134) & (3.147):

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.148)$$

The decoder, then, solves the system of equations of linear equations using matrix manipulation by first getting the upper triangle conversion:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.149)$$

After that, the decoder manipulates the matrix to get a mostly diagonal matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \quad (3.150)$$

Then, the decoder traverses the rows of the matrix twice; in the first, it looks for rows that has only one unknown, solves for it, and stores the guessed bit in the erasure pattern. In our example, we can see that row three has one unknown ( $b_7$ ). After solving for  $b_7$ , the error pattern will be  $\mathbf{b} = [ee0eee1]$ . Now we have at most one unknown in each row of the matrix. The decoder traverses the matrix once more and solves for all the unknowns, and gets  $c_{Node\ 2} = \mathbf{b} = [1000001]$ . Finally, the decoder uses this information to get  $c_{Node\ 3} = \mathbf{c} = [0101110]$ .

### 3.3.2.1. Decoding at Node 2

The same procedure is followed in Node 2. The decoder knows the following:

$$a_1 + a_2 + a_3 + a_5 = 0 \quad (3.151)$$

$$a_1 + a_2 + a_4 + a_6 = 0 \quad (3.152)$$

$$a_2 + a_3 + a_4 + a_7 = 0 \quad (3.153)$$

This is translated to the following matrix notation:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.154)$$

And,

$$c_1 + c_3 + c_4 + c_5 = 0 \quad (3.155)$$

$$c_1 + c_2 + c_3 + c_6 = 0 \quad (3.156)$$

$$c_1 + c_2 + c_4 + c_6 = 0 \quad (3.157)$$

Which translates to the following matrix notation:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.158)$$

Since  $r_i^{at Node 2} = a_i \oplus c_i$ , after demodulating the signal, and subtracting its own message, Node 1 ends up with the following erasure pattern:

$$\mathbf{r}^{at Node 2} = [0ee1110] \quad (3.159)$$

And thereby, we can rewrite the matrix relation in equation (3.158) can be rewritten as follows:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix} \quad (3.160)$$

Then the decoder merges the matrices in (3.154) & (3.160):



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad (3.161)$$

The decoder, then, solves the system of equations of linear equations using matrix manipulation by getting mostly diagonal matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (3.162)$$

Then, the decoder traverses the rows of the matrix twice and solves for erasures to get  $c_{Node\ 1} = \mathbf{a} = [0011110]$ . Finally, the decoder uses this information to get  $c_{Node\ 3} = \mathbf{c} = [0101110]$ .

### 3.3.2.1. Decoding at Node 3

The same procedure is followed at Node 3. The decoder knows the following:

$$a_1 + a_2 + a_3 + a_5 = 0 \quad (3.163)$$

$$a_1 + a_2 + a_4 + a_6 = 0 \quad (3.164)$$

$$a_2 + a_3 + a_4 + a_7 = 0 \quad (3.165)$$

This is translated to the following matrix notation:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.166)$$

And,

$$b_2 + b_3 + b_4 + b_5 = 0 \quad (3.167)$$

$$b_2 + b_3 + b_4 + b_6 = 0 \quad (3.168)$$

$$b_1 + b_3 + b_4 + b_7 = 0 \quad (3.169)$$

This is translated to the following matrix notation:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.170)$$

Since  $r_i^{at Node 3} = a_i \oplus b_i$ , after demodulating the signal, and subtracting its own message, Node 1 ends up with the following erasure pattern:

$$\mathbf{r}^{at Node 3} = [e0e0000] \quad (3.171)$$

And thereby, we can rewrite the matrix relation in equation (3.171) can be rewritten as follows:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.172)$$

Then the decoder merges the matrices in (3.166) & (3.172):

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.173)$$

The decoder, then, solves the system of equations of linear equations using matrix manipulation by getting mostly diagonal matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (3.174)$$

Then, the decoder traverses the rows of the matrix twice and solves for erasures to get  $c_{Node\ 1} = \mathbf{a} = [0011110]$ . Finally, the decoder uses this information to get  $c_{Node\ 2} = \mathbf{b} = [1000001]$ .

### 3.3.3. Algorithm for Decoding Using Table Lookup

Since there is a one to one relationship between the received erasure pattern at each receiver and the sent codewords, we can decode using lookup tables. The lookup tables are shown in Table 36, Table 37 and Table 38 for Node 1, Node 2 and Node 3 respectively. If we consider the same example in section 3.3.2, the decoded codewords at each receiver would be:

- Node 1: Erasure pattern is [ee0eeee]:
  - Decoded Node 2 message: 1000001
  - Decoded Node 3 message: 0101110
- Node 2: Erasure pattern is [0ee1110]:
  - Decoded Node 1 message: 0011110
  - Decoded Node 3 message: 0101110
- Node 3: Erasure pattern is [e0eeeeee]:
  - Decoded Node 1 message: 0011110
  - Decoded Node 2 message: 1000001

**Table 36: Decoding lookup table at Node 1**

Node 2 Info. Seq.	Node 3 Info. Seq.	Erasure Pattern	Node 2 Codeword	Node 3 Codeword	Num of erased bits	Decodable
<b>Sequences with number of erasures equal to 0</b>						
0 (0000)	0 (0000)	_____	0000000	0000000	0	yes
15 (1111)	15 (1111)	_____	1111111	1111111	0	yes
<b>Sequences with number of erasures equal to 1</b>						
1 (0001)	1 (0001)	___e_	0001111	0001101	1	yes
2 (0010)	2 (0010)	___e	0010111	0010110	1	yes
3 (0011)	7 (0111)	__e__	0011000	0111000	1	yes
4 (0100)	5 (0101)	__e__	0100110	0101110	1	yes
5 (0101)	13 (1101)	e_____	0101001	1101001	1	yes
6 (0110)	6 (0110)	__e_	0110001	0110101	1	yes
7 (0111)	5 (0101)	__e__	0111110	0101110	1	yes
8 (1000)	10 (1010)	__e_	1000001	1010001	1	yes
9 (1001)	9 (1001)	___e_	1001110	1001010	1	yes
10 (1010)	2 (0010)	e_____	1010110	0010110	1	yes
11 (1011)	10 (1010)	__e_	1011001	1010001	1	yes
12 (1100)	8 (1000)	_e_____	1100111	1000111	1	yes
13 (1101)	13 (1101)	___e	1101000	1101001	1	yes
14 (1110)	14 (1110)	___e	1110000	1110010	1	yes
<b>Sequences with number of erasures equal to 2</b>						
1 (0001)	3 (0011)	__e_e_	0001111	0011011	2	yes
1 (0001)	5 (0101)	_e___e	0001111	0101110	2	yes
1 (0001)	8 (1000)	e_e_____	0001111	1000111	2	yes
2 (0010)	3 (0011)	___ee_	0010111	0011011	2	yes
2 (0010)	6 (0110)	_e__e_	0010111	0110101	2	yes
2 (0010)	8 (1000)	e_e_____	0010111	1000111	2	yes
3 (0011)	0 (0000)	___ee_	0011000	0000000	2	yes
3 (0011)	3 (0011)	___ee	0011000	0011011	2	yes
3 (0011)	11 (1011)	e__e__	0011000	1011100	2	yes
4 (0100)	2 (0010)	__ee_	0100110	0010110	2	yes
4 (0100)	4 (0100)	___e_e	0100110	0100011	2	yes
4 (0100)	12 (1100)	e__e_	0100110	1100100	2	yes
5 (0101)	1 (0001)	__e_e_	0101001	0001101	2	yes
5 (0101)	4 (0100)	___e_e	0101001	0100011	2	yes
5 (0101)	7 (0111)	__e__e	0101001	0111000	2	yes
6 (0110)	4 (0100)	_e__e_	0110001	0100011	2	yes
6 (0110)	7 (0111)	___e_e	0110001	0111000	2	yes
6 (0110)	10 (1010)	ee_____	0110001	1010001	2	yes
7 (0111)	2 (0010)	_e__e_	0111110	0010110	2	yes
7 (0111)	7 (0111)	___ee	0111110	0111000	2	yes
7 (0111)	15 (1111)	e_____	0111110	1111111	2	yes
8 (1000)	0 (0000)	e_____	1000001	0000000	2	yes
8 (1000)	8 (1000)	___ee_	1000001	1000111	2	yes

8 (1000)	13 (1101)	_e_e__	1000001	1101001	2	yes
9 (1001)	5 (0101)	ee___	1001110	0101110	2	yes
9 (1001)	8 (1000)	__e_e_e	1001110	1000111	2	yes
9 (1001)	11 (1011)	__e_e_e	1001110	1011100	2	yes
10 (1010)	8 (1000)	__e_e_e	1010110	1000111	2	yes
10 (1010)	11 (1011)	__e_e_e	1010110	1011100	2	yes
10 (1010)	14 (1110)	__e_e_e	1010110	1110010	2	yes
11 (1011)	3 (0011)	e___e	1011001	0011011	2	yes
11 (1011)	11 (1011)	__e_e_e	1011001	1011100	2	yes
11 (1011)	13 (1101)	__ee___	1011001	1101001	2	yes
12 (1100)	4 (0100)	e__e__	1100111	0100011	2	yes
12 (1100)	12 (1100)	_____ee	1100111	1100100	2	yes
12 (1100)	15 (1111)	__ee___	1100111	1111111	2	yes
13 (1101)	7 (0111)	e_e____	1101000	0111000	2	yes
13 (1101)	9 (1001)	__e_e_e	1101000	1001010	2	yes
13 (1101)	12 (1100)	_____ee	1101000	1100100	2	yes
14 (1110)	7 (0111)	e__e__	1110000	0111000	2	yes
14 (1110)	10 (1010)	__e_e_e	1110000	1010001	2	yes
14 (1110)	12 (1100)	__e_e_e	1110000	1100100	2	yes
Sequences with number of erasures equal to 3						
0 (0000)	1 (0001)	__ee_e	0000000	0001101	3	yes
0 (0000)	2 (0010)	__e_ee_	0000000	0010110	3	yes
0 (0000)	4 (0100)	__e__ee	0000000	0100011	3	yes
0 (0000)	7 (0111)	__eee__	0000000	0111000	3	yes
0 (0000)	9 (1001)	e__e_e	0000000	1001010	3	yes
0 (0000)	10 (1010)	e__e_e	0000000	1010001	3	yes
0 (0000)	12 (1100)	ee__e__	0000000	1100100	3	yes
1 (0001)	2 (0010)	__ee_e	0001111	0010110	3	yes
1 (0001)	4 (0100)	__e_ee_	0001111	0100011	3	yes
1 (0001)	9 (1001)	e__e_e	0001111	1001010	3	yes
1 (0001)	15 (1111)	eee___	0001111	1111111	3	yes
2 (0010)	1 (0001)	__ee_e	0010111	0001101	3	yes
2 (0010)	4 (0100)	__ee_e__	0010111	0100011	3	yes
2 (0010)	10 (1010)	e__ee__	0010111	1010001	3	yes
2 (0010)	15 (1111)	ee_e__	0010111	1111111	3	yes
3 (0011)	1 (0001)	__e_e_e	0011000	0001101	3	yes
3 (0011)	2 (0010)	_____eee	0011000	0010110	3	yes
3 (0011)	9 (1001)	e__e_e	0011000	1001010	3	yes
3 (0011)	10 (1010)	e__e_e	0011000	1010001	3	yes
4 (0100)	0 (0000)	__e_ee_	0100110	0000000	3	yes
4 (0100)	6 (0110)	__e_ee_	0100110	0110101	3	yes
4 (0100)	8 (1000)	ee___e	0100110	1000111	3	yes
4 (0100)	14 (1110)	e_e_e__	0100110	1110010	3	yes
5 (0101)	0 (0000)	__e_e_e	0101001	0000000	3	yes
5 (0101)	3 (0011)	__ee_e__	0101001	0011011	3	yes
5 (0101)	5 (0101)	_____eee	0101001	0101110	3	yes
5 (0101)	6 (0110)	__eee__	0101001	0110101	3	yes
6 (0110)	0 (0000)	__ec__e	0110001	0000000	3	yes
6 (0110)	3 (0011)	__e_e_e__	0110001	0011011	3	yes
6 (0110)	13 (1101)	e__ee__	0110001	1101001	3	yes
6 (0110)	14 (1110)	e__ee__	0110001	1110010	3	yes
7 (0111)	3 (0011)	__e_e_e	0111110	0011011	3	yes
7 (0111)	6 (0110)	__e_ee_	0111110	0110101	3	yes
7 (0111)	11 (1011)	ee__e__	0111110	1011100	3	yes
7 (0111)	14 (1110)	e__ee__	0111110	1110010	3	yes
8 (1000)	1 (0001)	e__ee__	1000001	0001101	3	yes
8 (1000)	4 (0100)	ee__e__	1000001	0100011	3	yes
8 (1000)	9 (1001)	__e_ee_	1000001	1001010	3	yes
8 (1000)	12 (1100)	__e_e_e	1000001	1100100	3	yes
9 (1001)	1 (0001)	e__ee__	1001110	0001101	3	yes
9 (1001)	2 (0010)	e__ee__	1001110	0010110	3	yes
9 (1001)	12 (1100)	__e_e_e	1001110	1100100	3	yes
9 (1001)	15 (1111)	__ee_e	1001110	1111111	3	yes
10 (1010)	9 (1001)	_____eee	1010110	1001010	3	yes
10 (1010)	10 (1010)	_____eee	1010110	1010001	3	yes
10 (1010)	12 (1100)	__ee_e__	1010110	1100100	3	yes
10 (1010)	15 (1111)	__e_e_e	1010110	1111111	3	yes
11 (1011)	1 (0001)	e_e_e__	1011001	0001101	3	yes

11 (1011)	7 (0111)	ee__e	1011001	0111000	3	yes
11 (1011)	9 (1001)	__e__ee	1011001	1001010	3	yes
11 (1011)	15 (1111)	__e__ee	1011001	1111111	3	yes
12 (1100)	5 (0101)	e_e_e_e	1100111	0101110	3	yes
12 (1100)	6 (0110)	e_e_e_e	1100111	0110101	3	yes
12 (1100)	13 (1101)	__eee	1100111	1101001	3	yes
12 (1100)	14 (1110)	__e_e_e	1100111	1110010	3	yes
13 (1101)	0 (0000)	ee_e__	1101000	0000000	3	yes
13 (1101)	5 (0101)	e__ee	1101000	0101110	3	yes
13 (1101)	11 (1011)	__ee_e__	1101000	1011100	3	yes
13 (1101)	14 (1110)	__ee_e	1101000	1110010	3	yes
14 (1110)	0 (0000)	eee__	1110000	0000000	3	yes
14 (1110)	6 (0110)	e__e_e	1110000	0110101	3	yes
14 (1110)	11 (1011)	__e__ee	1110000	1011100	3	yes
14 (1110)	13 (1101)	__ee_e	1110000	1101001	3	yes
15 (1111)	3 (0011)	ee_e__	1111111	0011011	3	yes
15 (1111)	5 (0101)	e_e_e	1111111	0101110	3	yes
15 (1111)	6 (0110)	e_e_e	1111111	0110101	3	yes
15 (1111)	8 (1000)	__eee	1111111	1000111	3	yes
15 (1111)	11 (1011)	__e__ee	1111111	1011100	3	yes
15 (1111)	13 (1101)	__e__ee	1111111	1101001	3	yes
15 (1111)	14 (1110)	__ee_e	1111111	1110010	3	yes
Sequences with number of erasures equal to 4						
0 (0000)	3 (0011)	__ee__ee	0000000	0011011	4	yes
0 (0000)	5 (0101)	__e__eee	0000000	0101110	4	yes
0 (0000)	6 (0110)	__ee_e_e	0000000	0110101	4	yes
0 (0000)	8 (1000)	e__eee	0000000	1000111	4	yes
0 (0000)	11 (1011)	e__eee	0000000	1011100	4	yes
0 (0000)	13 (1101)	ee_e_e	0000000	1101001	4	yes
0 (0000)	14 (1110)	eee_e	0000000	1110010	4	yes
1 (0001)	0 (0000)	__eeeee	0001111	0000000	4	yes
1 (0001)	6 (0110)	__eee_e	0001111	0110101	4	yes
1 (0001)	11 (1011)	e_e__ee	0001111	1011100	4	yes
1 (0001)	13 (1101)	ee__ee	0001111	1101001	4	yes
2 (0010)	0 (0000)	__e__eee	0010111	0000000	4	yes
2 (0010)	5 (0101)	__eee_e	0010111	0101110	4	yes
2 (0010)	11 (1011)	e__e__ee	0010111	1011100	4	yes
2 (0010)	14 (1110)	ee_e_e	0010111	1110010	4	yes
3 (0011)	5 (0101)	__ee__ee	0011000	0101110	4	yes
3 (0011)	6 (0110)	__e__ee_e	0011000	0110101	4	yes
3 (0011)	13 (1101)	eee__e	0011000	1101001	4	yes
3 (0011)	14 (1110)	ee_e_e	0011000	1110010	4	yes
4 (0100)	1 (0001)	__e_e__ee	0100110	0001101	4	yes
4 (0100)	7 (0111)	__eeeee	0100110	0111000	4	yes
4 (0100)	9 (1001)	ee__ee	0100110	1001010	4	yes
4 (0100)	15 (1111)	e__ee_e	0100110	1111111	4	yes
5 (0101)	9 (1001)	ee__ee	0101001	1001010	4	yes
5 (0101)	10 (1010)	eeeee	0101001	1010001	4	yes
5 (0101)	12 (1100)	e__ee_e	0101001	1100100	4	yes
5 (0101)	15 (1111)	e_e__ee	0101001	1111111	4	yes
6 (0110)	1 (0001)	__eeeee	0110001	0001101	4	yes
6 (0110)	2 (0010)	__e__eee	0110001	0010110	4	yes
6 (0110)	12 (1100)	e_e_e_e	0110001	1100100	4	yes
6 (0110)	15 (1111)	e__eee	0110001	1111111	4	yes
7 (0111)	1 (0001)	__ee__ee	0111110	0001101	4	yes
7 (0111)	4 (0100)	__eee_e	0111110	0100011	4	yes
7 (0111)	9 (1001)	eee_e__	0111110	1001010	4	yes
7 (0111)	12 (1100)	e__ee_e	0111110	1100100	4	yes
8 (1000)	3 (0011)	e__ee_e	1000001	0011011	4	yes
8 (1000)	6 (0110)	eee_e__	1000001	0110101	4	yes
8 (1000)	11 (1011)	__eee_e	1000001	1011100	4	yes
8 (1000)	14 (1110)	__ee__ee	1000001	1110010	4	yes
9 (1001)	0 (0000)	e__eee	1001110	0000000	4	yes
9 (1001)	3 (0011)	e_e_e_e	1001110	0011011	4	yes
9 (1001)	13 (1101)	__e__eee	1001110	1101001	4	yes
9 (1001)	14 (1110)	__eeeee	1001110	1110010	4	yes
10 (1010)	0 (0000)	e_e__ee	1010110	0000000	4	yes
10 (1010)	3 (0011)	e__ee_e	1010110	0011011	4	yes

