

---

# **Topological arrangement of nodes in wireless networks suitable for the implementation of network coding**

---

**Dissertation submitted in fulfilment of the requirements for the degree Master of  
Engineering in Computer Electronic Engineering at the Potchefstroom campus of  
the North-West University**

---

**F.J. Böning**

20072155

**Supervisor:**

**Prof. A.S.J. Helberg**

**Co-Supervisor:**

**Mrs. M.J. Grobler**

**November 2010**

---

---

# Declaration

I, Frans Johan-Henry Böning hereby declare that the dissertation entitled “Topological arrangement of nodes in wireless networks suitable for the implementation of network coding” is my own original work and has not already been submitted to any other university or institution for examination.

---

F.J. Böning

Student number: 20072155

Signed on the 19<sup>th</sup> day of November 2010 at Potchefstroom.

---

# Acknowledgements

First of all I would like to thank God almighty for all of the blessings I received and the ability He gave me to complete this research.

My two study leaders, Prof. Albert Helberg and Leenta Grobler for the guidance and advice.

Eskom for their financial support through the masters degree bursary.

The Telkom SAAB-Grintek Centre of Excellence for their financial support to the TeleNet research group.

All of the members of the TeleNet research group for the additional guidance, advice and support.

My family and friends for all the prayers and moral support.

My wife, Eloïse for her love and patience which kept me going.

---

## Abstract

Network coding refers to the implementation of coding methods to utilize network connections more efficiently. Network coding is commonly researched in the information theory field, but very little research is being done on the physical implementation thereof. One exception is COPE where network coding is implemented in wireless networks for unicast transmission sessions.

In this dissertation, we discuss the physical arrangement of wireless nodes to form topologies suitable for the implementation of network coding. We implement linear network coding in wireless ad hoc networks for multicast transmission sessions.

We calculate the areas in which each wireless node must be located for a specific network coding suitable topology to be formed. The identified topologies are simulated in OPNET Modeler and then implemented on a six node testbed, to analyse the effect of implementing network coding in these topologies.

We provide results indicating the trade-off between reduced network load and higher end-to-end delay when our developed network coding algorithm is active in the respective topologies. The results indicate that the developed network coding scheme will produce better overall performance when implemented in sensor networks or highly congested ad hoc networks.

**Keywords:** *Ad hoc Networks, Network Coding, Node Placement, Topology Boundaries, Multicast transmission*



---

## Opsomming

Netwerkkodering verwys na die implementering van koderingsmetodes om netwerkverbindings meer effektief te benut. Netwerkkodering word algemeen in die veld van informasieteorie nagevors, maar baie min navorsing word tans op die praktiese implementering daarvan gedoen. 'n Uitsondering op hierdie waarneming is COPE, waar netwerkkodering in draadlose netwerke vir een-tot-een of "unicast" kommunikasiesessies geïmplementeer word.

In hierdie verhandeling bespreek ons praktiese draadlose netwerk topologieë wat geskik is vir netwerkkodering. Liniêre netwerkkodering word in draadlose ad hoc netwerke geïmplementeer vir een-tot-baie of "multicast" kommunikasiesessies.

Ons bereken die area waarin elke draadlose node moet voorkom vir 'n spesifieke netwerkkoderings geskikte topologie om gevorm te word. Ons simuleer die geïdentifiseerde netwerkkodering geskikte topologieë in die OPNET simulasië omgewing, waarna dit geïmplementeer word op 'n ses node toetsbed en analiseer die effek wat netwerkkodering op hierdie topologieë het.

Ons verskaf resultate wat die voordeel van 'n verlaagde netwerk las en die nadeel van 'n hoër eindpunt-tot-eindpunt netwerkpakkie vertraging aandui wanneer die ontwikkelde netwerkkoderingsalgoritme gebruik word in die geïdentifiseerde topologieë. Volgens hierdie resultate kan ons tot die slotsom kom dat die ontwikkelde netwerkkoderingsskema beter sal presteer wanneer dit in sensornetwerke of oorbelaaide ad hoc netwerke geïmplementeer word.

# Contents

<b>List of Figures</b>	<b>xiv</b>
<b>List of Tables</b>	<b>xviii</b>
<b>List of Acronyms</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Objective . . . . .	3
1.3 Issues to be addressed . . . . .	3
1.4 Methodology . . . . .	4
1.4.1 Literature study . . . . .	4
1.4.2 Network coding suitable topology identification . . . . .	6
1.4.3 Distance vs. transmission rate calculations . . . . .	6
1.4.4 Effects of interference and possible solutions . . . . .	7
1.4.5 Simulate topologies . . . . .	7
1.4.6 Implement topologies . . . . .	7
1.4.7 Discussion and conclusion . . . . .	8
1.5 Beneficiaries . . . . .	8
1.6 Validation and verification . . . . .	8