10 (1010)	5 (0101)	eeee__	1010110	0101110	4	yes
10 (1010)	6 (0110)	ee__ee	1010110	0110101	4	yes
11 (1011)	0 (0000)	e_ee_e	1011001	0000000	4	yes
11 (1011)	6 (0110)	ee_ee_	1011001	0110101	4	yes
11 (1011)	8 (1000)	__eeee_	1011001	1000111	4	yes
11 (1011)	14 (1110)	_e_e_ee	1011001	1110010	4	yes
12 (1100)	1 (0001)	ee_e_e_	1100111	0001101	4	yes
12 (1100)	2 (0010)	eee__e	1100111	0010110	4	yes
12 (1100)	9 (1001)	_e_ee_e	1100111	1001010	4	yes
12 (1100)	10 (1010)	__ee_ee	1100111	1010001	4	yes
13 (1101)	1 (0001)	ee_e_e_e	1101000	0001101	4	yes
13 (1101)	4 (0100)	e_e_ee	1101000	0100011	4	yes
13 (1101)	10 (1010)	__eee_e	1101000	1010001	4	yes
13 (1101)	15 (1111)	__e_eee	1101000	1111111	4	yes
14 (1110)	2 (0010)	ee_ee_	1110000	0010110	4	yes
14 (1110)	4 (0100)	e_e_ee	1110000	0100011	4	yes
14 (1110)	9 (1001)	_eee_e_	1110000	1001010	4	yes
14 (1110)	15 (1111)	__eeeee	1110000	1111111	4	yes
15 (1111)	1 (0001)	eee_e_	1111111	0001101	4	yes
15 (1111)	2 (0010)	ee_e_e_e	1111111	0010110	4	yes
15 (1111)	4 (0100)	e_eee_	1111111	0100011	4	yes
15 (1111)	7 (0111)	e___eee	1111111	0111000	4	yes
15 (1111)	9 (1001)	_ee_e_e_e	1111111	1001010	4	yes
15 (1111)	10 (1010)	_e_eee_	1111111	1010001	4	yes
15 (1111)	12 (1100)	__ee_ee	1111111	1100100	4	yes
Sequences with number of erasures equal to 5						
1 (0001)	7 (0111)	_ee_eee	0001111	0111000	5	yes
1 (0001)	10 (1010)	e_eeee_	0001111	1010001	5	yes
1 (0001)	12 (1100)	ee_e_ee	0001111	1100100	5	yes
2 (0010)	7 (0111)	_e_eeee	0010111	0111000	5	yes
2 (0010)	9 (1001)	e_eee_e	0010111	1001010	5	yes
2 (0010)	12 (1100)	eee_ee	0010111	1100100	5	yes
3 (0011)	4 (0100)	__eee_ee	0011000	0100011	5	yes
3 (0011)	12 (1100)	eeeee_	0011000	1100100	5	yes
3 (0011)	15 (1111)	ee_eee	0011000	1111111	5	yes
4 (0100)	3 (0011)	__eeee_e	0100110	0011011	5	yes
4 (0100)	11 (1011)	eeeee_e_	0100110	1011100	5	yes
4 (0100)	13 (1101)	e_eeee	0100110	1101001	5	yes
5 (0101)	8 (1000)	ee_eee_	0101001	1000111	5	yes
5 (0101)	11 (1011)	eee_e_e	0101001	1011100	5	yes
5 (0101)	14 (1110)	e_ee_ee	0101001	1110010	5	yes
6 (0110)	5 (0101)	__eeeee	0110001	0101110	5	yes
6 (0110)	8 (1000)	eee_ee_	0110001	1000111	5	yes
6 (0110)	11 (1011)	ee_ee_e	0110001	1011100	5	yes
7 (0111)	0 (0000)	__eeeee_	0111110	0000000	5	yes
7 (0111)	8 (1000)	eeeee_e	0111110	1000111	5	yes
7 (0111)	13 (1101)	e_e_eee	0111110	1101001	5	yes
8 (1000)	2 (0010)	e_e_eee	1000001	0010110	5	yes
8 (1000)	7 (0111)	eeeee_e	1000001	0111000	5	yes
8 (1000)	15 (1111)	__eeeee_	1000001	1111111	5	yes
9 (1001)	4 (0100)	ee_ee_e	1001110	0100011	5	yes
9 (1001)	7 (0111)	eee_ee_	1001110	0111000	5	yes
9 (1001)	10 (1010)	__eeeee	1001110	1010001	5	yes
10 (1010)	1 (0001)	e_ee_ee	1010110	0001101	5	yes
10 (1010)	4 (0100)	eee_e_e	1010110	0100011	5	yes
10 (1010)	7 (0111)	ee_eee_	1010110	0111000	5	yes
11 (1011)	2 (0010)	e_eeee	1011001	0010110	5	yes
11 (1011)	4 (0100)	eeeee_e_	1011001	0100011	5	yes
11 (1011)	12 (1100)	__eeee_e	1011001	1100100	5	yes
12 (1100)	0 (0000)	ee_eee	1100111	0000000	5	yes
12 (1100)	3 (0011)	eeeee_	1100111	0011011	5	yes
12 (1100)	11 (1011)	__eee_ee	1100111	1011100	5	yes
13 (1101)	3 (0011)	eee_ee	1101000	0011011	5	yes
13 (1101)	6 (0110)	e_eee_e	1101000	0110101	5	yes
13 (1101)	8 (1000)	_e_eeee	1101000	1000111	5	yes
14 (1110)	3 (0011)	ee_e_ee	1110000	0011011	5	yes
14 (1110)	5 (0101)	e_eeeee_	1110000	0101110	5	yes
14 (1110)	8 (1000)	__eee_ee	1110000	1000111	5	yes

Sequences with number of erasures equal to 6						
1 (0001)	14 (1110)	eeeee_e	0001111	1110010	6	yes
2 (0010)	13 (1101)	eeeee_	0010111	1101001	6	yes
3 (0011)	8 (1000)	e_eeeee	0011000	1000111	6	yes
4 (0100)	10 (1010)	eee_eee	0100110	1010001	6	yes
5 (0101)	2 (0010)	_eeeeee	0101001	0010110	6	yes
6 (0110)	9 (1001)	eeee_ee	0110001	1001010	6	yes
7 (0111)	10 (1010)	ee_eeee	0111110	1010001	6	yes
8 (1000)	5 (0101)	ee_eeee	1000001	0101110	6	yes
9 (1001)	6 (0110)	eeee_ee	1001110	0110101	6	yes
10 (1010)	13 (1101)	_eeeeee	1010110	1101001	6	yes
11 (1011)	5 (0101)	eee_eee	1011001	0101110	6	yes
12 (1100)	7 (0111)	e_eeeee	1100111	0111000	6	yes
13 (1101)	2 (0010)	eeeee_	1101000	0010110	6	yes
14 (1110)	1 (0001)	eeeee_e	1110000	0001101	6	yes
Sequences with number of erasures equal to 7						
0 (0000)	15 (1111)	eeeeeee	0000000	1111111	7	no
15 (1111)	0 (0000)	eeeeeee	1111111	0000000	7	no

**Table 37: Decoding lookup table at Node 2**

Node 1 Info. Seq.	Node 3 Info. Seq.	Erasure Pattern	Node 1 Codeword	Node 3 Codeword	Num of erased bits	Decodable
Sequences with number of erasures equal to 0						
0 (0000)	0 (0000)	_____	0000000	0000000	0	yes
15 (1111)	15 (1111)	_____	1111111	1111111	0	yes
Sequences with number of erasures equal to 1						
1 (0001)	3 (0011)	__e__	0001011	0011011	1	yes
2 (0010)	6 (0110)	_e___	0010101	0110101	1	yes
3 (0011)	2 (0010)	___e__	0011110	0010110	1	yes
4 (0100)	4 (0100)	__e_	0100111	0100011	1	yes
5 (0101)	5 (0101)	___e_	0101100	0101110	1	yes
6 (0110)	14 (1110)	e_____	0110010	1110010	1	yes
7 (0111)	7 (0111)	____e	0111001	0111000	1	yes
8 (1000)	8 (1000)	____e	1000110	1000111	1	yes
9 (1001)	1 (0001)	e_____	1001101	0001101	1	yes
10 (1010)	10 (1010)	___e_	1010011	1010001	1	yes
11 (1011)	11 (1011)	___e_	1011000	1011100	1	yes
12 (1100)	13 (1101)	__e__	1100001	1101001	1	yes
13 (1101)	9 (1001)	_e___	1101010	1001010	1	yes
14 (1110)	12 (1100)	__e__	1110100	1100100	1	yes
Sequences with number of erasures equal to 2						
1 (0001)	1 (0001)	___ee	0001011	0001101	2	yes
1 (0001)	4 (0100)	_e_e_	0001011	0100011	2	yes
1 (0001)	9 (1001)	e___e	0001011	1001010	2	yes
2 (0010)	1 (0001)	__ee_	0010101	0001101	2	yes
2 (0010)	2 (0010)	____ee	0010101	0010110	2	yes
2 (0010)	10 (1010)	e___e_	0010101	1010001	2	yes
3 (0011)	3 (0011)	___e_e	0011110	0011011	2	yes
3 (0011)	5 (0101)	__ee__	0011110	0101110	2	yes
3 (0011)	11 (1011)	e___e_	0011110	1011100	2	yes
4 (0100)	5 (0101)	__e_e_	0100111	0101110	2	yes
4 (0100)	6 (0110)	___e_e_	0100111	0110101	2	yes
4 (0100)	8 (1000)	ee_____	0100111	1000111	2	yes
5 (0101)	1 (0001)	_e___e	0101100	0001101	2	yes
5 (0101)	7 (0111)	__e_e_	0101100	0111000	2	yes
5 (0101)	12 (1100)	e_e__	0101100	1100100	2	yes
6 (0110)	2 (0010)	_e_e_	0110010	0010110	2	yes
6 (0110)	4 (0100)	___e_e_	0110010	0100011	2	yes
6 (0110)	7 (0111)	__e_e_	0110010	0111000	2	yes
7 (0111)	3 (0011)	_e___e_	0111001	0011011	2	yes



7 (0111)	6 (0110)	__ee__	0111001	0110101	2	yes
7 (0111)	13 (1101)	e_e__	0111001	1101001	2	yes
8 (1000)	2 (0010)	e_e__	1000110	0010110	2	yes
8 (1000)	9 (1001)	__ee__	1000110	1001010	2	yes
8 (1000)	12 (1100)	_e_e_e	1000110	1100100	2	yes
9 (1001)	8 (1000)	__e_e_e	1001101	1000111	2	yes
9 (1001)	11 (1011)	_e_e_e	1001101	1011100	2	yes
9 (1001)	13 (1101)	_e_e_e	1001101	1101001	2	yes
10 (1010)	3 (0011)	e_e__	1010011	0011011	2	yes
10 (1010)	8 (1000)	_e_e_e	1010011	1000111	2	yes
10 (1010)	14 (1110)	_e_e_e	1010011	1110010	2	yes
11 (1011)	7 (0111)	ee____	1011000	0111000	2	yes
11 (1011)	9 (1001)	_e_e_e	1011000	1001010	2	yes
11 (1011)	10 (1010)	__e_e_e	1011000	1010001	2	yes
12 (1100)	4 (0100)	e_e_e	1100001	0100011	2	yes
12 (1100)	10 (1010)	_ee__	1100001	1010001	2	yes
12 (1100)	12 (1100)	__e_e_e	1100001	1100100	2	yes
13 (1101)	5 (0101)	e_e_e	1101010	0101110	2	yes
13 (1101)	13 (1101)	_____ee	1101010	1101001	2	yes
13 (1101)	14 (1110)	__ee__	1101010	1110010	2	yes
14 (1110)	6 (0110)	e_e_e	1110100	0110101	2	yes
14 (1110)	11 (1011)	_e_e_e	1110100	1011100	2	yes
14 (1110)	14 (1110)	__ee_e	1110100	1110010	2	yes
Sequences with number of erasures equal to 3						
0 (0000)	1 (0001)	__ee_e	0000000	0001101	3	yes
0 (0000)	2 (0010)	_e_ee_	0000000	0010110	3	yes
0 (0000)	4 (0100)	_e_ee	0000000	0100011	3	yes
0 (0000)	7 (0111)	_eee__	0000000	0111000	3	yes
0 (0000)	9 (1001)	e_e_e_e	0000000	1001010	3	yes
0 (0000)	10 (1010)	e_e_e_e	0000000	1010001	3	yes
0 (0000)	12 (1100)	ee_e__	0000000	1100100	3	yes
1 (0001)	0 (0000)	__e_ee	0001011	0000000	3	yes
1 (0001)	5 (0101)	_e_e_e_e	0001011	0101110	3	yes
1 (0001)	8 (1000)	e_ee__	0001011	1000111	3	yes
1 (0001)	13 (1101)	ee____	0001011	1101001	3	yes
2 (0010)	0 (0000)	__e_e_e	0010101	0000000	3	yes
2 (0010)	3 (0011)	__eee__	0010101	0011011	3	yes
2 (0010)	8 (1000)	e_e_e_e	0010101	1000111	3	yes
2 (0010)	11 (1011)	e_e_e_e	0010101	1011100	3	yes
3 (0011)	1 (0001)	_e_ee	0011110	0001101	3	yes
3 (0011)	7 (0111)	_e_ee	0011110	0111000	3	yes
3 (0011)	9 (1001)	e_e_e_e	0011110	1001010	3	yes
3 (0011)	15 (1111)	ee__e	0011110	1111111	3	yes
4 (0100)	1 (0001)	_e_e_e_e	0100111	0001101	3	yes
4 (0100)	2 (0010)	_ee__e	0100111	0010110	3	yes
4 (0100)	12 (1100)	e____e	0100111	1100100	3	yes
4 (0100)	15 (1111)	e_ee__	0100111	1111111	3	yes
5 (0101)	0 (0000)	_e_ee	0101100	0000000	3	yes
5 (0101)	6 (0110)	__ee_e	0101100	0110101	3	yes
5 (0101)	11 (1011)	eee____	0101100	1011100	3	yes
5 (0101)	13 (1101)	e_e_e_e	0101100	1101001	3	yes
6 (0110)	0 (0000)	_ee_e_e	0110010	0000000	3	yes
6 (0110)	3 (0011)	_e_e_e_e	0110010	0011011	3	yes
6 (0110)	5 (0101)	__eee__	0110010	0101110	3	yes
6 (0110)	6 (0110)	_____eee	0110010	0110101	3	yes
7 (0111)	1 (0001)	_ee_e__	0111001	0001101	3	yes
7 (0111)	4 (0100)	__ee_e_e	0111001	0100011	3	yes
7 (0111)	10 (1010)	ee_e__	0111001	1010001	3	yes
7 (0111)	15 (1111)	e____e	0111001	1111111	3	yes
8 (1000)	0 (0000)	e_ee__	1000110	0000000	3	yes
8 (1000)	5 (0101)	ee_e__	1000110	0101110	3	yes
8 (1000)	11 (1011)	__ee_e_e	1000110	1011100	3	yes
8 (1000)	14 (1110)	_ee_e__	1000110	1110010	3	yes
9 (1001)	9 (1001)	_____eee	1001101	1001010	3	yes

9 (1001)	10 (1010)	__eee__	1001101	1010001	3	yes
9 (1001)	12 (1100)	_e_e__e	1001101	1100100	3	yes
9 (1001)	15 (1111)	__ee__e	1001101	1111111	3	yes
10 (1010)	2 (0010)	e___e_e	1010011	0010110	3	yes
10 (1010)	4 (0100)	eee___	1010011	0100011	3	yes
10 (1010)	9 (1001)	__ee__e	1010011	1001010	3	yes
10 (1010)	15 (1111)	_e_ee__	1010011	1111111	3	yes
11 (1011)	0 (0000)	e_ee___	1011000	0000000	3	yes
11 (1011)	3 (0011)	e___ee	1011000	0011011	3	yes
11 (1011)	13 (1101)	__ee__e	1011000	1101001	3	yes
11 (1011)	14 (1110)	_e_e_e_	1011000	1110010	3	yes
12 (1100)	0 (0000)	ee___e	1100001	0000000	3	yes
12 (1100)	6 (0110)	e_e_e__	1100001	0110101	3	yes
12 (1100)	8 (1000)	_e_ee__	1100001	1000111	3	yes
12 (1100)	14 (1110)	__e_ee	1100001	1110010	3	yes
13 (1101)	4 (0100)	e_e_e_e	1101010	0100011	3	yes
13 (1101)	7 (0111)	e_e__e_	1101010	0111000	3	yes
13 (1101)	12 (1100)	___eee_	1101010	1100100	3	yes
13 (1101)	15 (1111)	__e_e_e	1101010	1111111	3	yes
14 (1110)	2 (0010)	ee__e_	1110100	0010110	3	yes
14 (1110)	7 (0111)	e_ee__	1110100	0111000	3	yes
14 (1110)	10 (1010)	_e_e_e_e	1110100	1010001	3	yes
14 (1110)	15 (1111)	__e_ee	1110100	1111111	3	yes
15 (1111)	3 (0011)	ee_e__	1111111	0011011	3	yes
15 (1111)	5 (0101)	e_e__e	1111111	0101110	3	yes
15 (1111)	6 (0110)	e_e_e_e	1111111	0110101	3	yes
15 (1111)	8 (1000)	__eee__	1111111	1000111	3	yes
15 (1111)	11 (1011)	_e___ee	1111111	1011100	3	yes
15 (1111)	13 (1101)	__e_ee_	1111111	1101001	3	yes
15 (1111)	14 (1110)	___ee_e	1111111	1110010	3	yes
Sequences with number of erasures equal to 4						
0 (0000)	3 (0011)	__ee_ee	0000000	0011011	4	yes
0 (0000)	5 (0101)	_e_eee_	0000000	0101110	4	yes
0 (0000)	6 (0110)	__ee_e_e	0000000	0110101	4	yes
0 (0000)	8 (1000)	e___eee	0000000	1000111	4	yes
0 (0000)	11 (1011)	e_eee__	0000000	1011100	4	yes
0 (0000)	13 (1101)	ee_e_e_	0000000	1101001	4	yes
0 (0000)	14 (1110)	eee_e_	0000000	1110010	4	yes
1 (0001)	2 (0010)	___eee_e	0001011	0010110	4	yes
1 (0001)	7 (0111)	__ee_ee	0001011	0111000	4	yes
1 (0001)	10 (1010)	e_ee_e_	0001011	1010001	4	yes
1 (0001)	15 (1111)	eee_e__	0001011	1111111	4	yes
2 (0010)	4 (0100)	__ee_ee	0010101	0100011	4	yes
2 (0010)	7 (0111)	_e_ee_e	0010101	0111000	4	yes
2 (0010)	12 (1100)	eee__e	0010101	1100100	4	yes
2 (0010)	15 (1111)	ee_e_e_	0010101	1111111	4	yes
3 (0011)	0 (0000)	__eeee_	0011110	0000000	4	yes
3 (0011)	6 (0110)	_e_e_ee	0011110	0110101	4	yes
3 (0011)	8 (1000)	e_ee__e	0011110	1000111	4	yes
3 (0011)	14 (1110)	ee_ee__	0011110	1110010	4	yes
4 (0100)	0 (0000)	_e_eee	0100111	0000000	4	yes
4 (0100)	3 (0011)	__eeee_	0100111	0011011	4	yes
4 (0100)	13 (1101)	e_eee_	0100111	1101001	4	yes
4 (0100)	14 (1110)	e_e_e_e	0100111	1110010	4	yes
5 (0101)	2 (0010)	__eee_e	0101100	0010110	4	yes
5 (0101)	4 (0100)	__eeee	0101100	0100011	4	yes
5 (0101)	9 (1001)	ee_ee_	0101100	1001010	4	yes
5 (0101)	15 (1111)	e_e_eee	0101100	1111111	4	yes
6 (0110)	9 (1001)	eeee__	0110010	1001010	4	yes
6 (0110)	10 (1010)	ee___ee	0110010	1010001	4	yes
6 (0110)	12 (1100)	e_e_ee_	0110010	1100100	4	yes
6 (0110)	15 (1111)	e_ee_e	0110010	1111111	4	yes
7 (0111)	0 (0000)	___eee_e	0111001	0000000	4	yes
7 (0111)	5 (0101)	__e_eee	0111001	0101110	4	yes

7 (0111)	11 (1011)	ee_e_e	0111001	1011100	4	yes
7 (0111)	14 (1110)	e_e_ee	0111001	1110010	4	yes
8 (1000)	1 (0001)	e_e_ee	1000110	0001101	4	yes
8 (1000)	4 (0100)	ee_e_e	1000110	0100011	4	yes
8 (1000)	10 (1010)	_e_eee	1000110	1010001	4	yes
8 (1000)	15 (1111)	_eee_e	1000110	1111111	4	yes
9 (1001)	0 (0000)	e_ee_e	1001101	0000000	4	yes
9 (1001)	3 (0011)	e_e_ee	1001101	0011011	4	yes
9 (1001)	5 (0101)	ee_ee	1001101	0101110	4	yes
9 (1001)	6 (0110)	eeee_	1001101	0110101	4	yes
10 (1010)	0 (0000)	e_e_ee	1010011	0000000	4	yes
10 (1010)	6 (0110)	ee_ee_	1010011	0110101	4	yes
10 (1010)	11 (1011)	_eeee	1010011	1011100	4	yes
10 (1010)	13 (1101)	_eee_e_	1010011	1101001	4	yes
11 (1011)	1 (0001)	e_e_e_e	1011000	0001101	4	yes
11 (1011)	2 (0010)	e_eee_	1011000	0010110	4	yes
11 (1011)	12 (1100)	_eeee_	1011000	1100100	4	yes
11 (1011)	15 (1111)	_e_eee	1011000	1111111	4	yes
12 (1100)	1 (0001)	ee_ee_	1100001	0001101	4	yes
12 (1100)	7 (0111)	e_ee_e	1100001	0111000	4	yes
12 (1100)	9 (1001)	_e_e_ee	1100001	1001010	4	yes
12 (1100)	15 (1111)	_eeee_	1100001	1111111	4	yes
13 (1101)	0 (0000)	ee_e_e	1101010	0000000	4	yes
13 (1101)	3 (0011)	eee_e	1101010	0011011	4	yes
13 (1101)	8 (1000)	_e_ee_e	1101010	1000111	4	yes
13 (1101)	11 (1011)	_ee_ee_	1101010	1011100	4	yes
14 (1110)	0 (0000)	eee_e_	1110100	0000000	4	yes
14 (1110)	5 (0101)	e_ee_e	1110100	0101110	4	yes
14 (1110)	8 (1000)	_ee_ee	1110100	1000111	4	yes
14 (1110)	13 (1101)	_eee_e	1110100	1101001	4	yes
15 (1111)	1 (0001)	eee_e	1111111	0001101	4	yes
15 (1111)	2 (0010)	ee_e_e	1111111	0010110	4	yes
15 (1111)	4 (0100)	e_eee_	1111111	0100011	4	yes
15 (1111)	7 (0111)	e_eee	1111111	0111000	4	yes
15 (1111)	9 (1001)	_ee_e_e	1111111	1001010	4	yes
15 (1111)	10 (1010)	_e_eee_	1111111	1010001	4	yes
15 (1111)	12 (1100)	_ee_ee	1111111	1100100	4	yes
Sequences with number of erasures equal to 5						
1 (0001)	6 (0110)	_eeee_	0001011	0110101	5	yes
1 (0001)	11 (1011)	e_e_eee	0001011	1011100	5	yes
1 (0001)	14 (1110)	eeee_e	0001011	1110010	5	yes
2 (0010)	5 (0101)	_eee_ee	0010101	0101110	5	yes
2 (0010)	13 (1101)	eeee_	0010101	1101001	5	yes
2 (0010)	14 (1110)	ee_eee	0010101	1110010	5	yes
3 (0011)	4 (0100)	_eeee_e	0011110	0100011	5	yes
3 (0011)	10 (1010)	e_eeee	0011110	1010001	5	yes
3 (0011)	12 (1100)	eeee_e_	0011110	1100100	5	yes
4 (0100)	7 (0111)	_eeee_	0100111	0111000	5	yes
4 (0100)	9 (1001)	ee_ee_e	0100111	1001010	5	yes
4 (0100)	10 (1010)	eee_ee_	0100111	1010001	5	yes
5 (0101)	3 (0011)	_ee_eee	0101100	0011011	5	yes
5 (0101)	8 (1000)	ee_e_ee	0101100	1000111	5	yes
5 (0101)	14 (1110)	e_eeee	0101100	1110010	5	yes
6 (0110)	8 (1000)	eee_e_e	0110010	1000111	5	yes
6 (0110)	11 (1011)	ee_eee_	0110010	1011100	5	yes
6 (0110)	13 (1101)	e_ee_ee	0110010	1101001	5	yes
7 (0111)	2 (0010)	_e_eeee	0111001	0010110	5	yes
7 (0111)	9 (1001)	eee_ee	0111001	1001010	5	yes
7 (0111)	12 (1100)	e_eee_e	0111001	1100100	5	yes
8 (1000)	3 (0011)	e_eee_e	1000110	0011011	5	yes
8 (1000)	6 (0110)	eee_ee	1000110	0110101	5	yes
8 (1000)	13 (1101)	_e_eeee	1000110	1101001	5	yes
9 (1001)	2 (0010)	e_ee_ee	1001101	0010110	5	yes
9 (1001)	4 (0100)	ee_eee_	1001101	0100011	5	yes

9 (1001)	7 (0111)	eee_e_e	1001101	0111000	5	yes
10 (1010)	1 (0001)	e_eeee_	1010011	0001101	5	yes
10 (1010)	7 (0111)	ee_e_ee	1010011	0111000	5	yes
10 (1010)	12 (1100)	_ee_eee	1010011	1100100	5	yes
11 (1011)	5 (0101)	eee_ee_	1011000	0101110	5	yes
11 (1011)	6 (0110)	ee_ee_e	1011000	0110101	5	yes
11 (1011)	8 (1000)	_eeeeee	1011000	1000111	5	yes
12 (1100)	3 (0011)	eeee_e_	1100001	0011011	5	yes
12 (1100)	5 (0101)	e_eeee	1100001	0101110	5	yes
12 (1100)	11 (1011)	_eeee_e	1100001	1011100	5	yes
13 (1101)	1 (0001)	ee_eee	1101010	0001101	5	yes
13 (1101)	2 (0010)	eeeeee_	1101010	0010110	5	yes
13 (1101)	10 (1010)	_eee_ee	1101010	1010001	5	yes
14 (1110)	1 (0001)	eeee_e	1110100	0001101	5	yes
14 (1110)	4 (0100)	e_e_eee	1110100	0100011	5	yes
14 (1110)	9 (1001)	_eeee_e	1110100	1001010	5	yes
Sequences with number of erasures equal to 6						
1 (0001)	12 (1100)	ee_eeee	0001011	1100100	6	yes
2 (0010)	9 (1001)	e_eeee	0010101	1001010	6	yes
3 (0011)	13 (1101)	eee_eee	0011110	1101001	6	yes
4 (0100)	11 (1011)	eeee_ee	0100111	1011100	6	yes
5 (0101)	10 (1010)	eeee_e	0101100	1010001	6	yes
6 (0110)	1 (0001)	_eeeeee	0110010	0001101	6	yes
7 (0111)	8 (1000)	eeeeee_	0111001	1000111	6	yes
8 (1000)	7 (0111)	eeeeee_	1000110	0111000	6	yes
9 (1001)	14 (1110)	_eeeeee	1001101	1110010	6	yes
10 (1010)	5 (0101)	eeee_e	1010011	0101110	6	yes
11 (1011)	4 (0100)	eeee_ee	1011000	0100011	6	yes
12 (1100)	2 (0010)	eee_eee	1100001	0010110	6	yes
13 (1101)	6 (0110)	e_eeee	1101010	0110101	6	yes
14 (1110)	3 (0011)	ee_eeee	1110100	0011011	6	yes
Sequences with number of erasures equal to 7						
0 (0000)	15 (1111)	eeeeeee	0000000	1111111	7	no
15 (1111)	0 (0000)	eeeeeee	1111111	0000000	7	no

**Table 38: Decoding lookup table at Node 3**

Node 1 Info. Seq.	Node 2 Info. Seq.	Erasure Pattern	Node 1 Codeword	Node 2 Codeword	Num of erased bits	Decodable
Sequences with number of erasures equal to 0						
0 (0000)	0 (0000)	_____	0000000	0000000	0	yes
15 (1111)	15 (1111)	_____	1111111	1111111	0	yes
Sequences with number of erasures equal to 1						
1 (0001)	1 (0001)	___e___	0001011	0001111	1	yes
2 (0010)	2 (0010)	____e_	0010101	0010111	1	yes
3 (0011)	7 (0111)	_e_____	0011110	0111110	1	yes
4 (0100)	4 (0100)	____e	0100111	0100110	1	yes
4 (0100)	12 (1100)	e_____	0100111	1100111	1	yes
7 (0111)	5 (0101)	__e____	0111001	0101001	1	yes
7 (0111)	6 (0110)	___e___	0111001	0110001	1	yes
8 (1000)	9 (1001)	____e__	1000110	1001110	1	yes
8 (1000)	10 (1010)	___e____	1000110	1010110	1	yes
11 (1011)	3 (0011)	e_____	1011000	0011000	1	yes
11 (1011)	11 (1011)	____e	1011000	1011001	1	yes
12 (1100)	8 (1000)	_e_____	1100001	1000001	1	yes
13 (1101)	13 (1101)	____e_	1101010	1101000	1	yes
14 (1110)	14 (1110)	___e____	1110100	1110000	1	yes
Sequences with number of erasures equal to 2						
0 (0000)	3 (0011)	__ee___	0000000	0011000	2	yes
0 (0000)	8 (1000)	e____e	0000000	1000001	2	yes

1 (0001)	5 (0101)	_e_e_e	0001011	0101001	2	yes
2 (0010)	6 (0110)	_e_e_e	0010101	0110001	2	yes
3 (0011)	1 (0001)	_e_e_e	0011110	0001111	2	yes
3 (0011)	2 (0010)	_e_e_e	0011110	0010111	2	yes
3 (0011)	3 (0011)	_ee_e	0011110	0011000	2	yes
3 (0011)	9 (1001)	e_e_e	0011110	1001110	2	yes
3 (0011)	10 (1010)	e_e_e	0011110	1010110	2	yes
4 (0100)	1 (0001)	_e_e_e	0100111	0001111	2	yes
4 (0100)	2 (0010)	_ee_e	0100111	0010111	2	yes
5 (0101)	4 (0100)	_e_e_e	0101100	0100110	2	yes
5 (0101)	5 (0101)	_ee_e	0101100	0101001	2	yes
5 (0101)	7 (0111)	_e_e_e	0101100	0111110	2	yes
5 (0101)	13 (1101)	e_e_e	0101100	1101000	2	yes
6 (0110)	4 (0100)	_e_e_e	0110010	0100110	2	yes
6 (0110)	6 (0110)	_ee_e	0110010	0110001	2	yes
6 (0110)	7 (0111)	_ee_e	0110010	0111110	2	yes
6 (0110)	14 (1110)	e_e_e	0110010	1110000	2	yes
7 (0111)	3 (0011)	_e_e_e	0111001	0011000	2	yes
7 (0111)	11 (1011)	ee_e	0111001	1011001	2	yes
8 (1000)	4 (0100)	ee_e	1000110	0100110	2	yes
8 (1000)	12 (1100)	_e_e_e	1000110	1100111	2	yes
9 (1001)	1 (0001)	e_e_e	1001101	0001111	2	yes
9 (1001)	8 (1000)	_ee_e	1001101	1000001	2	yes
9 (1001)	9 (1001)	_ee_e	1001101	1001110	2	yes
9 (1001)	11 (1011)	_e_e_e	1001101	1011001	2	yes
10 (1010)	2 (0010)	e_e_e	1010011	0010111	2	yes
10 (1010)	8 (1000)	_e_e_e	1010011	1000001	2	yes
10 (1010)	10 (1010)	_e_e_e	1010011	1010110	2	yes
10 (1010)	11 (1011)	_e_e_e	1010011	1011001	2	yes
11 (1011)	13 (1101)	_ee_e	1011000	1101000	2	yes
11 (1011)	14 (1110)	_e_e_e	1011000	1110000	2	yes
12 (1100)	5 (0101)	e_e_e	1100001	0101001	2	yes
12 (1100)	6 (0110)	e_e_e	1100001	0110001	2	yes
12 (1100)	12 (1100)	_ee_e	1100001	1100111	2	yes
12 (1100)	13 (1101)	_e_e_e	1100001	1101000	2	yes
12 (1100)	14 (1110)	_e_e_e	1100001	1110000	2	yes
13 (1101)	9 (1001)	_e_e_e	1101010	1001110	2	yes
14 (1110)	10 (1010)	_e_e_e	1110100	1010110	2	yes
15 (1111)	7 (0111)	e_e_e	1111111	0111110	2	yes
15 (1111)	12 (1100)	_ee_e	1111111	1100111	2	yes
Sequences with number of erasures equal to 3						
0 (0000)	4 (0100)	_e_ee	0000000	0100110	3	yes
0 (0000)	5 (0101)	_e_e_e	0000000	0101001	3	yes
0 (0000)	6 (0110)	_ee_e	0000000	0110001	3	yes
0 (0000)	13 (1101)	ee_e	0000000	1101000	3	yes
0 (0000)	14 (1110)	eee	0000000	1110000	3	yes
1 (0001)	0 (0000)	_ee_e	0001011	0000000	3	yes
1 (0001)	2 (0010)	_eee	0001011	0010111	3	yes
1 (0001)	3 (0011)	_e_ee	0001011	0011000	3	yes
1 (0001)	8 (1000)	e_e_e	0001011	1000001	3	yes
1 (0001)	9 (1001)	e_e_e	0001011	1001110	3	yes
1 (0001)	11 (1011)	e_e_e	0001011	1011001	3	yes
2 (0010)	0 (0000)	_e_e_e	0010101	0000000	3	yes
2 (0010)	1 (0001)	_ee_e	0010101	0001111	3	yes
2 (0010)	3 (0011)	_ee_e	0010101	0011000	3	yes
2 (0010)	8 (1000)	e_e_e	0010101	1000001	3	yes
2 (0010)	10 (1010)	e_ee	0010101	1010110	3	yes
2 (0010)	11 (1011)	e_ee	0010101	1011001	3	yes
3 (0011)	4 (0100)	_eee	0011110	0100110	3	yes
3 (0011)	15 (1111)	ee_e	0011110	1111111	3	yes
4 (0100)	5 (0101)	_eee	0100111	0101001	3	yes
4 (0100)	6 (0110)	_e_ee	0100111	0110001	3	yes
4 (0100)	7 (0111)	_ee_e	0100111	0111110	3	yes
4 (0100)	15 (1111)	e_ee	0100111	1111111	3	yes

5 (0101)	0 (0000)	_e_ee_	0101100	0000000	3	yes
5 (0101)	1 (0001)	_e__ee	0101100	0001111	3	yes
5 (0101)	3 (0011)	_ee_e_	0101100	0011000	3	yes
5 (0101)	9 (1001)	ee__e_	0101100	1001110	3	yes
6 (0110)	0 (0000)	_ee_e_	0110010	0000000	3	yes
6 (0110)	2 (0010)	_e__e_e	0110010	0010111	3	yes
6 (0110)	3 (0011)	_e_e_e_	0110010	0011000	3	yes
6 (0110)	10 (1010)	ee__e_	0110010	1010110	3	yes
7 (0111)	7 (0111)	___eee	0111001	0111110	3	yes
7 (0111)	13 (1101)	e_e_e_e	0111001	1101000	3	yes
7 (0111)	14 (1110)	e__e_e	0111001	1110000	3	yes
7 (0111)	15 (1111)	e___ee_	0111001	1111111	3	yes
8 (1000)	0 (0000)	e__ee_	1000110	0000000	3	yes
8 (1000)	1 (0001)	e__e_e	1000110	0001111	3	yes
8 (1000)	2 (0010)	e_e_e_e	1000110	0010111	3	yes
8 (1000)	8 (1000)	___eee	1000110	1000001	3	yes
9 (1001)	5 (0101)	ee__e_	1001101	0101001	3	yes
9 (1001)	12 (1100)	_e_e_e_	1001101	1100111	3	yes
9 (1001)	13 (1101)	_e__e_e	1001101	1101000	3	yes
9 (1001)	15 (1111)	_ee_e_	1001101	1111111	3	yes
10 (1010)	6 (0110)	ee__e_	1010011	0110001	3	yes
10 (1010)	12 (1100)	_ee_e_	1010011	1100111	3	yes
10 (1010)	14 (1110)	_e__ee	1010011	1110000	3	yes
10 (1010)	15 (1111)	_e_ee_	1010011	1111111	3	yes
11 (1011)	0 (0000)	e_ee__	1011000	0000000	3	yes
11 (1011)	8 (1000)	__ee_e	1011000	1000001	3	yes
11 (1011)	9 (1001)	_e_ee_	1011000	1001110	3	yes
11 (1011)	10 (1010)	___eee_	1011000	1010110	3	yes
12 (1100)	0 (0000)	ee___e	1100001	0000000	3	yes
12 (1100)	11 (1011)	_eee__	1100001	1011001	3	yes
13 (1101)	4 (0100)	e__ee_	1101010	0100110	3	yes
13 (1101)	5 (0101)	e___ee	1101010	0101001	3	yes
13 (1101)	7 (0111)	e_e_e_	1101010	0111110	3	yes
13 (1101)	12 (1100)	__ee_e	1101010	1100111	3	yes
13 (1101)	14 (1110)	__ee_e_	1101010	1110000	3	yes
13 (1101)	15 (1111)	__e_e_e	1101010	1111111	3	yes
14 (1110)	4 (0100)	e_e_e_e	1110100	0100110	3	yes
14 (1110)	6 (0110)	e__e_e	1110100	0110001	3	yes
14 (1110)	7 (0111)	e_e_e_e	1110100	0111110	3	yes
14 (1110)	12 (1100)	__e_ee	1110100	1100111	3	yes
14 (1110)	13 (1101)	___eee	1110100	1101000	3	yes
14 (1110)	15 (1111)	___e_ee	1110100	1111111	3	yes
15 (1111)	1 (0001)	eee__	1111111	0001111	3	yes
15 (1111)	2 (0010)	ee_e__	1111111	0010111	3	yes
15 (1111)	9 (1001)	_ee___e	1111111	1001110	3	yes
15 (1111)	10 (1010)	_e_e_e_e	1111111	1010110	3	yes
15 (1111)	11 (1011)	_e__ee_	1111111	1011001	3	yes
Sequences with number of erasures equal to 4						
0 (0000)	1 (0001)	___eeee	0000000	0001111	4	yes
0 (0000)	2 (0010)	__e_eee	0000000	0010111	4	yes
0 (0000)	9 (1001)	e_eee_	0000000	1001110	4	yes
0 (0000)	10 (1010)	e_e_ee_	0000000	1010110	4	yes
0 (0000)	11 (1011)	e_ee_e_e	0000000	1011001	4	yes
1 (0001)	4 (0100)	_e_ee_e	0001011	0100110	4	yes
1 (0001)	6 (0110)	__eee_e	0001011	0110001	4	yes
1 (0001)	7 (0111)	_ee_e_e	0001011	0111110	4	yes
1 (0001)	12 (1100)	ee_ee_	0001011	1100111	4	yes
1 (0001)	13 (1101)	ee___ee	0001011	1101000	4	yes
1 (0001)	15 (1111)	eee_e_	0001011	1111111	4	yes
2 (0010)	4 (0100)	_ee_ee	0010101	0100110	4	yes
2 (0010)	5 (0101)	__eeee	0010101	0101001	4	yes
2 (0010)	7 (0111)	_e_e_ee	0010101	0111110	4	yes
2 (0010)	12 (1100)	eee_e_	0010101	1100111	4	yes
2 (0010)	14 (1110)	ee__e_e	0010101	1110000	4	yes