---

1.6.1	Validation . . . . .	8
1.6.2	Verification . . . . .	9
1.7	The research process . . . . .	9
1.7.1	Literature surveys . . . . .	10
1.7.2	Research proposal . . . . .	11
1.7.3	Research work plan . . . . .	11
1.7.4	Conducting research . . . . .	11
1.7.5	Research document and publications . . . . .	11
1.8	Document structure . . . . .	11
<b>2</b>	<b>Literature study</b>	<b>13</b>
2.1	Network coding . . . . .	13
2.1.1	Unicast sessions . . . . .	14
2.1.2	Multicast sessions . . . . .	15
2.2	Wireless networks: IEEE 802.11 standards . . . . .	16
2.2.1	IEEE 802.11a . . . . .	17
2.2.2	IEEE 802.11b . . . . .	17
2.2.3	IEEE 802.11g . . . . .	18
2.2.4	IEEE 802.11n . . . . .	18
2.3	Wireless networks: Concepts . . . . .	18
2.3.1	Interference . . . . .	19
2.3.2	Fresnel zone . . . . .	20
2.3.3	Multipath interference . . . . .	20
2.3.4	Fading . . . . .	21
2.3.5	Shadowing . . . . .	22
2.3.6	Communication transmission rate and distance . . . . .	22

---

2.3.7	Effects of interference on transmitted signal . . . . .	22
2.3.8	Ray tracing . . . . .	24
2.3.9	Empirical models . . . . .	24
2.3.10	Simplified path loss model . . . . .	25
2.3.11	Combined path loss and shadowing . . . . .	25
2.3.12	Statistical multipath channel models . . . . .	26
2.3.13	The hidden node problem . . . . .	26
2.4	Network performance measurements . . . . .	27
2.4.1	Network throughput, load and end-to-end delay . . . . .	28
2.4.2	Jitter or IP packet delay variation . . . . .	28
2.5	Other network coding schemes . . . . .	28
2.5.1	COPE . . . . .	29
2.5.2	Avalanche . . . . .	29
2.6	Simulation software . . . . .	30
2.7	Implementation software . . . . .	30
2.8	Conclusion . . . . .	31
<b>3</b>	<b>Network coding suitable topology identification</b>	<b>32</b>
3.1	Previous work done . . . . .	32
3.2	Network coding suitable topologies . . . . .	32
3.2.1	Linear topology . . . . .	33
3.2.2	Bow-tie topology . . . . .	34
3.2.3	Butterfly topology . . . . .	35
3.2.4	Extended butterfly topology . . . . .	35
3.2.5	Hybrid topologies . . . . .	36
3.3	Conclusion . . . . .	37

---

---

<b>4</b>	<b>Wireless communication rate, distance and area calculations</b>	<b>39</b>
4.1	Communication rates . . . . .	39
4.2	Communication distance calculations . . . . .	40
4.3	Communication area calculations . . . . .	45
4.3.1	Node locations within identified areas . . . . .	48
4.4	Hidden nodes . . . . .	49
4.5	Conclusion . . . . .	51
<b>5</b>	<b>The effects of interference</b>	<b>52</b>
5.1	Introduction . . . . .	52
5.2	Interference models . . . . .	53
5.2.1	Log-distance model - Indoor application . . . . .	53
5.2.2	Log-distance model - Outdoor application . . . . .	55
5.3	Comparison of log-distance model and free-space attenuation . . . . .	58
5.4	Signal attenuation caused by objects . . . . .	60
5.5	Conclusion . . . . .	61
<b>6</b>	<b>Simulation of identified topologies</b>	<b>62</b>
6.1	Introduction to OPNET . . . . .	62
6.1.1	OPNET radio transceiver pipeline . . . . .	63
6.1.2	ICI packets . . . . .	63
6.1.3	OPNET functions used . . . . .	65
6.2	OPNET standard models . . . . .	65
6.3	OPNET custom model . . . . .	66
6.3.1	Changes to standard model . . . . .	68
6.3.2	New processor: Network coding layer . . . . .	69

---

6.3.3	New sink processors . . . . .	72
6.3.4	New ICI for inter-layer communication . . . . .	74
6.3.5	Network coding algorithm . . . . .	74
6.4	Impact of algorithm design choices . . . . .	79
6.5	Comparison of simulation model to OPNET's standard model . . . . .	79
6.6	Conclusion . . . . .	80
<b>7</b>	<b>Simulation results</b>	<b>81</b>
7.1	Constant bit stream as source . . . . .	81
7.1.1	Bow-tie topology . . . . .	82
7.1.2	Butterfly topology . . . . .	83
7.1.3	Hybrid butterfly topology . . . . .	85
7.1.4	Summary of results - Constant bit stream . . . . .	87
7.1.5	Variation of packet delay - Constant bit stream . . . . .	87
7.2	Variable bit stream as source . . . . .	88
7.2.1	Bow-tie topology . . . . .	88
7.2.2	Butterfly topology . . . . .	91
7.2.3	Hybrid butterfly topology . . . . .	93
7.2.4	Summary of results - Variable bit stream . . . . .	97
7.2.5	Variation of packet delay - Variable bit stream . . . . .	97
7.2.6	Longer simulation time . . . . .	98
7.2.7	Variation of packet delay - Longer simulation time . . . . .	98
7.3	Conclusion . . . . .	99
<b>8</b>	<b>Implementation of identified topologies</b>	<b>100</b>
8.1	Introduction to Click modular router . . . . .	100