2 (0010)	15 (1111)	ee_e_e_	0010101	1111111	4	yes
3 (0011)	0 (0000)	__eeee_	0011110	0000000	4	yes
3 (0011)	11 (1011)	e___eee	0011110	1011001	4	yes
4 (0100)	0 (0000)	_e___eee	0100111	0000000	4	yes
4 (0100)	8 (1000)	ee__ee_	0100111	1000001	4	yes
4 (0100)	9 (1001)	ee_e__e	0100111	1001110	4	yes
4 (0100)	10 (1010)	eee_e_	0100111	1010110	4	yes
5 (0101)	6 (0110)	__eee_e	0101100	0110001	4	yes
5 (0101)	12 (1100)	e__e__ee	0101100	1100111	4	yes
5 (0101)	14 (1110)	e_eee__	0101100	1110000	4	yes
5 (0101)	15 (1111)	e_e__ee	0101100	1111111	4	yes
6 (0110)	5 (0101)	__ee__ee	0110010	0101001	4	yes
6 (0110)	12 (1100)	e_e_e_e	0110010	1100111	4	yes
6 (0110)	13 (1101)	e_ee_e_	0110010	1101000	4	yes
6 (0110)	15 (1111)	e_ee_e	0110010	1111111	4	yes
7 (0111)	0 (0000)	__eee_e	0111001	0000000	4	yes
7 (0111)	1 (0001)	_ee__ee	0111001	0001111	4	yes
7 (0111)	2 (0010)	_e_eee_	0111001	0010111	4	yes
7 (0111)	8 (1000)	eeee__	0111001	1000001	4	yes
8 (1000)	7 (0111)	eeee__	1000110	0111110	4	yes
8 (1000)	13 (1101)	_e_eee_	1000110	1101000	4	yes
8 (1000)	14 (1110)	_ee__ee	1000110	1110000	4	yes
8 (1000)	15 (1111)	_eee_e	1000110	1111111	4	yes
9 (1001)	0 (0000)	e_ee_e	1001101	0000000	4	yes
9 (1001)	2 (0010)	e_ee_e	1001101	0010111	4	yes
9 (1001)	3 (0011)	e_e_e_e	1001101	0011000	4	yes
9 (1001)	10 (1010)	__ee__ee	1001101	1010110	4	yes
10 (1010)	0 (0000)	e_e__ee	1010011	0000000	4	yes
10 (1010)	1 (0001)	e_eee__	1010011	0001111	4	yes
10 (1010)	3 (0011)	e_e__ee	1010011	0011000	4	yes
10 (1010)	9 (1001)	__eee_e	1010011	1001110	4	yes
11 (1011)	5 (0101)	eee__e	1011000	0101001	4	yes
11 (1011)	6 (0110)	ee_e_e	1011000	0110001	4	yes
11 (1011)	7 (0111)	ee__ee	1011000	0111110	4	yes
11 (1011)	15 (1111)	_e___eee	1011000	1111111	4	yes
12 (1100)	4 (0100)	e_eee	1100001	0100110	4	yes
12 (1100)	15 (1111)	__eeee	1100001	1111111	4	yes
13 (1101)	0 (0000)	ee_e_e	1101010	0000000	4	yes
13 (1101)	1 (0001)	ee_e_e	1101010	0001111	4	yes
13 (1101)	3 (0011)	eee_e_	1101010	0011000	4	yes
13 (1101)	8 (1000)	_e_e__ee	1101010	1000001	4	yes
13 (1101)	10 (1010)	__eeee	1101010	1010110	4	yes
13 (1101)	11 (1011)	_ee__ee	1101010	1011001	4	yes
14 (1110)	0 (0000)	eee_e_	1110100	0000000	4	yes
14 (1110)	2 (0010)	ee__ee	1110100	0010111	4	yes
14 (1110)	3 (0011)	ee__ee	1110100	0011000	4	yes
14 (1110)	8 (1000)	_ee_e_e	1110100	1000001	4	yes
14 (1110)	9 (1001)	_eee_e_	1110100	1001110	4	yes
14 (1110)	11 (1011)	_e_ee_e	1110100	1011001	4	yes
15 (1111)	4 (0100)	e_ee_e	1111111	0100110	4	yes
15 (1111)	5 (0101)	e_e__ee	1111111	0101001	4	yes
15 (1111)	6 (0110)	e_eee	1111111	0110001	4	yes
15 (1111)	13 (1101)	_e_eee	1111111	1101000	4	yes
15 (1111)	14 (1110)	__eeee	1111111	1110000	4	yes
Sequences with number of erasures equal to 5						
0 (0000)	7 (0111)	_eeee_	0000000	0111110	5	yes
0 (0000)	12 (1100)	ee_eee	0000000	1100111	5	yes
1 (0001)	10 (1010)	e_eee_e	0001011	1010110	5	yes
2 (0010)	9 (1001)	e_ee__ee	0010101	1001110	5	yes
3 (0011)	5 (0101)	__ee_eee	0011110	0101001	5	yes
3 (0011)	6 (0110)	_e_eeee	0011110	0110001	5	yes
3 (0011)	12 (1100)	eeee_e	0011110	1100111	5	yes
3 (0011)	13 (1101)	eee__ee	0011110	1101000	5	yes
3 (0011)	14 (1110)	ee_eee_	0011110	1110000	5	yes

4 (0100)	13 (1101)	e_eeee	0100111	1101000	5	yes
4 (0100)	14 (1110)	e_e_eee	0100111	1110000	5	yes
5 (0101)	2 (0010)	_eee_ee	0101100	0010111	5	yes
5 (0101)	8 (1000)	ee_ee_e	0101100	1000001	5	yes
5 (0101)	10 (1010)	eeee_e_	0101100	1010110	5	yes
5 (0101)	11 (1011)	eee_e_e	0101100	1011001	5	yes
6 (0110)	1 (0001)	_eeee_e	0110010	0001111	5	yes
6 (0110)	8 (1000)	eee_ee	0110010	1000001	5	yes
6 (0110)	9 (1001)	eeee_	0110010	1001110	5	yes
6 (0110)	11 (1011)	ee_e_ee	0110010	1011001	5	yes
7 (0111)	4 (0100)	_eeeee	0111001	0100110	5	yes
7 (0111)	12 (1100)	e_eeee_	0111001	1100111	5	yes
8 (1000)	3 (0011)	e_eeee_	1000110	0011000	5	yes
8 (1000)	11 (1011)	_eeeee	1000110	1011001	5	yes
9 (1001)	4 (0100)	ee_e_ee	1001101	0100110	5	yes
9 (1001)	6 (0110)	eeee_	1001101	0110001	5	yes
9 (1001)	7 (0111)	eee_ee	1001101	0111110	5	yes
9 (1001)	14 (1110)	_eeee_e	1001101	1110000	5	yes
10 (1010)	4 (0100)	eee_e_e	1010011	0100110	5	yes
10 (1010)	5 (0101)	eeee_e_	1010011	0101001	5	yes
10 (1010)	7 (0111)	ee_ee_e	1010011	0111110	5	yes
10 (1010)	13 (1101)	_eee_ee	1010011	1101000	5	yes
11 (1011)	1 (0001)	e_e_eee	1011000	0001111	5	yes
11 (1011)	2 (0010)	e_eeee	1011000	0010111	5	yes
12 (1100)	1 (0001)	ee_eee_	1100001	0001111	5	yes
12 (1100)	2 (0010)	eee_ee_	1100001	0010111	5	yes
12 (1100)	3 (0011)	eeee_e	1100001	0011000	5	yes
12 (1100)	9 (1001)	_e_eeee	1100001	1001110	5	yes
12 (1100)	10 (1010)	_ee_eee	1100001	1010110	5	yes
13 (1101)	6 (0110)	e_ee_ee	1101010	0110001	5	yes
14 (1110)	5 (0101)	e_eee_e	1110100	0101001	5	yes
15 (1111)	3 (0011)	ee_eee	1111111	0011000	5	yes
15 (1111)	8 (1000)	_eeeee_	1111111	1000001	5	yes
Sequences with number of erasures equal to 6						
1 (0001)	14 (1110)	eeee_ee	0001011	1110000	6	yes
2 (0010)	13 (1101)	eeee_e	0010101	1101000	6	yes
3 (0011)	8 (1000)	e_eeee	0011110	1000001	6	yes
4 (0100)	3 (0011)	_eeeeee	0100111	0011000	6	yes
4 (0100)	11 (1011)	eeeeee_	0100111	1011001	6	yes
7 (0111)	9 (1001)	eee_eee	0111001	1001110	6	yes
7 (0111)	10 (1010)	ee_eeee	0111001	1010110	6	yes
8 (1000)	5 (0101)	ee_eeee	1000110	0101001	6	yes
8 (1000)	6 (0110)	eee_eee	1000110	0110001	6	yes
11 (1011)	4 (0100)	eeeeee_	1011000	0100110	6	yes
11 (1011)	12 (1100)	_eeeee	1011000	1100111	6	yes
12 (1100)	7 (0111)	e_eeee	1100001	0111110	6	yes
13 (1101)	2 (0010)	eeee_e	1101010	0010111	6	yes
14 (1110)	1 (0001)	eeee_ee	1110100	0001111	6	yes
Sequences with number of erasures equal to 7						
0 (0000)	15 (1111)	eeeeeee	0000000	1111111	7	no
15 (1111)	0 (0000)	eeeeeee	1111111	0000000	7	no

As can be seen in the lookup tables above, a received erasure pattern is decodable if and only if there is a one-to-one relationship between that received erasure sequence and the possible sent codewords from two senders. It is interesting to note that the size of the lookup table increases exponentially with the increase of the size of the packet. Thus, it is not the preferred method of decoding.



### 3.3.4. Theoretical Limit of Performance of Algebraic Linear Block Codes in a Noiseless Channel (Erasures-Only)

As seen from the example above, the used block code, without taking into consideration noise in the channel, can separate the combined signal successfully given that the number of erasures in the received codeword is less or equal to 6 erasures. Since the number of parity bits, and hence the number of orthogonal linear relationships, is equal to six at each receiving node, the code can solve for up to six variables (six erasures).

$$\begin{aligned}
 & \textit{Erasure geussing capability } (g) \\
 & \leq \textit{number of parity bits } (k) \\
 & \times (\textit{number of communicating nodes} - 1)
 \end{aligned} \tag{3.175}$$

The previous equation holds true if all the parity bit linear relationships are orthogonal. For the (7,4) hamming code and three communicating nodes seen in the example above, the equation reduces to:

$$g \leq 3 \times (3 - 1) = 6 \textit{ bits/codeword} \tag{3.176}$$

This leads us to examining the worst-case scenario where 7 bits are erased at a recipient. Decoding is not possible at the receiver in this case. At the third node (Node 3), for example, this occurs when  $a_i = b_i + 1$ . The probability of such event is equal to:

$$P(a_i = b_i + 1) = P((a_i = 1 \textit{ and } b_i = 0) \textit{ or } (a_i = 0 \textit{ and } b_i = 1)) \tag{3.177}$$

By looking at the code-books for Node 1 and Node 2. We can conclude that this is possible when:

$$\begin{aligned}
 & (a = (0000 \ 000) \textit{ and } b = (1111 \ 111)) \\
 & \textit{ or } (a = (1111 \ 111) \textit{ and } b = (0000 \ 000))
 \end{aligned} \tag{3.178}$$

The probability of these combinations is equal to:

$$\begin{aligned}
 &P(\text{erasure of all bits at Node 3}) \\
 &= P(a = 0000\ 000)P(b = 1111\ 111) \\
 &+ P(a = 1111\ 111)P(b = 0000\ 000)
 \end{aligned} \tag{3.179}$$

$$P(\text{erasure of all bits at Node 3}) = \frac{1}{16} \times \frac{1}{16} + \frac{1}{16} \times \frac{1}{16} = \frac{1}{128} \tag{3.180}$$

With that in mind, we can build the decoder in a smart way so that it detects if all the bits are erased in a code-word, and decodes the messages of  $a$  and  $b$  as all ones or all zeros. This will guarantee that one of the decoded messages is good. Consequently, the probability of erasures will be halved.

$$P(\text{bit erasure at Node 3}) = \frac{1}{2} \times \frac{1}{128} = \frac{1}{256} = 0.0039 \tag{3.181}$$

Now, let's derive the theoretical bound for minimum packet error rate when using  $(n, k)$  block codes. In this case, there are  $(n - k)$  parity bits, and thereby,  $(n - k)$  linear relationships per encoder, and  $2 \times (n - k)$  collaborative linear relationships at the recipient. If we assume that all of these are linearly independent/orthogonal, then, the collaborative algebraic code may decode  $(n - k)$  erasures in a combined codeword of length  $n$ . Hence, the packet error probability is:

$$\begin{aligned}
 &P(\text{received combined signal is not decodable}) \\
 &= P(\text{number of erasures} > (n - k)) = \frac{\sum_{e=n-k+1}^n \binom{n}{e}}{2^n}
 \end{aligned} \tag{3.182}$$

Where  $2^n$  is the number of possible erasure patterns, and  $e$  is the number of erasures in a received pattern. From (3.174), Another metric we can look at, is the effective bit rate of the overall system, assuming BPSK modulation, and 1 bit/channel use, then the maximum effective bit rate of a Two-Sender Y-Channel-Network when using algebraic linear block codes is:

*Effective Throughput (bits/s)<sub>Two-Sender Y-Channel</sub>*

$$\leq \frac{4}{2} \times \frac{k}{n} \times \left( 1 - \frac{\sum_{e=n-k+1}^n \binom{n}{e}}{2^n} \right) \quad (3.183)$$

And the maximum effective throughput in a Three-Sender Y-Channel-Relay, when every terminal is multicasting to the other two, is:

*Effective Throughput (bits/s)<sub>Three-Sender Y-Channel-Relay</sub>*

$$\leq \frac{6}{2} \times \frac{k}{n} \times \left( 1 - \frac{\sum_{e=n-k+1}^n \binom{n}{e}}{2^n} \right) \quad (3.184)$$

The equations above were modeled using a computer program for an information sequence of length 200 ( $k = 200$ ), and different coding rate per sender. Figure 60 shows the results. It can be seen that the maximum effective theoretical rate 2 bits/sec in the Three-Sender Y-Channel-Relay problem. That is better than the CNC solution discussed in the previous chapter. In addition, we can achieve zero packet error rate ( $PER < 10^{-5}$ ) when using rate 2/3 linear block code at each receiver or lower. In the next section, a few linear block codes with different rates are simulated. It is worth noting that getting good performance from a relatively high rate algebraic linear block code solution is not an easy task. The code should be constructed in such a way that pertain one-to-one mapping between each two senders' codewords, and their erasure pattern.

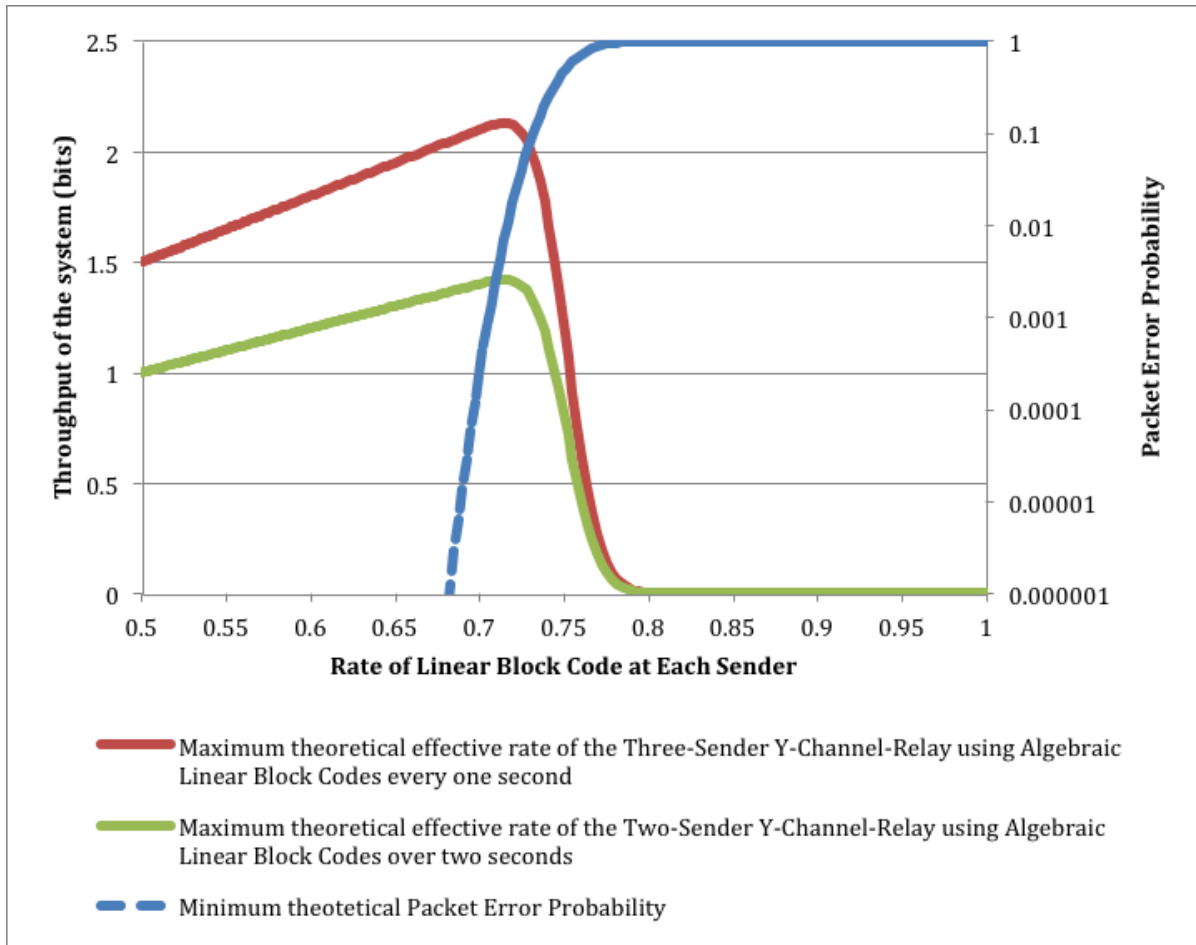


Figure 60: Theoretical performance limits of an algebraic linear block codes solution with (n=200)

Table 39 shows theoretical limits for different information bit sequence lengths, and code rates. In the next section, a few of algebraic linear block codes are simulated. The results of the simulation reflect the data represented by the model below.

Table 39: Theoretical limits for different information bit sequence lengths, and different algebraic linear block code rate

Information Bit Sequence Length (k)	Number of Parity Bits (n-k)	Rate of Code (k/n)	Number of possible erasure patterns	Minimum theoretical number of unsolvable erasure patterns	Minimum theoretical Packet Error Probability	Maximum theoretical effective rate of the Three-Sender Y-Channel-Relay using Algebraic Linear Block Codes every one second
20	14	0.588235	1.72E+10	3.31E+05	1.93E-05	1.76467
12	11	0.521739	8.39E+06	1.00E+00	1.19E-07	1.56522
13	12	0.52	3.36E+07	1.00E+00	2.98E-08	1.56
20	19	0.512821	5.50E+11	1.00E+00	1.82E-12	1.53846
150	70	0.681818	1.69E+66	2.95E+61	1.75E-05	2.05E+00
200	91	0.68728	3.98E+8	2.58E+82	6.48E-06	2.06184

### 3.3.5. Simulation of the (7,4) Algebraic Linear Block Code Example

The linear block codes based collaborative scheme described in section 3.3.2 was simulated using a decode-and-forward approach. Three equi-probable binary sources generate an endless stream of bits, and the following linear block code encoders were used for each source:

After encoding, the bits are passed into a BPSK modulator with levels  $\{-\sqrt{E_b}, +\sqrt{E_b}\}$ . An AWGN channel was used for the simulation, and we assume that none of the nodes receive any partial data from the other two nodes' transmissions. The initial energy level is chosen to give an SINR of 0 dB. Once a predetermined confidence value for  $P_b$  is earned from the simulating at that SINR level,  $E_b$  is changed to obtain a higher SINR, and the process is repeated to get the  $P_b$  curve.

For each SINR level, the following is done to obtain the  $P_b$ :

- Produce a continuous stream of bits from the three binary sources.
- Encode the bits and modulate them.
- Combine the relative output from each node, and add AWGN to it based on the chosen  $E_b$  and  $N_0$  to get the desired SINR. This simulates mixing the signals in the air at the relay.
- The received signal is then decoded in the received node.

Here is the curve for the  $P_b$  when simulating the collaborative linear block based solution at node 3 when compared to the uncoded solution:

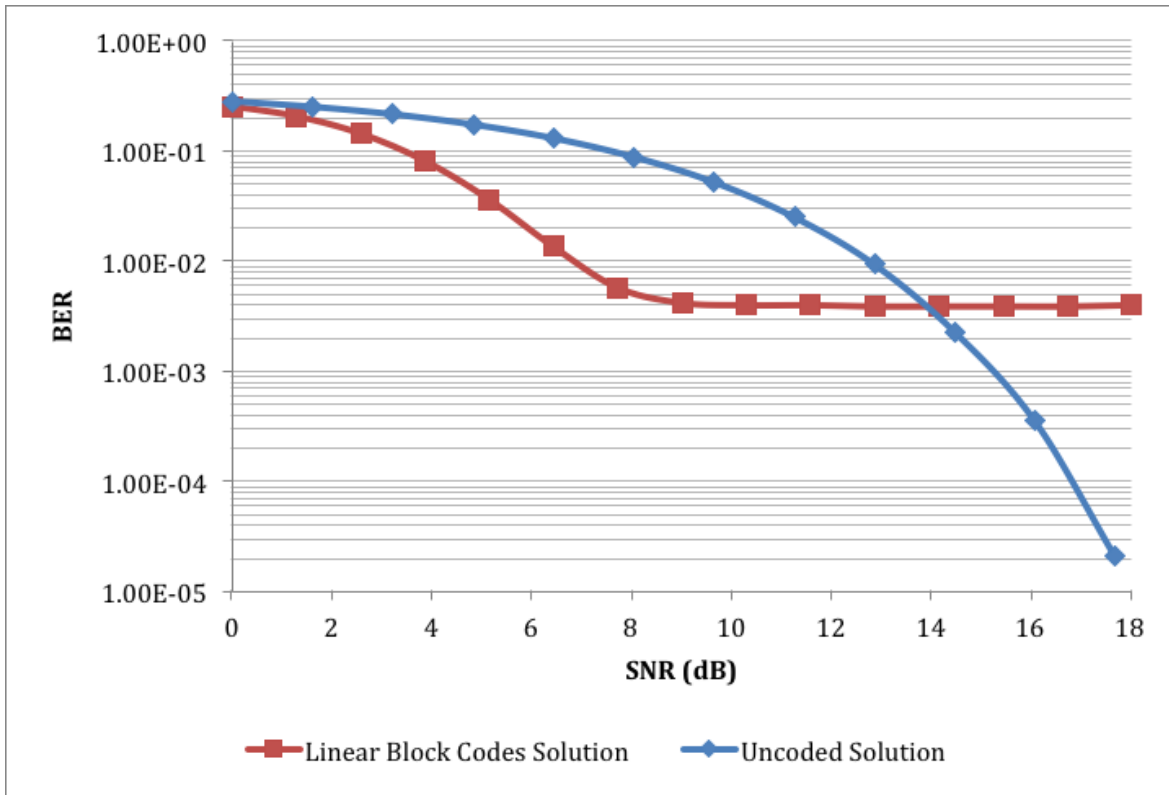


Figure 61: The simulated BER curve of the (7,4) Hamming code solution

The following can be noted from the curves above; the linear block codes solution performs way better than the uncoded method. The enhancement reaches a maximum 7 dBs when using (7, 4) Hamming Codes. The error floor is identical to the theoretical limit calculate in the previous section. The error floor can be lowered significantly by increasing the length of the used codes. One way to increase the length of the code is to increase the number of parity bits and info bits either by selecting a longer block code or BCH code as seen in Figure 60 and Table 39, or by cascading the used code with another code (LDPC code for example).

### 3.3.6. Effect of Increasing Rate of the Code

As demonstrated in the previous section, it is possible to construct linearly independent block codes at each node to collaboratively decode at the destination in a Y-Channel-Relay with ANC problem. Figure 62 shows simulation results showing different code rate. We can see that increasing the rate of the code at each sender, lowers the error floor significantly.







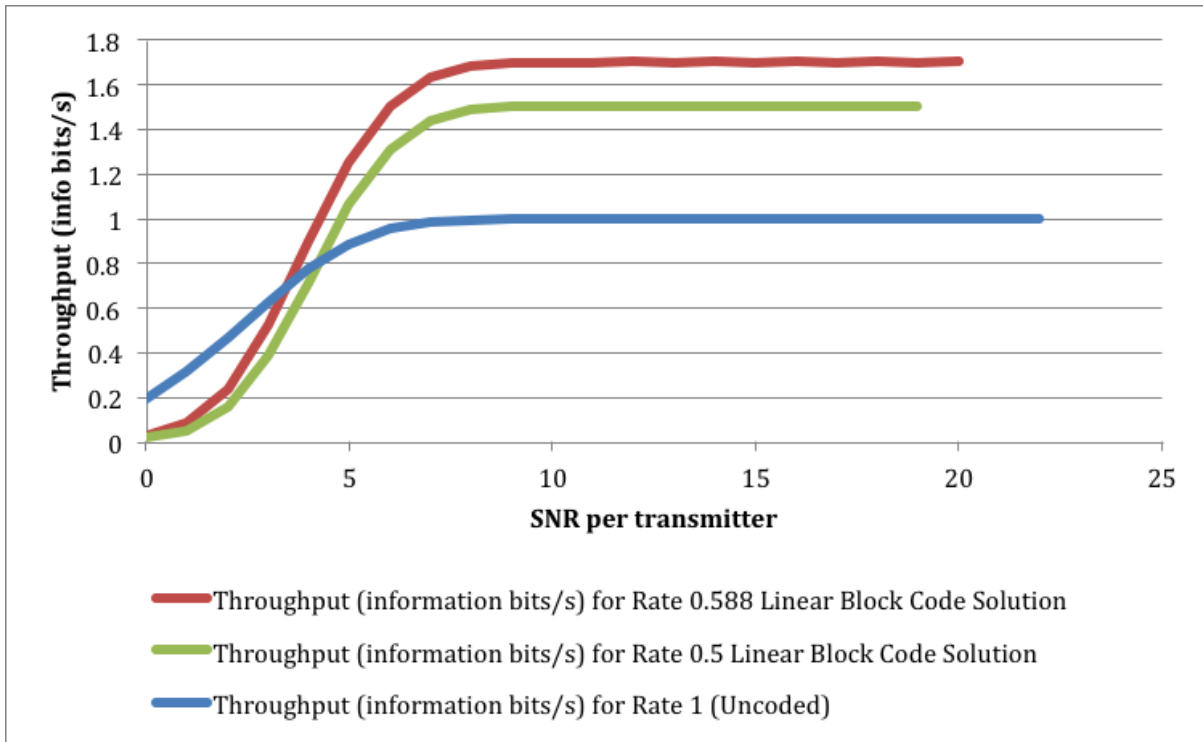


Figure 63: SNR versus Information Bit Throughput in Y-Channel-Relay Scenario

As expected, we can notice in the SNR versus information bit throughput results that in the low SNR region, uncoded performance is better than the collaborative linear block code solution. This is due to the fact that in that region, BER related to noise in the channel is dominant. However as noise subsidizes in the mid to high SNR region, we can see that the solution presented in this thesis is far more superior than any point to point un-collaborative system since our system is good at guessing erasures.

It is interesting to note that we can construct longer block codes simply by duplicating the matrix above to create multiples of 34 and 14. The same generator matrix, and hence, the same decoding table can be used to construct a (34, 14) code, a (68, 28) code, a (340, 140), a (3400, 1400) code...etc. In other words, increasing the block size of the code in this case won't increase the complexity of the decoder.

### 3.4. Comparison of The Different Coding Techniques

The following diagram compares the maximum throughput we get from the existing solutions that we explored as well as the solutions that we have proposed as part of this research. Maximum throughput can only be achieved when each node in a Y-Channel-Relay scenario wants to broadcast its packet to the other two.

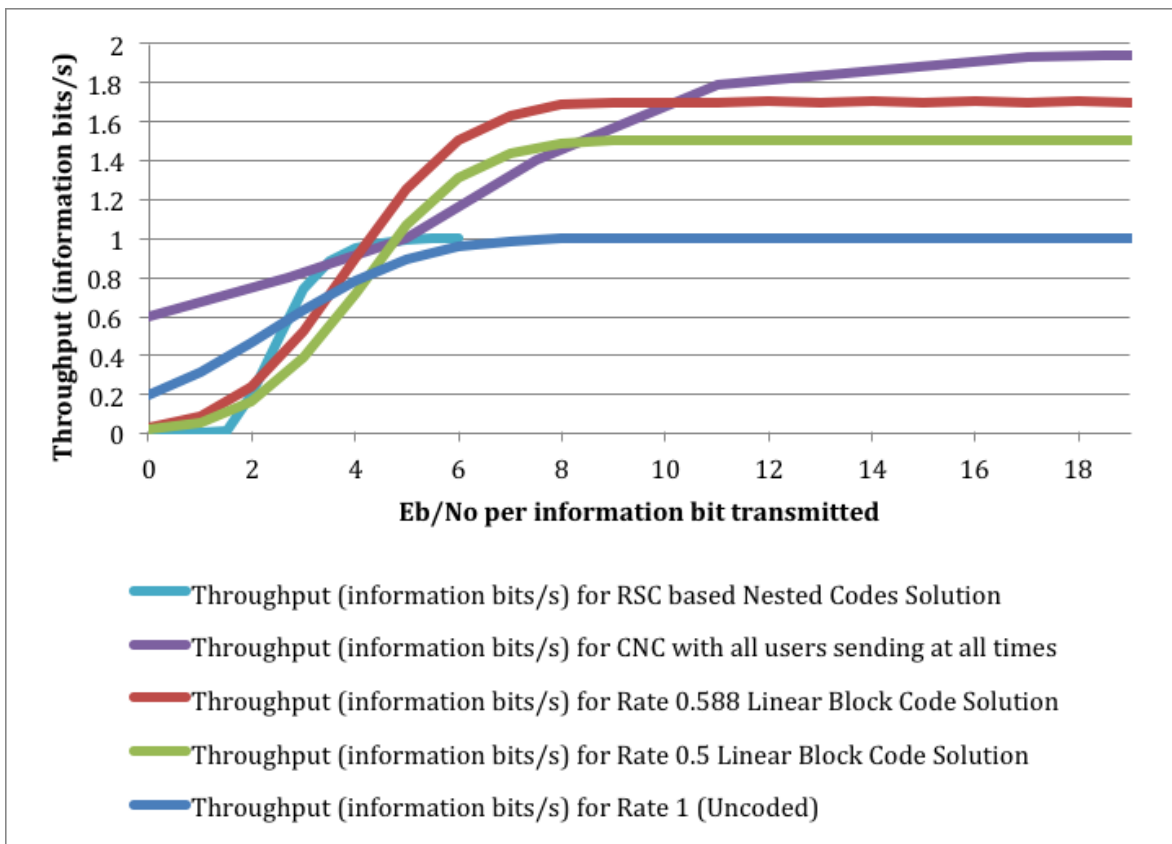


Figure 64: System throughput performance proposed solutions and existing solutions

We can see that in the high SNR (above 12 dB), the CNC solution (proposed by B. W. Khoueiry, H. Khoshnevis, and M. R. Soleymani) has better system throughput than any of the proposed solutions. However, the novel method that they have developed requires the users to send at different rates (a maximum of two users send at rate one at any time). Thus, making the system more complicated, especially in Wireless Mesh Network deployments. In addition, the practical system that they have proposed relies on the assumption that all users have data to send at all times, which is not always the case in a Wireless Mesh Network topology with bursty traffic.

Let's demonstrate this degradation with an example; if only two users want to be sent to one user for example (Two-Sender Scenario), the throughput of the system degrades. The reason for this is because in the first time slot, the two users send at rate 1 to the relay. In the second time slot, the relay broadcasts a mixed signal. Upon reception of this signal, the third user (receiver) cannot decode the messages, and has to wait for an additional transmission of at least rate 0.5, which he would receive in two transmission slots. I.e. four time slots would be required.

However, when using the linear block codes solution proposed in this thesis, users always send using codes of equal rate, and hence, making this scenario suitable for any centralized/decentralized wireless systems. Also, the throughput gains of CNC are minimal when compared with linear block codes solution of rate 0.588. The generality, flexibility and simplicity we gain from using algebraic linear block codes outweighs the benefits of the slight improvement of system throughput. Moreover, the linear block codes solution requires only two time slots to complete communication for *any* transmission scenario (Two-Sender and Three-Sender). In most real life scenarios, users send data in bursts. The linear block codes solution outperforms the CNC solution in these scenarios.

In the SNR region between 4 and 8 dB, the linear block codes based solution outperforms the CNC solution. In low SNR regions, CNC is superior. Last but not least, the theoretical limits derived for the algebraic linear block codes solution show that there is still room to use codes with rate  $2/3$ , and thereby, achieve throughput higher than CNC even in streaming multicast applications, where all terminals have data to be sent at all times.

## Chapter 4: Applications to Modern Wireless Networks

### 4.1. LTE Multimedia Broadcast/Multicast Service (eMBMS)

LTE Multimedia Broadcast/Multicast Service is a specification for broadcast services over LTE. [33] It supports audio and video streams, and was first deployed by Verizon in 2013. Currently, it is used to broadcast live sporting events from within sports venues. Any number of cells that transmit on the same frequency band can be chosen to transmit a common eMBMS service. With eMBMS, the provider dedicates a fixed portion of its carrier for broadcasting purposes, thus, affecting the capacity of data services in the cell. From a UE perspective, the eMBMS signal from multiple cells is combined, and therefore, the SINR for the eMBMS portion of the carrier in overlapping cell coverage is relatively higher than the SINR for the data portion of that carrier. The bandwidth requirement here depends on the number of services/channels (video streams), and the quality of video (HD/SD).

Let's assume that we have an LTE deployment based on a 15MHz carrier using two cells, and a UE positioned in an overlapping coverage area as seen in Figure 65. Having relatively close cells broadcasting on the same frequency is common in high capacity urban deployments such as in a stadium since the provider would sacrifice a bit of data-user experience in favor of increasing the number of connected users. Let's also assume that the provider has dedicated 5MHz of its carrier for an eMBMS service. In this case, the UE will be receiving synchronized eMBMS signals from both cells while it will only be connected to the data service of only one of those cells. The neighbor cell data portion of the carrier is considered interference to the data portion of the serving cell.

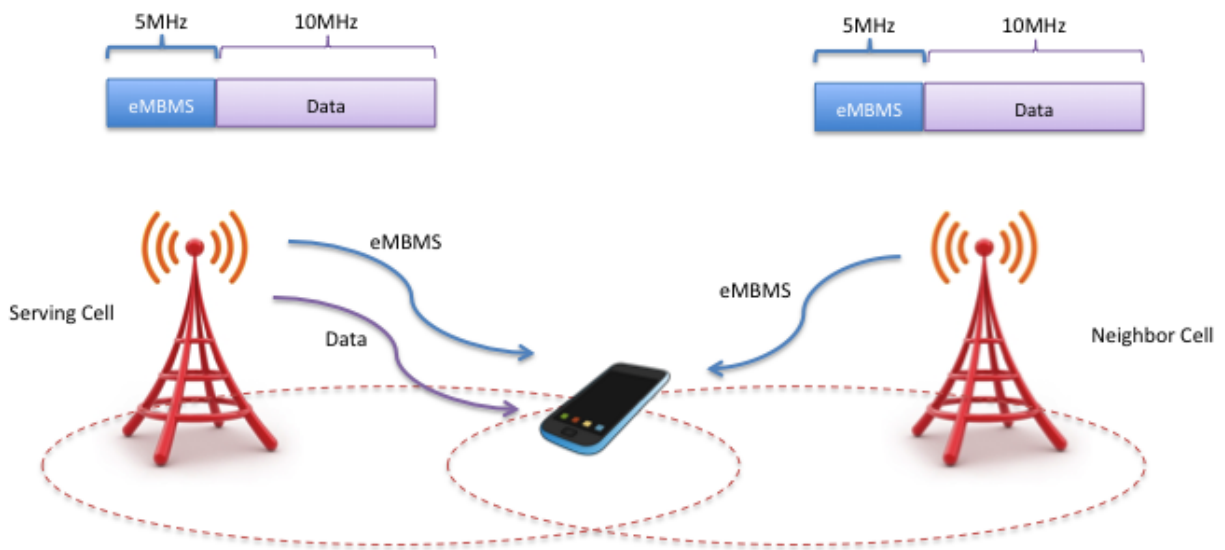


Figure 65: eMBMS deployment example

It is interesting to note here that the UE receives a combined synchronized signal of both cells for eMBMS. This is very similar to a relay receiving signals from two users simultaneously when using the analog network techniques we discussed earlier in this thesis, except in the case of eMBMS, the transmitted signal from both users is exactly the same. This led us to think of a modification to the eMBMS system that uses the algebraic linear block code introduced in this thesis. In this proposal, the serving cell transmits half of the MBMS bits while the neighboring cell transmits the other half. Each cell will use the rate 0.588 algebraic linear block code scheme we introduced earlier. The result is a theoretical reduction of 15% in bandwidth requirement for eMBMS. This bandwidth can go back to support data traffic in the cell, and therefore, enhancing cell data throughput with eMBMS service turned on. Figure 66 describes this solution by showing the carrier content of each cell. Upon reception of the combined signal of the eMBMS-1 and eMBMS-2 blocks, it is possible for the UE to separate the data.

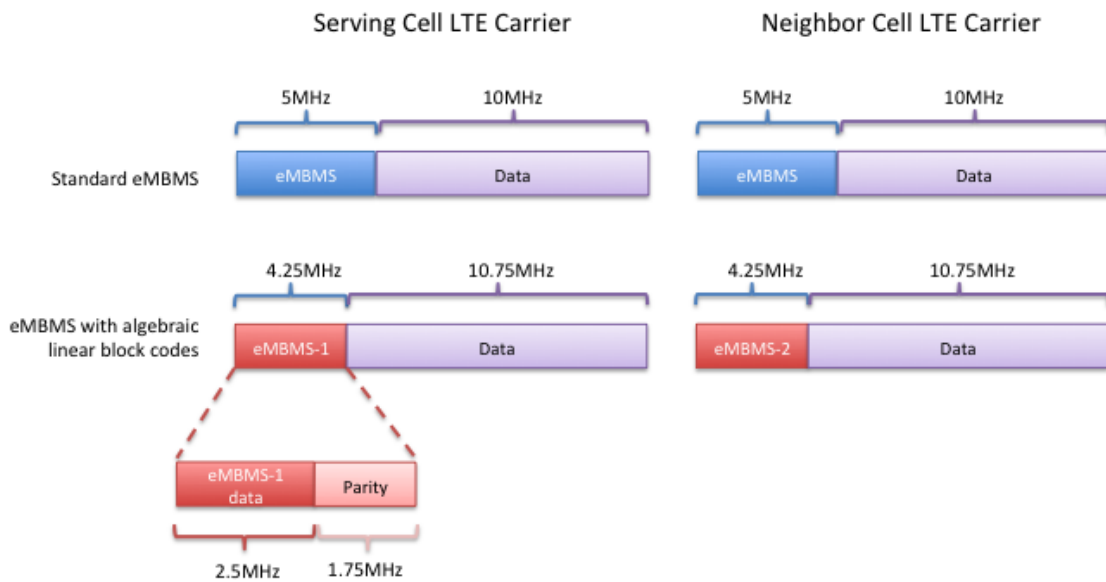


Figure 66: Example of a carrier configuration when ANC and algebraic linear block codes are used

## 4.2. Use in Wireless Mesh Network

The proposed solution can be used to enhance the performance of wireless mesh network by up to three times! There are already practical proposed systems for WMN that rely on a combination of analog and digital network coding along with more traditional opportunistic routing techniques (like MIXIT). However, the methods presented are limited to two-node relay opportunities, and don't discuss three-node to N-node communication with relay.

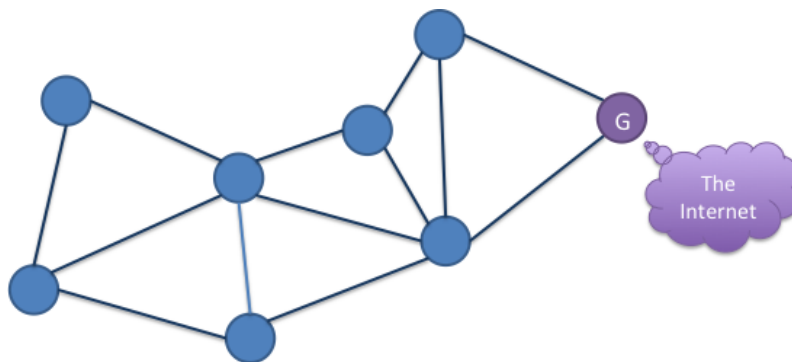


Figure 67: A Mesh Network

The collaborative coding schemes are meant to complement other existing/proposed analog network coding schemes. The relay itself can detect when an opportunity for three nodes to

communicate with each other through it, and dynamically allocate a combination of turbo encoders to the three nodes.

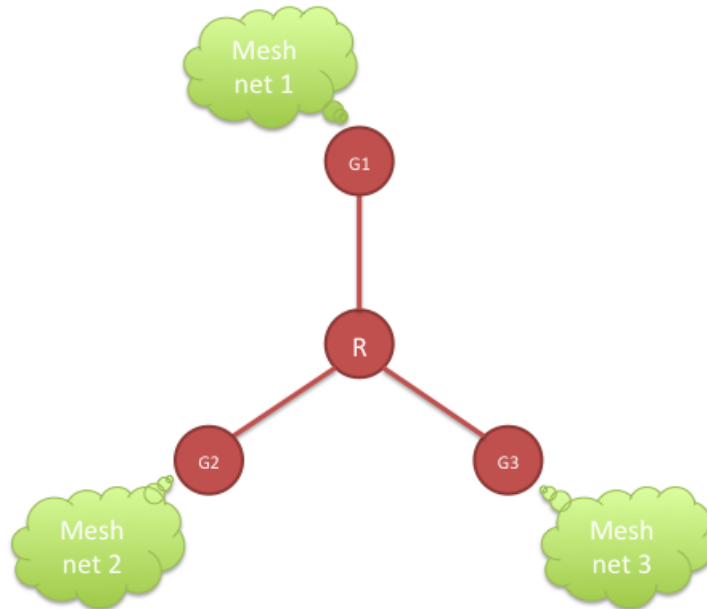
In summary, a practical system with the solutions presented here would require the following:

- The ability of a relay to sense an opportunity where three nodes can communicate through it.
- Once that is established, the relay should be able to dynamically allocate an optimum combination of turbo encoders to three nodes. More research should go into choosing optimum encoders/decoders that take into consideration known RF conditions to choose efficient encoders.
- The ability of the nodes to synchronize their transmission so that the relay receives respective bits from each node at relatively the same time.