---

---

8.1.1	Elements and connections . . . . .	101
8.1.2	Router configurations . . . . .	102
8.2	Wireless testbed setup . . . . .	103
8.2.1	Testbed hardware architecture . . . . .	104
8.2.2	Testbed software architecture . . . . .	104
8.2.3	Testbed experimental parameters . . . . .	105
8.3	Created Click elements . . . . .	106
8.3.1	The "NetworkCoding" element . . . . .	107
8.3.2	The "HostEtherFilter2Addr" element . . . . .	108
8.4	Click setup for each node . . . . .	109
8.4.1	Source nodes . . . . .	109
8.4.2	Coding node . . . . .	110
8.4.3	Forwarding node . . . . .	110
8.4.4	Receiving nodes . . . . .	110
8.5	Conclusion . . . . .	112
<b>9</b>	<b>Implementation results</b>	<b>113</b>
9.1	Constant bit stream as source . . . . .	113
9.1.1	Bow-tie topology . . . . .	114
9.1.2	Butterfly topology . . . . .	115
9.1.3	Hybrid butterfly topology . . . . .	117
9.1.4	Summary of results - Constant bit stream . . . . .	119
9.1.5	Variation of packet delay . . . . .	119
9.2	Conclusion . . . . .	119
<b>10</b>	<b>Discussion of results</b>	<b>121</b>

---

---

10.1	Discussion of node placement calculation results . . . . .	121
10.2	Discussion of simulation results . . . . .	122
10.2.1	Constant bit stream . . . . .	122
10.2.2	Variable bit stream . . . . .	124
10.2.3	Variable bit stream - Longer simulation time . . . . .	125
10.3	Discussion of implementation results . . . . .	127
10.3.1	Comparing simulation and implementation results . . . . .	129
10.4	Conclusion . . . . .	130
<b>11</b>	<b>Conclusion</b>	<b>131</b>
11.1	Conclusion . . . . .	131
11.2	Validation and verification . . . . .	133
11.2.1	Comparing results to theory . . . . .	133
11.2.2	Comparison of coding scheme to other schemes . . . . .	133
11.2.3	Published papers . . . . .	134
11.3	Future work . . . . .	134
	<b>Bibliography</b>	<b>135</b>
	<b>Appendices</b>	
<b>A</b>	<b>Conference contributions from dissertation</b>	<b>139</b>
<b>B</b>	<b>OPNET's radio transceiver pipeline</b>	<b>140</b>
<b>C</b>	<b>OPNET's Kernel Procedures used</b>	<b>143</b>
<b>D</b>	<b>Click modular router scripts</b>	<b>146</b>
D.1	General Click script information . . . . .	146

---



---

D.2	Source nodes . . . . .	147
D.3	Coding node . . . . .	148
D.4	Forwarding node . . . . .	148
D.5	Receiving nodes . . . . .	149
<b>E</b>	<b>Click modular router implementation code</b>	<b>151</b>
<b>F</b>	<b>Data CD</b>	<b>157</b>

# List of Figures

1.1	Work breakdown structure. . . . .	5
2.1	Simple example of network coding . . . . .	14
2.2	Unicast network coding example . . . . .	15
2.3	Example of a network coding suitable topology. . . . .	16
2.4	Fresnel zone example . . . . .	21
2.5	Path loss, shadowing and multipath versus distance . . . . .	23
2.6	The hidden node problem example . . . . .	27
3.1	Linear network coding topology . . . . .	33
3.2	Bow-tie network coding topology . . . . .	34
3.3	Butterfly network coding topology . . . . .	35
3.4	Extended butterfly network coding topology . . . . .	36
3.5	Hybrid butterfly network coding topology . . . . .	37
4.1	Mathematical calculations: Transmit power . . . . .	43
4.2	Mathematical calculations: Receiver sensitivity . . . . .	43
4.3	Mathematical calculations: Communication distance . . . . .	44
4.4	Mathematical calculations: Fresnel radius . . . . .	44
4.5	Communication area: Linear topology . . . . .	45

---

4.6	Communication area: Bow-tie topology . . . . .	46
4.7	Communication area: Butterfly topology . . . . .	47
4.8	Communication area: Hybrid butterfly topology . . . . .	47
4.9	Node displacement: Bottom right node . . . . .	48
4.10	Node displacement: Top right node . . . . .	49
4.11	Hidden node problem: Bow-tie topology . . . . .	50
5.1	Indoor log-distance model communication distance . . . . .	54
5.2	Indoor log-distance model communication area for bow-tie topology . .	55
5.3	Outdoor log-distance model communication distance . . . . .	56
5.4	Outdoor log-distance model communication area for bow-tie topology .	57
5.5	Comparing log-distance and free-space signal attenuation models . . . .	59
5.6	Attenuation caused by objects . . . . .	60
5.7	Example of a wall in a bow-tie topology . . . . .	61
6.1	OPNET radio transceiver module overview . . . . .	64
6.2	OPNET workspace example . . . . .	65
6.3	OPNET standard WLAN station . . . . .	66
6.4	OPNET standard WLAN workstation . . . . .	67
6.5	OPNET custom WLAN network coding station . . . . .	69
6.6	OPNET custom WLAN network coding layer . . . . .	70
6.7	Custom network coding node attributes . . . . .	73
6.8	OPNET custom ICI packet . . . . .	74
6.9	Flowchart of coding node . . . . .	75
6.10	Buffer used in coding node . . . . .	76
6.11	Flowchart of decoding node . . . . .	77
6.12	Buffer used in decoding node . . . . .	78

---

---

6.13	Workspace: Comparing models . . . . .	80
7.1	Global load, bow-tie simulation . . . . .	82
7.2	Global end-to-end delay, bow-tie simulation . . . . .	83
7.3	Global media access delay, bow-tie simulation . . . . .	83
7.4	Global load, butterfly simulation . . . . .	84
7.5	Global end-to-end delay, butterfly simulation . . . . .	84
7.6	Global media access delay, butterfly simulation . . . . .	85
7.7	Global load, hybrid butterfly simulation . . . . .	85
7.8	Global end-to-end delay, hybrid butterfly simulation . . . . .	86
7.9	Global media access delay, hybrid butterfly simulation . . . . .	87
7.10	Source node one load, bow-tie simulation . . . . .	89
7.11	Source node two load, bow-tie simulation . . . . .	89
7.12	Global load, bow-tie simulation . . . . .	90
7.13	Global end-to-end delay, bow-tie simulation . . . . .	90
7.14	Global media access delay, bow-tie simulation . . . . .	91
7.15	Source node one load, butterfly simulation . . . . .	91
7.16	Source node two load, butterfly simulation . . . . .	92
7.17	Global load, butterfly simulation . . . . .	92
7.18	Global end-to-end delay, butterfly simulation . . . . .	93
7.19	Global media access delay, butterfly simulation . . . . .	94
7.20	Source node one load, hybrid butterfly simulation . . . . .	94
7.21	Source node two load, hybrid butterfly simulation . . . . .	95
7.22	Global load, hybrid butterfly simulation . . . . .	95
7.23	Global end-to-end delay, hybrid butterfly simulation . . . . .	96
7.24	Global media access delay, hybrid butterfly simulation . . . . .	96

---

8.1	Sample element . . . . .	102
8.2	Sample router configuration . . . . .	102
8.3	Testbed layout . . . . .	103
8.4	Network coding element . . . . .	108
8.5	MAC address filter element . . . . .	109
8.6	Source node . . . . .	110
8.7	Click setup for each node . . . . .	111
9.1	Global load, bow-tie implementation . . . . .	114
9.2	Global end-to-end delay, bow-tie implementation: NC . . . . .	115
9.3	Global end-to-end delay, bow-tie implementation: No NC . . . . .	115
9.4	Global load, butterfly implementation . . . . .	116
9.5	Global end-to-end delay, butterfly implementation: NC . . . . .	116
9.6	Global end-to-end delay, butterfly implementation: No NC . . . . .	117
9.7	Global load, hybrid butterfly implementation . . . . .	117
9.8	Global end-to-end delay, hybrid butterfly implementation: NC . . . . .	118
9.9	Global end-to-end delay, hybrid butterfly implementation: No NC . . . . .	118
10.1	Comparison of load reduction . . . . .	129

# List of Tables

2.1	IEEE 802.11 specifications . . . . .	17
4.1	Atheros AR5112 chipset specifications: Receiver sensitivity . . . . .	40
4.2	Atheros AR5112 chipset specifications: Transmit power . . . . .	40
5.1	Cisco Aironet hardware specifications (802.11b, 1Mbps) [1, 2, 3] . . . . .	58
7.1	Simulation results for a constant bit stream as source . . . . .	87
7.2	Variation of IP packet delay for a constant bit stream as source . . . . .	88
7.3	Simulation results for a variable bit stream as source . . . . .	97
7.4	Variation of IP packet delay for a variable bit stream as source . . . . .	97
7.5	Results: Variable bit stream - Longer simulation time . . . . .	98
7.6	Variation of packet delay: Variable bit stream - Longer simulation time . . . . .	99
8.1	Experimental parameters . . . . .	105
9.1	Implementation results for a constant bit stream as source . . . . .	119
9.2	Variation of IP packet delay for a constant bit stream as source . . . . .	119
10.1	Simulation results for a constant bit stream as source . . . . .	122
10.2	Results for a variable bit stream - Longer simulation time . . . . .	125
10.3	Implementation results for a constant bit stream as source . . . . .	127

# List of Acronyms

**ACI** Adjacent Channel Interference

**ARF** Auto Rate Fallback

**CCK** Complementary Code Keying

**CPU** Central Processing Unit

**CTS** Clear To Send

**DDR** Double Data Rate

**FSL** Free-Space Loss

**ICI** Interface Control Information

**IEEE** Institute of Electrical and Electronics Engineers

**IP** Internet Protocol

**KP** Kernel Procedure

**LLC** Logical Link Control

**LOS** Line Of Sight

**MAC** Medium Access Control

**MIMO** Multiple Input Multiple Output

**MTU** Maximum Transmission Unit

---

**OFDM** Orthogonal Frequency Division Multiplexing

**OSI** Open System Interconnection

**PAN** Personal Area Network

**PCI** Peripheral Component Interconnect

**PLCP** Physical Layer Convergence Protocol

**RAM** Random Access Memory

**RBAR** Receiver Based Auto Rate

**RF** Radio Frequency

**RTS** Request To Send

**RX** Receive

**SNR** Signal to Noise Ratio

**TX** Transmit

**WLAN** Wireless Local Area Network

**XOR** Exclusive OR



# Chapter 1

## Introduction

---

*This chapter serves as an introduction to the document. We give some background on wireless ad hoc networks and network coding. We then explain the objective of the research conducted, the issues to be addressed, the methodology followed and the validation and verification process used.*

---

### 1.1 Background

In today's modern society, communication forms a vital part of everyday living. The establishment and improvement of communication between electronic devices have become an interesting and extensive research area over the past few decades. Researchers constantly aim to improve communication, thus increasing the speed, reliability and security of electronic communication. Wireless communications is one of the most relevant components of the information and communications technology time period we are currently living in [4]. In our study, we will mainly focus on wireless ad hoc networks.