These practical systems for wireless mesh networks really exploits the physical aspects of the wireless medium, and blur the lines between the lower three layers of the OSI model to provide significant efficiency improvements.

### **4.3. In Backbone of Wireless Networks**

Using all the previous ideas, a practical VoIP solution using ANC for backbone network and digital network coding with opportunistic routing in local wireless mesh networks is proposed.



**Figure 68: The proposed practical system for VOIP applications using Network Coding**

Each local mesh network is connected to a gateway. The gateways are connected to a central relay, and provide two-way to N-way communication using ANC. Each local mesh network, on the other hand, can have practical solutions that make use of both analog and digital network coding for efficient opportunity routing.



## Chapter 5: The Mesh Network Simulator

Wireless mesh networks are extremely difficult to analyze and simulate due to the fact that there is no static geographical topology for the network itself. Thus, defining the network using text files, and analyzing the data using Command Line Interface (CLI) makes examining the simulated network a tedious task.

In order to streamline the simulation of wireless mesh networks in general, and two-way/three-way communication with relay problems in particular, an intuitive easy to use GUI-based simulator was necessary. This was realized early on in this research, and the IF Wireless Mesh Network Simulator was born.

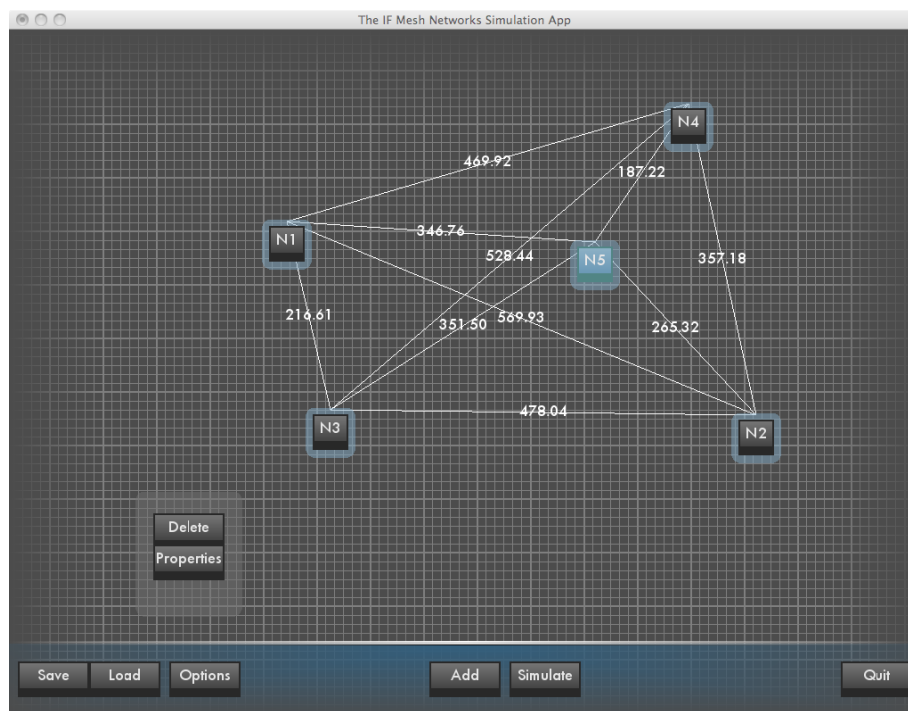


Figure 69: The IF-Mesh network simulation app

The screenshot above shows the GUI simulator in action. In the following sections, the objectives of this application as well as philosophy of its architectural design are discussed.

## 5.1. Design Objectives

Developing the simulator was based on the following design objectives:

- **Fast, Stable & Reliable;** the simulator had to be fast and reliable especially given the fact that complex multi-step encoding/decoding schemes are used. It also had to manage memory usage responsibly especially when performing soft-input soft-output decoding. Moreover, extra care should be taken when creating and destroying dynamic structures to ensure iterative decoding schemes are stable and reliable.
- **Flexible;** the simulator should support a wide range of encoders/decoders, different types of noise and relay fading, multi-step multi-hop communication to support analog and digital network coding. In addition, the simulator should support various mesh network layouts with the ability to change system parameters on the fly before repeating the simulator.
- **Extensible & Expandable;** additional features, encoding/decoding schemes, different kinds of random number generators could be easily added not only by me, but also by other developers and researchers in the future. An Object-Oriented architecture is key in this aspect.
- **Cross Platform;** the codebase should be easily compiled on any operating system. Using cross platform APIs is essential.

## 5.2. Software Architecture

The following diagram shows the high-level software architecture of the IF Mesh Simulator.

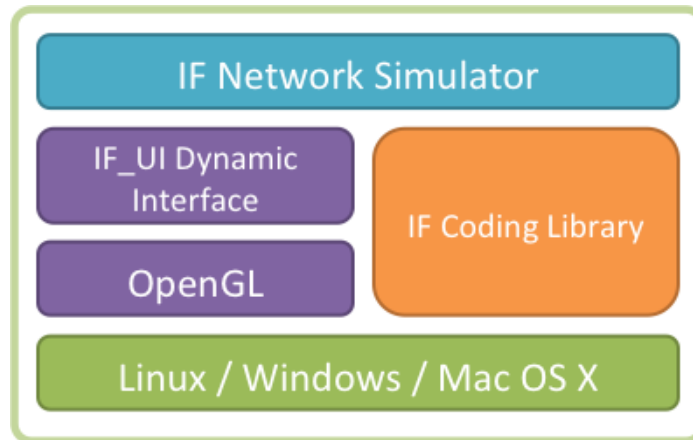


Figure 70: High-Level Software Architecture of the IF Mesh Network Simulator

The high-level architecture is explained as follows:

- **Operating System;** the simulator is written in a platform independent fashion. Development started in Linux, and was moved to Mac OS X in the middle of the development cycle. In addition, later versions of the code were compiled and tested in Windows.
- **IF\_UI Dynamic Human Interface;** this API is the foundation for the GUI. It was written in an Object-Oriented fashion using C++ on top of OpenGL graphics library to enable hardware accelerated rendering of the UI. It also incorporates an animation framework, a UI widget library that includes buttons, sliders, dialogs...etc. a sophisticated dynamic Model-View-Control system to connect widgets to actions on the fly (for example, clicking a button triggers an action), and a fast and reliable data structure library. I had solely written this API to be the foundation of the natural user-interface of the award-winning Multi-User Multi-Touch Interactive Surface Using Computer Vision. [34]
- **IF Coding Library;** the coding library was written in object-oriented fashion in C++ to provide random number generators, encoders and decoders, modulators and demodulators, noise and fading support, binary sources, different interleavers...etc. In other words, this library is the heart and soul of the simulator. It is important to note that it was written in a way to facilitate inheriting from base classes to provided expanded functionality or features.

- **IF Network Simulator**; the mesh network simulator itself is built on top of the pillars discussed above. It not only has an intuitive GUI to enable building specific network configurations easily and fast, but it also enables the user to save/load configurations on the fly. In addition, network and simulation parameters can be changed at any point before starting a simulation. This flexibility was key throughout the period of the research conducted for this thesis; mainly due to the fact that a wide range of encoding and decoding algorithms with different complexities and configurations were simulated.

As of this writing, the simulator has the following features implemented and working reliably:

- A suite of random number generators including an equi-probable bit generator, a Gaussian number generator, and a Rayleigh number generator. The efficient random process is 64-bit allowing for true randomness in the simulation.
- Encoding/Decoding tuples include linear block codes, recursive convolution codes, turbo codes, and all the collaborative coding schemes discussed in this thesis. The user can easily modify the generator matrix of a code, thus, allowing him or her to experiment with different encoders and decoders.
- AWGN channel support: dynamically generate noise given a certain signal to noise ratio.
- Random interleavers for a sequence of numbers of any length.
- An extremely efficient simulation engine that is easily configured specifying the channel (Y-Channel for example), and the encoder used in each node. The simulator takes care of the rest.

The interface of the simulator is described in detail in Appendix B.

## Chapter 6: Conclusions

Analog network coding was utilized as a mean to exploit the physical aspect of the medium in modern wireless network deployments such as wireless mesh networks and certain multicast applications (like LTE eMBMS) to achieve better performance on network level. In a Y-Channel-Relay scenario, the number of transmissions is reduced to only two time slots (a maximum reduction of three folds) when ANC is used. Several practical solutions that embed this opportunistic efficiency in wireless mesh networks were proposed in recent years.

The problem with exploiting analog network coding in the Three-Node-problem is the introduction of erasures at the receiver end. In this thesis, we looked in depth into those scenarios, and calculated the maximum theoretical capacity for different transmission cases. We also looked at existing solutions in literature that depends on Nested Codes and CNC. Then, we proposed two new solutions to the problem; the first is a Nested Codes based solution using Turbo Codes. The other is a novel algebraic linear block codes solution.

Upon studying the existing and proposed solution, we found that the maximum theoretical system throughput on CNC outperforms the linear block codes solution by a slight margin. To achieve this maximum, all three nodes have to be streaming data at all times. However, in most scenarios, data traffic is bursty in nature. The linear block codes solution outperforms the CNC solution both in the number of required transmissions as well as the overall system throughput. Thus, making the linear block codes solution a favorable solution in this case.

When going in depth into the study of analog (physical) network coding, we saw how the line between the physical layer of the OSI model and higher levels can be blurred to bring about better efficiency when looking at overall network performance. In other words, zooming out and looking at the big picture, optimizing the physical layer to work in harmony with the data-link and network layers can significantly reduce the overall number of required time-slots (transmissions) and thereby enhance the performance of wireless networks.

In the last few sections of this thesis, we propose practical areas where Analog Network Coding can be used. That being said, research is still in its beginnings in this area, and there are many practical hurdles that need to be overcome before Analog (Physical) Network Coding can be deployed in live wireless networks.

## 6.1. Future Research

There are many ways the solutions proposed in this thesis can be extended or further analyzed. A few proposals for future research are discussed here:

- The proposed solutions for the Y-Channel-Relay with relay problem can be generalized to include communication between four or more terminals through a common relay.
- A solution based on Low-Density Parity-Check (LDPC) codes [35] can be easily derived and implemented. LDPC are a class of linear error correcting codes that use a sparse parity-check matrix. They can offer performance near the Shannon limit.
- The turbo code solution provided in this thesis can be further enhanced and optimized. The simulated decoding algorithm, and the erasure guessing procedure algorithm could be optimized to gain relatively better performance with lower number of iterations.
- Research can go in the direction of developing higher-layer protocols to support ANC for Two-Way-Relay and Y-Channel-Relay problems.
- As discussed earlier, for ANC solutions to be practically viable, corresponding bits from different terminals should be received in synchronization at the relay, regardless of the distance between any of the nodes and the relay. In addition, most users are mobile in wireless scenarios. More research and effort can go into solving this problem and proposing feasible practical solutions.
- Enhancing the MIXIT system proposed by MIT to include Y-Channel-Relay scenarios; this requires defining methods and procedures in the relay to recognize an opportunity where Y-Channel-Relay communication with ANC is viable. This is a complex problem that can be solved by the use of metaheuristics for example.
- Examples for higher rate algebraic linear block codes can be explored.

- The discussion in this thesis was limited to the binary domain. Research can go into extending the proposed solution to include m-ary systems to achieve higher throughput.

## References

- [1] L. Xiao, T. Fuja, J. Kliewer and D. Costello, "Nested Codes with Multiple Interpretations," in *Information Sciences and Systems, 2006 40th Annual Conference*, March 2006.
  
- [2] S. Katti, D. Katabi, H. Balakrishnan and M. Medrad, "Symbol-Level Network Coding for Wireless Mesh Networks," in *SIGCOMM*, Seattle, 2008.
  
- [3] S. Katti and D. Katabi, "MIXIT: The Network Meets the Wireless Channel," in *Proc. of the Sixth ACM Workshop on Hot Topics in Networks (HotNets-VI)*, 2007.
  
- [4] B. Nazer and M. Gastpar, "Reliable Physical Layer Network Coding," in *Proceedings of the IEEE*, 2011.
  
- [5] S. Katti, S. Gollakota and D. Katabi, "Embracing Wireless Interference: Analog Network Coding," in *SIGCOMM*, Kyoto, 2007.
  
- [6] B. W. Khoueiry, H. Khoshnevis and M. R. Soleymani, "A novel coding strategy for the Y channel," in *submitted to the 2014 IEEE International Symposium on Information Theory (ISIT)*, 2013.



- [7] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Mesh Networks," *IEEE Wireless Communications*, vol. 10, no. 5, pp. 8-14, October 2003.
- [8] S. Chen, P. Lin, D.-W. Huang and S.-R. Yang, "A study on distributed/centralized scheduling for wireless mesh network," in *International Conference on Wireless Communications and Mobile Computing*, Vancouver, 2006.
- [9] "IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," May 2004.
- [10] J. Zhang, Y. P. Chang and I. Marsic, "Network Coding Via Opportunistic Forwarding in Wireless Mesh Networks," *Wireless Communications and Networking Conference*, pp. 1775-1780, 2008.
- [11] T. M. Cover and A. E. Gamal, "Capacity theorems for the relay channel," *IEEE Trans. Inform. Theory*, vol. 25, pp. 572-584, September 1979.
- [12] H. Sato, "Information transmission through a channel with relay (Aloha system technical report)," University of HAWAII, Honolulu, 1976.
- [13] S. Borade, L. Zheng and R. Gallager, "Amplify-and-forward in wireless relay networks: Rate, diversity, and network size," *IEEE Trans. Inform. Theory*, vol. 53, no. 10, pp. 3302-3318, October 2007.

- [14] R. F. Wyrembelski, T. J. Oechtering and H. Boche, "Decode-and-forward strategies for bidirectional relaying," in *Proc. of IEEE-PIMRC*, Cannes, France, 2008.
- [15] R. Ahlswede, N. Cai, S. R. Li and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204-1216, July 2000.
- [16] R. Ahlswede, N. Cai, S. R. Li and R. W. Yeung, "Network Information Flow," *IEEE Trans. on Info. Theory*, pp. 1204-1216, July 2000.
- [17] C. Gkantsidis and P. R. Rodriguez, "Network Coding for Large Scale Content Distribution," in *IEEE Infocom*, 2005.
- [18] H. Gacanin and F. Adachi, "Channel Capacity of Analog Network Coding in a Wireless Channel," 2009.
- [19] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Merard and J. Crowcroft, "XORs in The Air: Practical Wireless Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497-510, September 2008.
- [20] P. Popovski and H. Yomo, "Wireless network coding by amplify-and-forward for bi-directional traffic flows," *IEEE Commun. Lett.*, vol. 11, no. 1, pp. 16-18, January 2007.

- [21] A. Maaref, R. Annavajjala and J. Zhang, "Comparison of Analog and Digital Network Coding Approaches for Bidirectional Relaying with Private Messages to the Relay," in *The 8th Annual IEEE Consumer Communications and Networking Conference*, 2011.
- [22] D. Gundez, A. Yener, A. Goldsmith and H. V. Poor, "The Multiway Relay Channel," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, Korea, Jun.–Jul. 2009.
- [23] M. Park and S. K. Oh, "An Iterative Network Code Optimization for Three-Way Relay Channels," in *Proc. IEEE VTC '09*, September 2009.
- [24] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed., McGraw Hill, 1995.
- [25] T. Cover and J. Thomas, *Elements of information theory*, 1st Edition ed., New York: Wiley-Interscience, 1991.
- [26] R. K. a. M. Medard, "An algebraic approach to Network Coding," in *IEEE/ACM Trans. Netw.*, Oct. 2003.
- [27] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi and B. Leong, "A random linear network coding approach to multicast," in *IEEE Trans. Inf. Theory*, Oct. 2006.
- [28] B. W. Khoueiry, H. Khoshnevis and M. R. Soleymani, "Mobile and Base Station Assisted Cooperative Communications in LTE Advanced Networks: An Overview and Recent

Advances," 2013.

- [29] C.-H. Liu and A. Arapostathis, "Joint network coding and superposition coding for multi-user information exchange in wireless relaying networks," *IEEE-Globecom*, 2008.
- [30] P. Popovski and H. Yomo, "The anti-packets can increase the achievable throughput of a wireless multi-hop network," in *Proc. IEEE Int. Conf. Commun.*, Istanbul, Turkey, Jun. 2006.
- [31] S. Lin and D. J. C. Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed ed., New Jersey: Pearson Prentice Hall, 2004.
- [32] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," in *IEEE Trans. Inf. Theory*, May 2006.
- [33] "3GPP TS 36.440: Evolved Universal Terrestrial Radio Access Network (E-UTRAN); General aspects and principles for interfaces supporting Multimedia Broadcast Multicast Service (MBMS) within E-UTRAN".
- [34] I. Al-Fanek, S. Sadighi and E. Louis, "Multi-User Interactive Surface Using Computer Vision," *IEEE Canadian Review*, no. 62, pp. 20-21, 2010.
- [35] R. G. Gallager, "Low Density Parity Check Codes," *Monograph*, 1963.

- [36] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," Vols. IT-20, pp. 284-287, March 1974.
- [37] P. Robertson, E. Villebrun and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *ICC 95, Gateway to Globalization*, Seattle, June 1995.
- [38] J. H. a. P. Hoeher, "A Viterbi algorithm with soft decision outputs and its applications," in *IEEE GLOBECOM*, Dallas, TX, November 1989.
- [39] C. Berrou, Î. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE International Conference on Communication (ICC)*, Geneva, Switzerland, 1993.
- [40] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-Codes," in *ICC '93*, May 1993.
- [41] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.

# Appendix A: Channel Coding Techniques

In this section, channel coding techniques that were looked at and used as part of this research are described and explained.

## A.1. Linear block codes

A block code (n,k) adds a constant number of parity bits to a message of certain length (k) to give an output of length (n). [24]

A block code takes n-information bits and adds to them a certain number of parity bits to create a coded word of length n. The minimum distance between codewords in the block code code-book governs the detection and correction capability of the code.

Assuming that a block code is used strictly to correct errors in a binary symmetric channel, a t-error-correcting (n,k) linear code can correct a total of  $2^{n-k}$  error patterns.

The relation between the minimum distance between any two codewords in the codebook and the number of bits the code can correct in a message is ruled by the following equation:

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \quad (6.1)$$

Where  $t$  is the number of bits the code can correct, and  $d_{min}$  is the minimum distance between any two codewords in the codebook of the linear block code.

Based on that, the probability that a received message is decoded incorrectly will be less than or equal to the probability that we receive a message more than  $t$  bits in error:

$$P(\text{Message received with error}) = P_M \leq \sum_{j=t+1}^n \binom{n}{j} p^j (1-p)^{n-j} \quad (6.2)$$

From that, we can get an approximation for the decoded bit error probability:

$$P_B \approx \frac{1}{n} \sum_{j=t+1}^n j \binom{n}{j} p^j (1-p)^{n-j} \quad (6.3)$$

The decoding is done by choosing the closest codeword to the received vector based on hard or soft distances.

### A.1.1. Hamming codes

Hamming codes are the simplest form of linear block codes. They have the capability of correcting one bit error at least in an encoded message. Hamming codes will be used later on in this thesis to derive a solution to digital network coding for three-way communication through a relay. [24] The probability of error reduces to the following:

$$P(\text{Message received with error}) = P_M \leq \sum_{j=2}^n \binom{n}{j} p^j (1-p)^{n-j} \quad (6.4)$$

And thus, the probability of bit error:

$$P_B \approx \frac{1}{n} \sum_{j=2}^n j \binom{n}{j} p^j (1-p)^{n-j} \quad (6.5)$$

### A.1.2. Bose-Chaudhuri-Hocquenghem (BCH) codes

BCH codes are a set of linear block codes that can be efficiently decoded using simple algebraic algorithms. They are known to perform really well for low to moderate block lengths. In addition, they have various rates and block code lengths that are well established and tabulated.

The design parameters for BCH codes are defined by the following equations:

$$n = 2^m - 1 \quad (6.6)$$

$$n - k \leq mt \quad (6.7)$$

$$d_{min} \geq 2t + 1 \quad (6.8)$$

Where  $n$  is the block length,  $m$  is the info-message length,  $k$  is,  $t$  is the least number of bit errors the code can correct, and  $d_{min}$  is the minimum distance between any two codewords in the codebook of the BCH code.

Usually well known BCH codes are tabulated in triplet format  $(n, k, t)$ .

## **A.2. Convolutional (Trellis) Codes**

### **A.2.1. Description**

Convolutional codes are a class of codes that is generated by passing a sequence of bits into a finite state register. Thus, they can be conveniently described using a state graph or trellis. Each bit of the output of the encoder is the result of adding (XORing) pre-determined bits from the registers.

### **A.2.2. Recursive Convolutional Codes**

Recursive convolutional codes are a special breed of convolutional codes. They are realized by having a feedback shift register. Hence, they are mathematically represented using a transform domain generator matrix that is comprised of ratios of polynomials. The recursive nature of these codes causes them to have infinite-length impulse responses.



### A.2.3. Encoding

Assuming that we have binary information, data are passed in groups of  $k$ -bits shifted  $k$ -times every time through  $K$  registers of size  $k$ -bits. Hence, the total number of stages is equal to  $kK$ . After each shift,  $n$  adders determine the encoded sequence. [24]  $K$  is called the “Constraint Length”. The following diagram shows a  $(n, k, K)$  convolutional encoder.

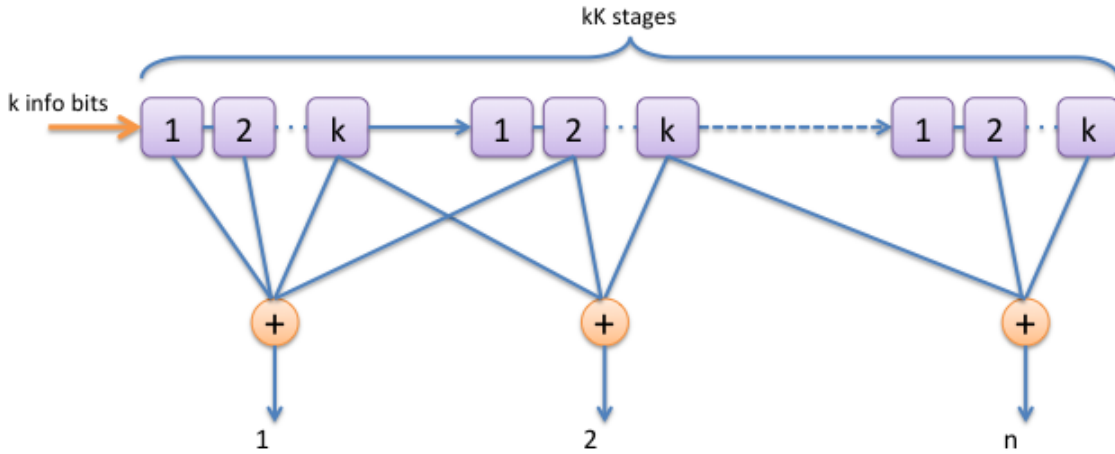


Figure 71: Convolutional Encoder example

Throughout this thesis, for the sake of simplicity, registers of one-bit size are used. i.e.  $k$  is assumed to be equal to 1.

Since the encoder can be thought of as a finite state machine, it can be represented graphically using a state transition diagram or a trellis, or mathematically using its impulse response as a function. In general, a rate  $\frac{k}{n}$  encoder with constraint length ( $K$ ) has  $2^{k(K-1)}$ . In addition, the code can be represented using the transform domain generator matrix. The code-word  $c$  is a result of multiplying the info bit sequence  $u$  by the transform domain generator matrix  $G$ . The generator matrix for an RSC is given, in general, in the following form:

$$G(D) = [I_k | P(D)] \quad (6.9)$$

One of the factors that determine the performance of the convolutional code is the minimum free distance that can be expressed by the following upper bound:

$$d_{free} \leq \min_{l>1} \left\lfloor \frac{2^{l-1}}{2^l - 1} (K + l - 1)n \right\rfloor \quad (6.10)$$

#### A.2.4. Decoding

There are many decoding algorithms for convolutional codes. Some rely on hard-output, hard-decision decoding, and others on soft-output iterative algorithms.

In this research, the interest is in iterative soft-output algorithms. The reason for that is because in the collaborative recursive code that is introduced later on in this thesis acts as a concatenated code; the soft output of one decoder can be used as input for the other decoder. Hence, allowing for intuitive iterative decoding algorithms.

The best metric to use for soft-output algorithms is the a posteriori probability of the detected symbol conditioned on the received signal. Assuming we have a binary PSK modulation that passes through an AWGN channel, the received signal vector can be expressed as follows:

$$r = (2c - 1)\sqrt{\varepsilon_c} + n, \text{ where } c \text{ is binary output of the encoder} \quad (6.11)$$

Decoding decisions are based on the following maximum a posteriori probability (MAP) of an info bit in a sequence:

$$P(x_i = 0|r) = 1 - P(x_i = 1|r), \text{ where } x_i \text{ is the info bit} \quad (6.12)$$

The algorithm makes a decision on each received symbol by selecting the info bit that corresponds to the maximum MAP value. The smaller the difference between two MAP values for an info bit, the less reliable the decoding decision. This criterion minimizes the probability of symbol error, and hence, MAP is the optimum soft output metric for decoding decisions.

The BCJR algorithm is a recursive-decoding algorithm that produces a hard decision on an info bit, and a soft output value for the reliability of that hard decision based on MAP calculations. This algorithm is used in subsequent sections to derive a collaborative solution for solving the three-node with relay using recursive convolutional codes. The coding gain of convolutional code over an un-coded binary PSK system is as follows:

$$\text{Coding gain} \leq 10 \log_{10}(R_c d_{free}), \text{ where } R_c \text{ is the rate of the code} \quad (6.13)$$

#### ***A.2.4.1. The BCJR soft decoding algorithm***

The BCJR algorithm is a symbol-by-symbol recursive soft decoding algorithm that is based on MAP calculations rather than searching for the most probable input sequence. [36]

As discussed before, convolutional codes can be represented as finite state machines, and therefore, they can be expressed mathematically by finite states as outputs that rely on the current state of the decoder and the new input.

The probability of error using a BCJR soft decoder and BPSK modulation is given by the following formula (Proakis page 514):

$$P_b \leq \frac{1}{k} \frac{\partial}{\partial Y} T(Y, Z) |_{Y=1, Z=\exp(-R_c \gamma_b)} \quad (6.14)$$

Where:

$T(Y, Z)$  is the transfer function of the convolutional code.

$R_c$  is the rate of the code.

$\gamma_b$  is the signal to noise ratio per bit or SNR per bit:

$$\gamma_b = \frac{\mathcal{E}_b}{N_0} \quad (6.15)$$

The BCJR algorithm is a soft input soft output algorithm that not only gives a decision for each symbol, but also a soft value that represents the confidence in the decoder decision for each decoded symbol. Let's assume we have an RSC with  $k = 1$ , and binary input bit  $u_i \in \{0,1\}$ , the following mathematical functions will govern the encoder:

$\mathbf{c}_i = f_c(u_i, \sigma_{i-1})$ , where  $\mathbf{c}_i$  is the codeword based on the previous state of the encoder  $\sigma_{i-1}$  and the input bit  $u_i$

$\sigma_i = f_s(u_i, \sigma_{i-1})$ , where  $\sigma_i$  is the current state of the encoder based on the previous state  $\sigma_{i-1}$  and the input bit  $u_i$ .

If we receive the sequence  $\mathbf{y} = (y_1, y_2, \dots, y_N)$ , the maximum symbol by symbol a posteriori will decode the input  $u_i$  based on:

$$\begin{aligned} \hat{u}_i &= \arg \max_{u_i \in \{0,1\}} P(u_i | \mathbf{y}) \\ &= \arg \max_{l \in \{0,1\}} \sum_{(\sigma_i, \sigma_{i-1}) \in S_l} p(\sigma_{i-1}, \sigma_i, \mathbf{y}) \end{aligned} \quad (6.16)$$

Where  $(\sigma_i, \sigma_{i-1}) \in S_l$  corresponds to all the state pairs based on an input bit  $l \in \{0,1\}$ .

The probability in the last equation can be reduced to the following:

$$p(\sigma_{i-1}, \sigma_i, \mathbf{y}) = p(\sigma_{i-1}, \mathbf{y}_1^{(i-1)}) p(\sigma_i, \mathbf{y}_i | \sigma_{i-1}) p(\mathbf{y}_{i+1}^{(N)} | \sigma_i) \quad (6.17)$$

The probability is the result of multiplying three terms together. The first term depends on the past (the previous state of the decoder and the last received bit), the second term depends on the

present (the current decoder state and the current received bit given a past state), and the last term depends on the future (the next received bit given the current state of the decoder). We can define the following to simplify the formula above:

$$\alpha_{i-1}(\sigma_{i-1}) = p(\sigma_{i-1}, \mathbf{y}_1^{(i-1)}) \quad (6.18)$$

$$\beta_i(\sigma_i) = p(\mathbf{y}_{i+1}^{(N)} | \sigma_i) \quad (6.19)$$

$$\gamma_i(\sigma_{i-1}, \sigma_i) = p(\sigma_i, \mathbf{y}_i | \sigma_{i-1}) \quad (6.20)$$

The probability above reduces to:

$$p(\sigma_{i-1}, \sigma_i, \mathbf{y}) = \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i) \quad (6.21)$$

Hence, the  $i$ -th bit estimation is equivalent to:

$$\hat{u}_i = \arg \max_{l \in \{0,1\}} \sum_{(\sigma_i, \sigma_{i-1}) \in S_l} \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i) \quad (6.22)$$

The beauty of the last equation is that alpha, beta, and gamma values can be computed recursively:

**Forward Recursion for  $\alpha_i(\sigma_i)$ :**

$$\alpha_i(\sigma_i) = \sum_{\sigma_{i-1} \in \Sigma} \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \quad (6.23)$$

Where  $\sigma_{i-1} \in \Sigma$  corresponds to all possible previous states. We can assume that the initial state of the decoder is all zeros, and hence the initial condition of the decoder is:

$$\alpha_0(\sigma_0) = P(\sigma_0) = \begin{cases} 1 & \sigma_0 = 0 \\ 0 & \sigma_0 \neq 0 \end{cases} \quad (6.24)$$

From the previous two equations, we can calculate all the equations for  $\alpha_i(\sigma_i)$  recursively.

**Backward Recursion for  $\beta_i(\sigma_i)$ :**

$$\beta_{i-1}(\sigma_{i-1}) = \sum_{\sigma_i \in \Sigma} \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i) \quad (6.25)$$

Where  $\sigma_{i-1} \in \Sigma$  corresponds to all possible previous states. We can assume that the final state of the decoder is all zeros, and hence the final condition of the decoder for a sequence length of  $N$  is:

$$\beta_N(\sigma_N) = \begin{cases} 1 & \sigma_N = 0 \\ 0 & \sigma_N \neq 0 \end{cases} \quad (6.26)$$

From the previous two equations, we can calculate all the equations for  $\beta_i(\sigma_i)$  recursively.

**Calculation of  $\gamma_i(\sigma_{i-1}, \sigma_i)$ :**

In the calculation of gamma, we are going to assume that we have equiprobable sources ( $P(u_i = 0) = P(u_i = 1) = \frac{1}{2}$ ), and that the transition between state  $\sigma_{i-1}$  and  $\sigma_i$  is possible.

As a soft output algorithm, the BCJR algorithm provides a value for the level of certainty for the decision made per bit (or symbol) in the form of  $P(u_i | \mathbf{y})$ . The likelihood ratio is given as:

$$\begin{aligned} L(u_i) &= \ln \frac{P(u_i = 1 | \mathbf{y})}{P(u_i = 0 | \mathbf{y})} \\ &= \frac{\sum_{(\sigma_i, \sigma_{i-1}) \in S_1} \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i)}{\sum_{(\sigma_i, \sigma_{i-1}) \in S_0} \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i)} \end{aligned} \quad (6.27)$$

Unfortunately, the following equations can't be numerically stable when computed using a computer program. The solution to that is using the Log-APP algorithm.

#### A.2.4.2. The log-APP algorithm

The log-APP algorithm [37] is a modification to the original BCJR algorithm to make it numerically stable. [38] We define the following term by using the logarithms of BCJR terms:

$$\tilde{\alpha}_i(\sigma_i) = \ln(\alpha_i(\sigma_i)) = \ln \left[ \sum_{\sigma_{i-1} \in \Sigma} \alpha_{i-1}(\sigma_{i-1}) \gamma_i(\sigma_{i-1}, \sigma_i) \right] \quad (6.28)$$

$$\tilde{\beta}_i(\sigma_i) = \ln(\beta_i(\sigma_i)) = \ln \left[ \sum_{\sigma_{i-1} \in \Sigma} \gamma_i(\sigma_{i-1}, \sigma_i) \beta_i(\sigma_i) \right] \quad (6.29)$$

$$\tilde{\gamma}_i(\sigma_{i-1}, \sigma_i) = \ln(\gamma_i(\sigma_{i-1}, \sigma_i)) \quad (6.30)$$

The a posteriori  $L$  can be calculated after modifying the initial conditions to go with the updated equations:

$$L(u_i) = \max_{(\sigma_i, \sigma_{i-1}) \in S_1} \{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \tilde{\gamma}_i(\sigma_{i-1}, \sigma_i) + \tilde{\beta}_i(\sigma_i) \}^* - \max_{(\sigma_i, \sigma_{i-1}) \in S_0} \{ \tilde{\alpha}_{i-1}(\sigma_{i-1}) + \tilde{\gamma}_i(\sigma_{i-1}, \sigma_i) + \tilde{\beta}_i(\sigma_i) \}^* \quad (6.31)$$

Some approximations can be used to simplify the latter term further.

### A.3. Turbo Codes

#### A.3.1. Encoding

Turbo encoders are built using two or more recursive systematic convolutional encoders. Each one of those encoders is fed with a differently interleaved version of the input sequence. That

parallel concatenation and interleaving process is what gives turbo codes its high performance attributes. [39]

The choice of the used RSC components has an effect on the performance of the code. For a well-designed turbo code, RSC with high free distance should be used.

The following diagram shows a turbo encoder (This is the same turbo encoder used in the simulation):

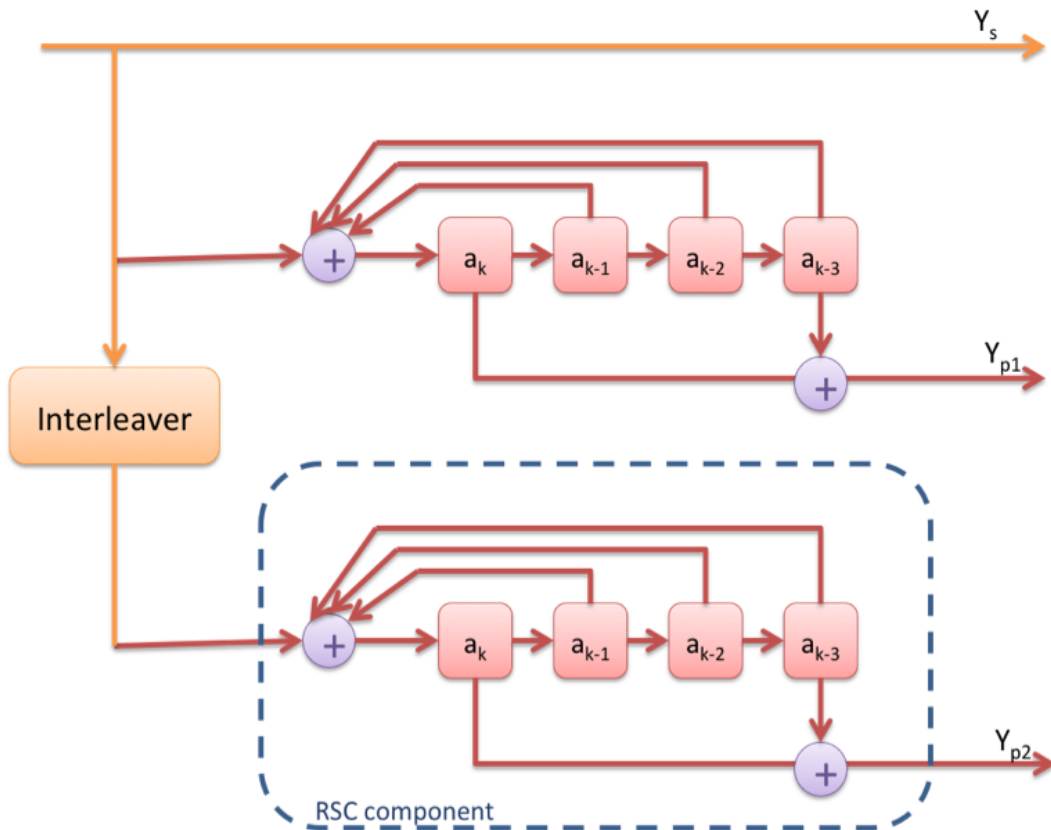


Figure 72: Turbo Encoder

As seen in the diagram, this turbo encoder consists of two RSC encoders concatenated in parallel. The first RSC encoder produces the first parity bit. The second RSC encoder, on the other hand, receives an interleaved version of the input sequence, and produces the second parity bit. It is interesting to note here that all turbo encoders are systematic. Thus, the input bit itself is part of the codeword. Optional puncturing can be used in the output to reduce the rate from 1:3



to 1:2 (other puncturing schemes can be used as well). In addition, the two RSC encoders shouldn't necessarily be identical.

### ***A.3.1.1. Interleaving***

Interleaving is the process of shuffling the input bits before feeding them to the different RSC encoders that constitute the turbo encoder. This mechanism helps avoid producing low weight codes from both RSC encoders at the same time. In other words, if one RSC encoder produces a low weight codeword, interleaving will minimize the probability of having a low codeword produced by the second RSC.

There are many types of interweavers that can be used. One way is to use a look up table in which the interleaving pattern is defined. Another way is to use a row-column interleaver in which data is written row wise but read column wise.

The following is a short diagram that explains how the row-column interleaver works (This is the same concept used in the simulation for interleaving).

**Table 40: Interleaver Input**

<b>X1</b>	<b>X2</b>	<b>X3</b>	<b>X4</b>
<b>X5</b>	<b>X6</b>	<b>X7</b>	<b>X8</b>
<b>X9</b>	<b>X10</b>	<b>X11</b>	<b>X12</b>
<b>X13</b>	<b>X14</b>	<b>X15</b>	<b>X16</b>

**Table 41: Interleaver Output**

<b>X1</b>	<b>X5</b>	<b>X9</b>	<b>X13</b>	<b>X2</b>	<b>X6</b>	<b>X10</b>	<b>X14</b>	<b>X3</b>	<b>X7</b>	<b>X11</b>	<b>X15</b>	<b>X4</b>	<b>X8</b>	<b>X12</b>	<b>X16</b>
-----------	-----------	-----------	------------	-----------	-----------	------------	------------	-----------	-----------	------------	------------	-----------	-----------	------------	------------

We can see that the table in the bottom is an interleaved version of the input sequence.

### A.3.2. Iterative Decoding

Turbo decoders are built using Soft Input/Soft Output (SISO) components and use an iterative approach to decode a sequence. Basically, a SISO decoder produces a soft output that decides not only whether a received bit originally a high or low bit, but also how probable that decision is. An APP (A Posteriori Probability) algorithm for soft outputs was first proposed by Bahl, but later modified by Berrou & others. [40]The algorithm is now known as the BCJR algorithm.

The APP states that a decoded data bit is equal to a high bit or a low bit is obtained by the summing the joint probability over all states. i.e.:

$$P\{d_k = i | R_1^N\} = \sum_m \lambda_k^{i,m} \quad \text{Where } i = 0, 1 \quad (6.32)$$

Then, the log-likelihood ratio (LLR) is the logarithm of the ratio of APPs

$$\Rightarrow L(d_k) = \log \frac{\sum_m \lambda_k^{1,m}}{\sum_m \lambda_k^{0,m}}$$

⇒ Logarithm of the likelihood that the bit is 1 over the likelihood that the bit is zero

⇒ LLR represents a soft output. We decide that the received bit was originally one if he LLR is positive, and zero if it was negative.

$$d_k = 1 \quad \text{if} \quad L(d_k) > 0$$

$$d_k = 0 \quad \text{if} \quad L(d_k) < 0$$

The basic idea of a feedback turbo decoder is to communicate messages between two (or more) SISO decoders iteratively. During iteration, SISO2 receives a message from SISO1. SISO2 uses it as priory data (pre-known data), appends it to its likelihood ratio of the original input bit, and then passes an extrinsic piece of information back to the first SISO decoder who will in turn use that extrinsic data as priory data, and so on so forth until we reach a pre determined condition on the likelihood ratio, or do a known number of iterations. It is important to note the extrinsic data has to be interleaved (or de-interleaved) before being fed from one SISO to another.