---

Wireless communication provides a means to exchange information on the move and to a large amount of people who does not have access to fixed line communication. The total number of cellular phones used worldwide, exceeded the number of land-lines in 2002 [5]. Every new notebook purchased is supplied with a wireless Ethernet card. Office buildings communicate between branches with the usage of wireless network links. Social interactions are changing rapidly and have captured the bulk of the research community's attention [5].

Within this wireless communicating world, wireless ad hoc networks are becoming increasingly popular. The advantages of implementing wireless ad hoc networks instead of wired networks or infrastructure based wireless networks include: increased mobility, lower installation complexity and ease of use. In developing countries, people can communicate in spite of a lack of infrastructure and the great distances that separate them by using multi-hop wireless ad hoc networks to cover a large area.

The disadvantage however, is a lower achievable transmission rate as a result of limited bandwidth and interference. Wireless ad hoc networks also require more complex routing algorithms, which in turn increases overhead. One documented attempt to counter these disadvantages, is network coding.

Network coding was first introduced by Ahlswede et al in 2000 [6]. The concept of network coding has been extensively researched in theory after it has been introduced, but very little research has been done on the practical implementation and feasibility of network coding. Two well-known applications of network coding have emerged in the form of COPE [7] and Avalanche [8], demonstrating the practical implementation of network coding.

This document describes the research done on the arrangement of nodes in wireless networks containing network coding suitable topologies. Previous research and recommendations on "Using Topological Information in Opportunistic Network Coding" [9] are used. The network coding suitable topologies identified are mathematically analysed, simulated and implemented to find the boundaries on the placement

of nodes and evaluate the practicality of implementing network coding within these topologies. This research will serve as a step towards the successful implementation of network coding in a practical wireless ad hoc network, containing network coding suitable topologies.

## 1.2 Objective

The objective of our study is to define the physical dimensions and optimal node placement of various wireless network topologies that are suitable for the implementation of linear network coding. These definitions are based on the distance at which communication can reliably be executed, the speed at which communication can take place and other aspects including interference. A mathematical model that can calculate these dimensions and node positions is needed to establish a theoretical reference which simulations can be based on. The mathematical model must be able to describe practical situations accurately, therefore the usage of a practical path-loss model is necessary. The resulting topologies must be simulated, implemented and analysed to evaluate the practicality thereof and to ensure that network coding suitable topologies can be exploited successfully in reality.

## 1.3 Issues to be addressed

The main issues addressed in this study are:

- **Literature study:** This includes a comprehensive study on the field of wireless networks, network coding and interference involved in the implementation of wireless networks.
- **Network coding suitable topology identification:** Topologies suitable for network coding are identified and investigated to ensure that it can be practically

implemented in wireless networks.

- **Distance vs. transmission rate calculations:** The identified topologies are analysed to determine the communication distance that can be achieved with commercially available hardware and the achievable communication rates at the various distances.
- **Effects of interference and possible solutions:** The effects of interference on a physical network are investigated and possible solutions or remedies identified.
- **Simulate and implement topologies:** The identified network coding suitable topologies are simulated and implemented with the determined physical node positions and their boundaries as parameters.
- **Compare and discuss results:** The results obtained from mathematical analysis, simulation and implementation are discussed.

A detailed work breakdown structure is depicted in figure 1.1.

## 1.4 Methodology

The methodology followed in this dissertation is described in the next subsections.

### 1.4.1 Literature study

A literature study is done on various fields including network coding, available wireless network technology, routing protocols and interference in wireless networks. Current certified Medium Access Control (MAC) methods used in commercial hardware are studied to ensure optimal performance is obtained in the given application domain. Advances in new hardware and software development such as the implementation of Multiple Input Multiple Output (MIMO) techniques to enhance throughput in wireless networks, are also studied. Network coding is studied to gain knowledge of the



Figure 1.1: Work breakdown structure.

concept; the basic idea behind network coding, how to implement it in a physical environment, the mathematics involved in the field and the identification of network coding suitable topologies.

### 1.4.2 Network coding suitable topology identification

The work of Grobler et al's work, entitled "Using Topological Information in Opportunistic Network Coding" [9, 10] is used as a basis for the topology identification stage. The authors recommend the use of topological information to identify opportunities for the implementation of network coding in wireless networks. They do this by looking for "known" network coding suitable topologies within a larger network. The method they used comprised of the following five steps:

1. Select a network coding suitable topology of which the gain and capacity is known (a bow-tie or butterfly topology for instance).
2. Derive the connection matrix of the larger network from a suitable distance vector routing algorithm.
3. Search the larger network matrix for the known topology structure.
4. Implement network coding at the appropriate nodes.
5. Re-iterate steps (3) and (4) after a routing update.

### 1.4.3 Distance vs. transmission rate calculations

The mathematics required to determine reliable communication distance of each node is studied and used. Documentation on the average commercially available hardware is studied to define the distance at which reliable communication can theoretically take place at each different packet transfer speed setting. This differs for each possible MAC standard and hardware alternative.

#### 1.4.4 Effects of interference and possible solutions

Interference from other electro-magnetic wave sources and obstacles in the radio propagation path, have an extreme impact on the quality of wireless communication between nodes. Interference can cause packet loss and induce errors in transmitted data. In severe cases, communication can be completely lost. It is therefore important to be aware of possible interference sources and the impact they may have on communication. Possible solutions or remedies are identified. This knowledge is important during the simulation and implementation phases of the project to ensure realistic and usable results are obtained.

#### 1.4.5 Simulate topologies

OPNET modeler wireless suite will be used to simulate all the identified network coding suitable topologies.

During the simulation phase, network coding will not be implemented in its final working state where it can be used in current working wireless ad hoc networks, as this is out of the scope of the project. Packets are transferred from source nodes to the destination nodes as well as the “smart” nodes. At the “smart” nodes, packets are combined with a simple Exclusive OR (XOR) operation and sent to destination nodes. This will be a simple simulation of network coding and will test the practicality of the identified network coding suitable topologies with nodes located within their identified boundaries.

#### 1.4.6 Implement topologies

The last phase of this project is to implement the identified wireless network topologies. Click modular router will be used to replace the standard network stack of a wireless node with a custom network stack implementing network coding and decod-

ing.

The algorithm used during the implementation phase will be the same as the one used during the simulation phase to enable the useful comparison between the simulation and implementation results.

### **1.4.7 Discussion and conclusion**

Theoretical, simulation and implementation results on the limitations and practicality of the implementation of network coding in the identified topologies, will be discussed.

## **1.5 Beneficiaries**

The TeleNet research group of the North-West University is the primary beneficiary of this project. The research will extend the group's domain knowledge.

The research was presented at national and international conferences, contributing to the network coding research field.

## **1.6 Validation and verification**

The validation and verification procedures used in this dissertation, are described in the next two subsections.

### **1.6.1 Validation**

The research done in this project is supplementary to previous research done by the TeleNet group and is therefore contributing to the development of the group.



The research process, described in the next section, will be followed to ensure the production of a structured dissertation, that is acceptable and of value to the research and other communities.

A work in progress as well as a two full peer reviewed conference papers were submitted and accepted for publication (refer to appendix A). The submission and publication of papers is important in any research project to ensure that the research done is up to standard, is not duplicating other research and that the global community find the research done of value.

### **1.6.2 Verification**

The results obtained from the mathematical model, simulations and implementation phases will be compared. The comparison of the various results will be used to verify and conclude the research done in this project. Simulation and implementation results will be compared to other network coding implementations including COPE [7] and Avalanche [8].

## **1.7 The research process**

According to [11] “Research is a systematic way of asking questions in order to expand the knowledge base and to solve problems.” The purpose of research can be divided into different categories. This includes [11]:

- The advancement of knowledge without specific benefits,
- The gathering of knowledge intended for a specific application,
- Experimental development of new products, systems or services.

In this research project, we investigate and experiment with already existing concepts with the aim of developing a communication system using these concepts, therefore the research conducted in this project falls under the third category.

It is important for all researchers to answer the following questions [11]:

- What must be researched?
- Why is the particular research important?
- How will the research process proceed?
- When will research take place?
- What resources are needed for the research?.

The first four questions were answered in sections 1.1, 1.2, 1.3 and 1.4. The resources needed for the simulations are discussed in chapter 6 and the resources needed for the implementation phase are discussed in chapter 8.

The research process must be followed to ensure valid and applicable results are obtained. The elements of the research process, described next, never follow in a linear fashion. For example the literature survey is often done throughout the research process and the research problem may change as more insight is gained. The work plan may also be constantly updated [11].

### **1.7.1 Literature surveys**

Why the research is necessary is further identified. The specific research field must be studied through literature surveys. The research problem must be defined and the specific methodology to be used, identified. This was the first step taken during our research.

### **1.7.2 Research proposal**

The problem statement (what), background (why) and research method (how) must be discussed in this document. Our research proposal was submitted to a committee and accepted.

### **1.7.3 Research work plan**

The research method, the how question, is expanded upon. Further details includes a time schedule, personnel and equipment needed as well as a budget (payment schedule).

### **1.7.4 Conducting research**

Data is acquired, stored and interpreted. Further details may include a case study, testing, experimental research, modelling, simulations, etc. For this phase, we made use of mathematical modelling, simulations and experimental implementation.

### **1.7.5 Research document and publications**

Published articles and dissertations must be used to make the research results known.

## **1.8 Document structure**

The report documenting this project will have the following structure:

- Chapter 1 - Introduction to document
- Chapter 2 - An in depth literature study

- Chapter 3 - Identification of network coding suitable topologies
- Chapter 4 - Wireless communication rate, distance and area calculations
- Chapter 5 - The effects of interference on topologies identified in chapter 3
- Chapter 6 - Simulation of the identified topologies
- Chapter 7 - Present simulation results
- Chapter 8 - Implementation of identified topologies
- Chapter 9 - Present implementation results
- Chapter 10 - Discussion and comparison of simulation and implementation results
- Chapter 11 - Conclusion and recommendations

# Chapter 2

## Literature study

---

*In this chapter, we present a literature study on network coding, wireless networks, network performance measurements, other network coding schemes, the simulation and implementation software needed. In the wireless network section, we discuss Institute of Electrical and Electronics Engineers (IEEE) standards and other concepts including interference, communication rates, achievable communication distances and the hidden node problem. This chapter provides the theoretical principals on which the research conducted in this project is based.*

---

### 2.1 Network coding

Network coding refers to the implementation of coding methods to utilise network connections more efficiently. In an article: "Network Coding: An Instant Primer" [12], it is stated that: "With network coding, intermediate nodes may send out packets that are linear combinations of previously received information." By sending a combined packet in a single time slot, throughput and thus efficiency of the network is enhanced at the cost of having intelligent nodes capable of combining and decoding packets.