Decoding with a feedback loop

$$L(d_k) = L_c(x_k) + L_e(d_k) = \log \left[ \frac{p(x_k | d_k = 1)}{p(x_k | d_k = 0)} \right] + L_e(d_k) \quad (6.33)$$

Where  $L(d_k)$  is the soft output of the decoder,  $L_c(x_k)$  is the LLR channel measurements (has only to do with the channel).  $L_e(d_k)$  is the redundant (extrinsic) information supplied by the decoder, and it doesn't depend on the decoder input ( $x_k$ ). The following diagram shows a turbo decoder.

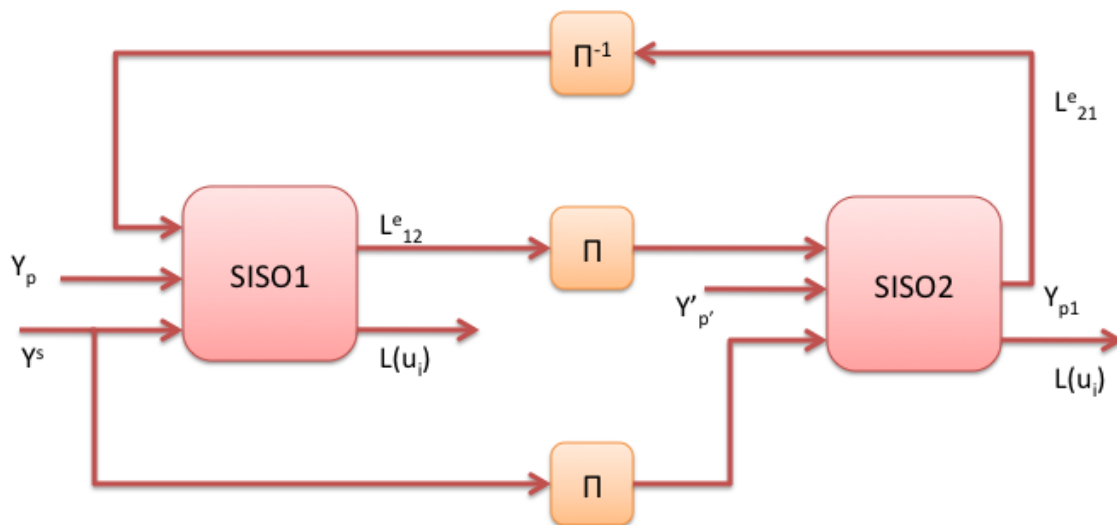


Figure 73: Turbo Decoder

#### A.4. Concatenated Codes

Concatenating codes are used to get better performance out of a communication system by having two codes, one binary and one non-binary in a way where the binary code codewords are seen as symbols to the non-binary code. The binary code  $(n, k)$  is usually called the inner code, and the non-binary code  $(N, K)$  is called the outer code. The overall communication system would look as follows:

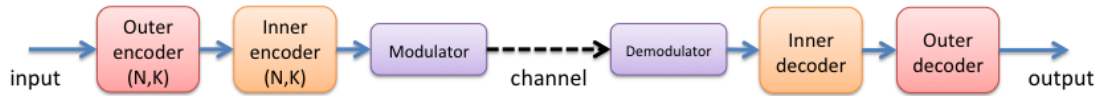


Figure 74: A communication system with concatenated codes

The system above is equivalent to a  $(Nn, Kk)$  long binary code with  $Kk$  information bits and  $Nn$  overall length. The minimum distance of the code will be  $d_{min}D_{min}$  where  $d_{min}$  is the minimum distance of the inner code and  $D_{min}$  is the minimum distance of the outer code. Concatenation is one of the methods where codes can be combined to achieve longer codes, and thus, better performance out of a communication system.

### A.5. LDPC Codes

Low-Density Parity-Check (LDPC) codes are a class of linear block codes with performance close to the Shannon limit. As the name suggests, those codes are constructed from a sparse parity-check matrix. [35]

Encoding in LDPC requires constructing a low-density (sparse) polynomial generator. Usually the sparse polynomial is constructed based on specific constraints that govern the number of 1's in a column, and in a row of the matrix.

To decode an LDPC practically, an iterative belief-propagation algorithm can be used. This is done by treating each parity bit as an independent single parity check code, decoding it separately using SISO techniques (similar to decoding turbo codes). The soft decision information from each SISO decoder is crosschecked and updated with other redundant SISO decoders that use the same information bit as input. This is repeated for a number of iterations before making a final decoding decision. LDPC codes have exceptional performance that rivals turbo codes.

## Appendix B: The Mesh Network Simulator Interface

In this section, the main elements of the interface of the simulator are described and discussed. It is important to note that the discussion here is based on the current features and implementation as of the date of this writing. However, the code of the simulator is still under development, and the final implementation of the interface might be different than the one described here.

### B.1. The Home Menu

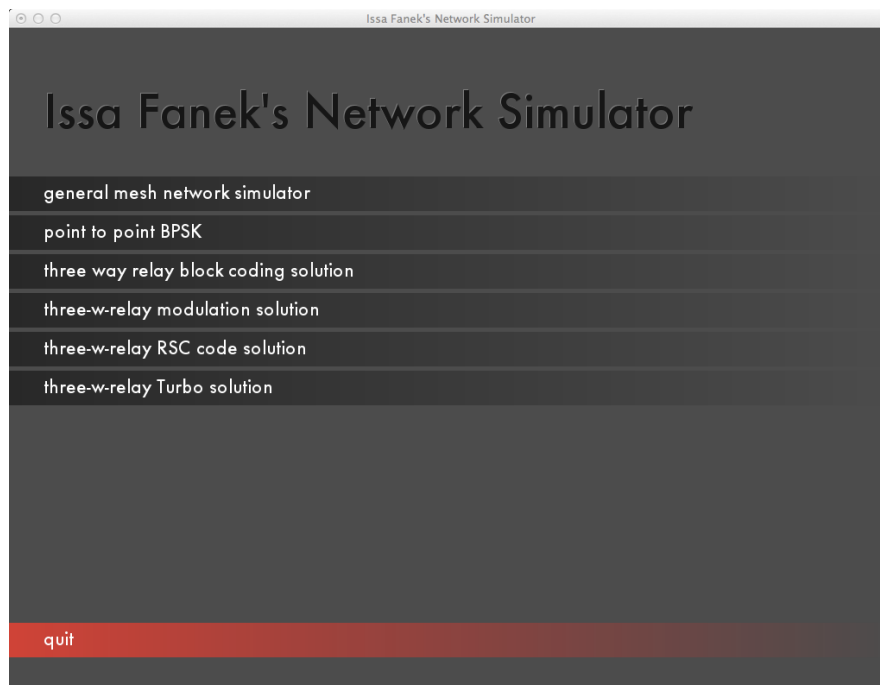


Figure 75: The Home Menu (Hub) of the IF Mesh Network Simulator

The Home Menu is the first dialog that greets the user when he starts the simulator. It serves as the hub to general main interface as well as the different saved pre-configuration.

### B.2. The Main Interface

The main interface is shown after selecting a configuration from the Main Menu.

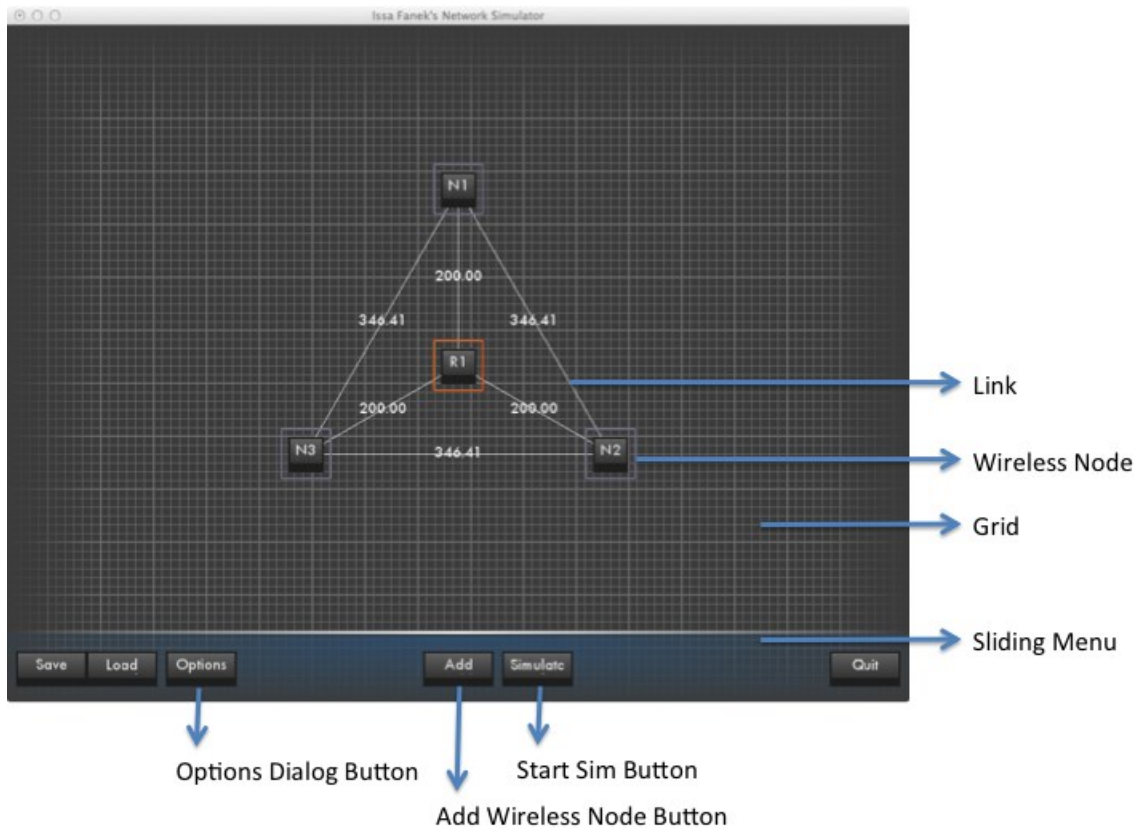


Figure 76: The Main Interface of IF Mesh Network Simulator

The main interface consists of the following:

- **The Grid**; the grid is the canvas in which you can freely add and drag wireless mesh networks. The distance between nodes is measured in points, but can conveniently be transformed to the metric system using a scaling factor.
- **Wireless Node**; this widget represents a wireless node in a wireless mesh network. Nodes can be dragged and placed on the grid. They can also be added and deleted dynamically. Clicking on a Node opens its properties dialog in which you can change the node's modulator, encoder and other options.
- **The Link**; the link represents the wireless medium and embodies underneath the noise and fading characteristics of the channel. It is displayed on the grid as a line between two wireless nodes, and the distance in points is represented on the line.

- **Sliding Menu (Drawer)**; this feature is what sets apart this human interface than other simulators in the market. It intuitively reveals itself with a smooth sliding animation when the mouse pointer goes to the bottom of the window to show more functionality, and disappears when the user is done with it so that it is out of the way when he or she is working on the layout of the network. The drawer includes an “Add” button that adds a new wireless node to the layout, a “Simulate” button that starts the simulation, and an “Options” button that shows the options dialog box when clicked. The following figure shows the two states of the sliding drawer.

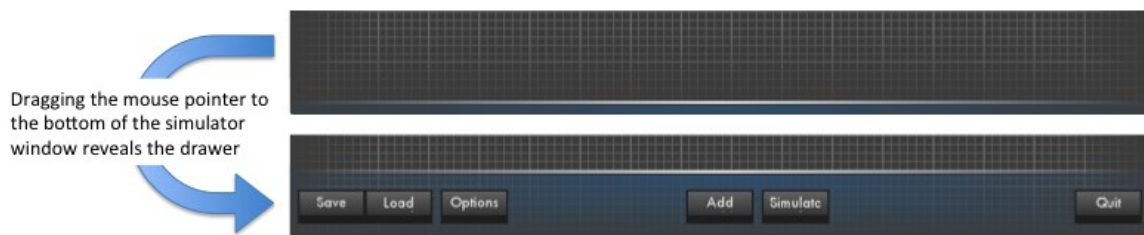
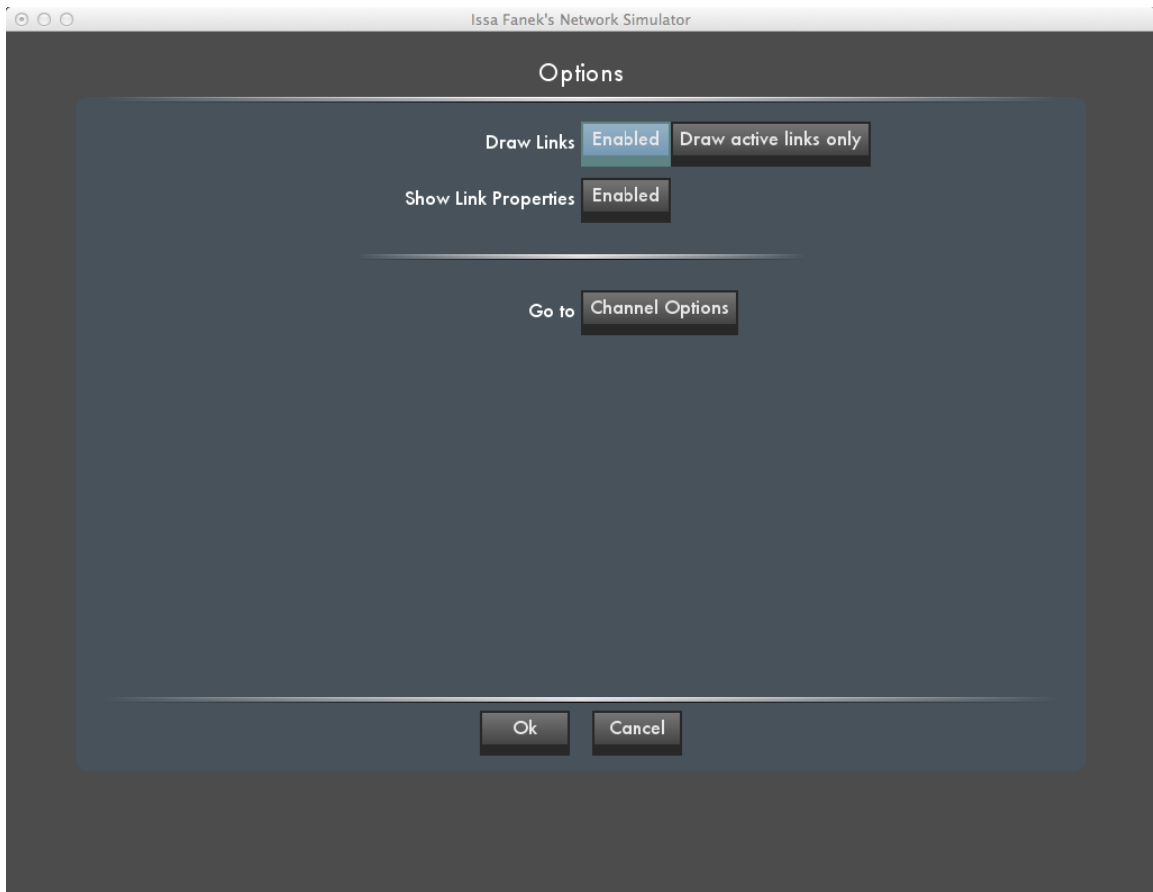


Figure 77: The Sliding Menu (Drawer)

### B.3. The Options Dialog

The Options Dialog includes a number of buttons to govern the general look and feel of the Grid. In addition, it has a “Channel Options” button that takes the user to the wireless medium options dialog.

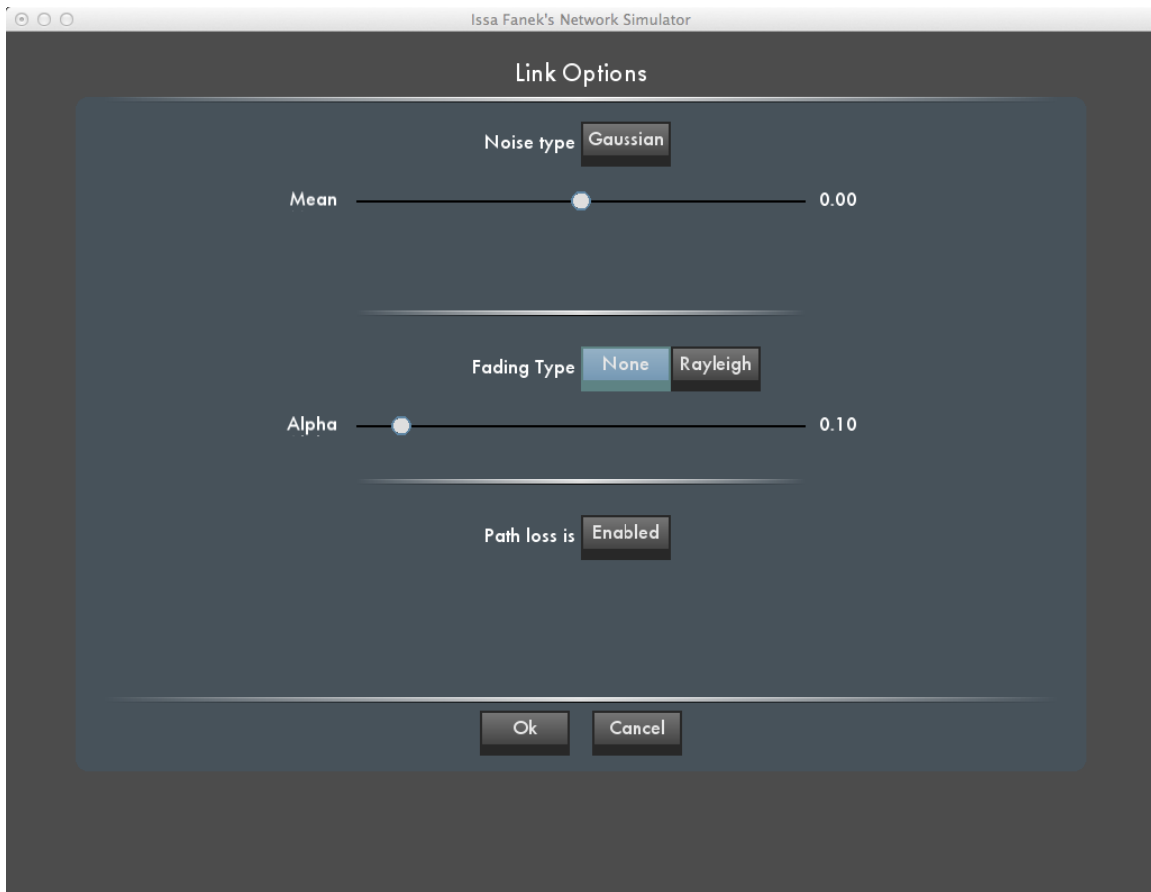


**Figure 78: Options Dialog**

#### **B.4. Link (Channel) Options Dialog**

The Link (Channel) Options dialog includes a number of buttons that sliders that control the properties of the wireless medium. From here, one can set the Noise type and the mean of Noise as well as the channel fading type and the alpha of the channel.





**Figure 79: Link (Channel) Options Dialog**

## **B.5. The Node Properties Dialog**

The Node Properties dialog is displayed when one clicks on any of the wireless nodes in the grid. From this view, one can control several aspects of the node like the used coding and modulation schemes, whether the node sends or receives data, and to whom it should send or receive from.



**Figure 80: Node Properties Dialog**

The included functionality in the simulator and the GUI were limited to the intents and purposes of this research. However, the final goal for this project is to have a versatile, intuitive and powerful interface to simulate any complex wireless mesh network in a fast and reliable fashion. For this reason, the code that I have developed for this project will be released at the end of this research for others to tinker with, expand and enhance.