---

### 2.1.1 Unicast sessions

Network coding can be used in different applications for different purposes. It is crucial to remember that the transmission medium is shared (the air) and thus to prevent collisions, only one node can send a packet at a given moment in a given communication channel. In [12], a simple example of two nodes exchanging information is given. Node A and node C needs to exchange information through an intermediate node B. With the use of traditional communication methods, node A would send a packet to node B in one time slot, node C would send a packet to node B in the next time slot and node B would send each individual packet at the next two different time slots to their respective destinations. With network coding implemented, the third and fourth transmission can be combined into a single transmission, which can then be decoded at the destinations. This respective processes is illustrated in figure 2.1.

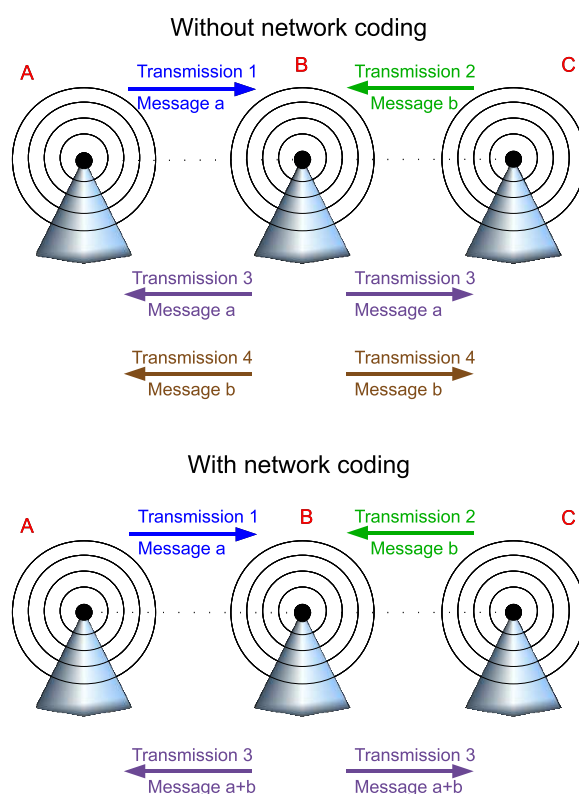


Figure 2.1: Simple example of network coding

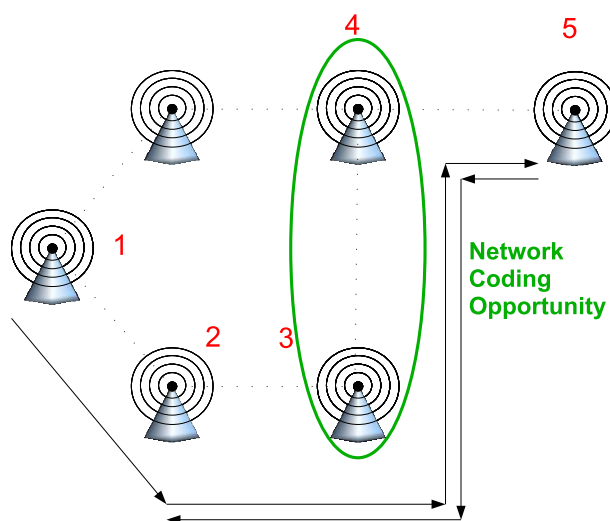


Figure 2.2: Unicast network coding example

Another example of the implementation of network coding, is the application in a network consisting of more than three nodes where two unicast sessions exist simultaneously [13]. Refer to figure 2.2. As can be seen, node 1 needs to transmit a data frame to node 5 and node 5 needs to transmit a data frame to node 2. Due to range limitations of the nodes, transmission of the frames must be done through several intermediate nodes. The transmission paths of the two sessions overlap, creating an opportunity for the implementation of network coding.

### 2.1.2 Multicast sessions

By combining several transmissions into one, the transmission medium is used more efficiently, increasing the total achievable throughput. By implementing this principle in existing wireless networks by finding and using network coding suitable topologies, total throughput of the network can be increased. An example of such a network coding suitable topology is shown in figure 2.3. In this example node A must send a data frame to node D and E, together with node B that must send a data frame to node D and E. Due to communication range limitations, some communication between node A and E as well as communication between node B and D, must be done through node

C. Packets arriving at C from A and B, can be combined into a single network coded packet and sent to D and E simultaneously. Two multicast sessions, each requiring 2 transmissions, were combined to reduce the total number of transmissions from 4 to 3.

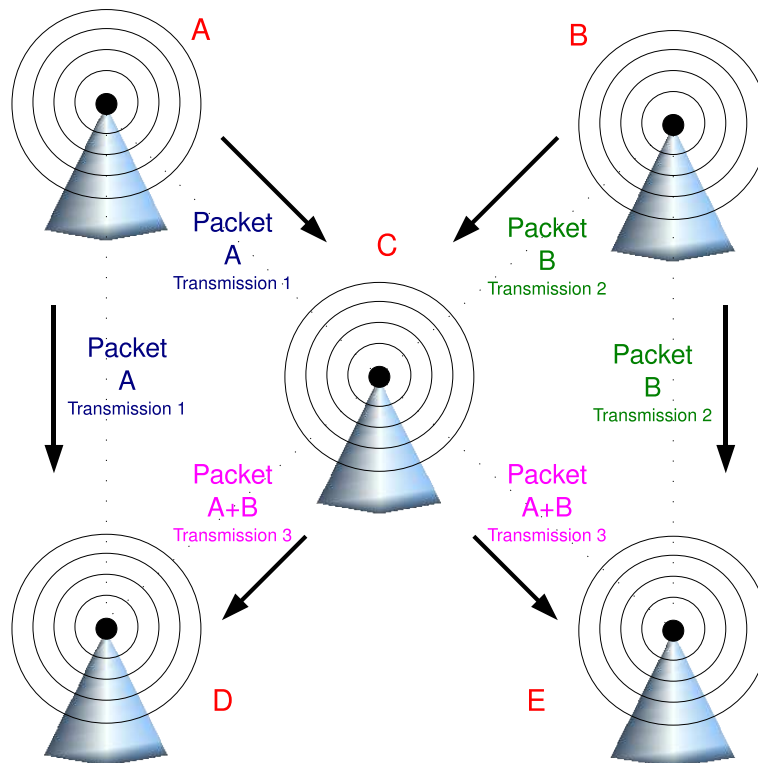


Figure 2.3: Example of a network coding suitable topology.

## 2.2 Wireless networks: IEEE 802.11 standards

Wireless networks can utilize different protocols and MAC methods. In this project, the IEEE 802.11a/b/g MAC methods are used in the calculation, simulation and implementation phases. These methods are popular and hardware utilizing them are commercially available. The official specifications of these standards are shown in table 2.1 [14].



Table 2.1: IEEE 802.11 specifications

	802.11b	802.11a	802.11g
Standard approved:	July 1999	July 1999	June 2003
Maximum data rate:	11 Mbps	54 Mbps	54 Mbps
Modulation:	CCK	OFDM	OFDM and CCK
Data rates:	1, 2, 5.5, 11 Mbps	6, 9, 12, 18, 24, 36, 48, 54 Mbps	CCK: 1, 2, 5.5, 11 OFDM: 6, 9, 12, 18, 24, 36, 48, 54 Mbps
Frequencies:	2.4 - 2.497 GHz	5.15 - 5.35 GHz 5.425 - 5.675 GHz 5.725 - 5.875 GHz	2.4 - 2.497 GHz

### 2.2.1 IEEE 802.11a

Refer to table 2.1. The 802.11a standard operates in the 5 GHz frequency band and was developed for high bandwidth applications with transmission speeds of up to 54 Mbps using Orthogonal Frequency Division Multiplexing (OFDM) modulation on up to 12 discrete channels. The fact that this standard operates in the 5 GHz spectrum causes communication distance to be limited and employment costs to be relatively high compared to other standards. This has resulted in limited market acceptance [14].

### 2.2.2 IEEE 802.11b

Refer to table 2.1. The 802.11b standard operates in the 2.4 GHz frequency band with transmission speeds of up to 11 Mbps using Complementary Code Keying (CCK) modulation on up to 3 nonoverlapping channels or up to 13 overlapping channels on the lowest two rates. With its affordable implementation prices, relatively long communication distance and acceptable transmission speeds, this standard was the market leader before the development of IEEE 802.11g [14].

### 2.2.3 IEEE 802.11g

Refer to table 2.1. The 802.11g standard operates in the 2.4 GHz frequency band with transmission speeds of up to 54 Mbps using OFDM modulation on up to 3 non-overlapping channels or 13 overlapping channels. This standard satisfies the market's bandwidth needs globally and economically. It is backward compatible with the 802.11b standard, making the transition from the 802.11b standard to the 802.11g standard more economically viable [14].

### 2.2.4 IEEE 802.11n

The new IEEE wireless standard, 802.11n, was completed in late 2009. Hardware utilizing this standard operates in the 2.4 and 5 GHz bands and delivers transfer speeds of up to 300 Mbps using new technologies which includes:

- MIMO
- Packet aggregation
- Channel bonding (40 MHz channels)

This standard is backwards compatible with the 802.11 a, b and g standards, making the transition to this standard more economically viable [15].

## 2.3 Wireless networks: Concepts

Interference phenomena and the effects thereof on wireless communication, is discussed in the next subsections. An overview of mathematical models used to calculate the effects of different types of interference, is also given.

### 2.3.1 Interference

In wireless networks, the transmission medium (air) is shared by all communicating parties. In a multi-radio environment, simultaneous communication can occur when different frequencies are used which do not overlap in the frequency spectrum, therefore the 2.4 GHz and 5 GHz open spectrums are divided into multiple communication channels. When implementing simultaneous communication methods, which make use of multiple radios to enhance throughput of a network, Adjacent Channel Interference (ACI) can occur. According to [16], there are mainly two types of ACI that occur, namely Receive (RX)-Transmit (TX) ACI and TX-TX ACI:

- RX-TX interference occurs when one node consisting of multiple radios transmits on one channel and due to imperfect filters in the hardware, outputs a part of the transmitted power on a adjacent channel on which another radio is receiving, interfering with the data being received [16].
- TX-TX interference occurs when a transmitting radio outputs a part of its transmitted power on a adjacent channel which is mistakenly recognized as an active carrier on that channel, causing another transmitter which is transmitting on this channel, to back off and not transmit [16].

It should therefore be noted that interference between multiple radios, transmitting and receiving on different channels, can occur and should be taken into account when designing and implementing wireless networks.

Interference in wireless networks can also be caused by other equipment such as cordless and cellular phones as well as microwaves. Such sources of interference must be identified and, if possible, removed or the effects lessened for a wireless network to function effectively.

Other phenomena such as multi-path interference, which is especially prominent when implementing wireless networks in buildings, degrades the quality of wireless data links. These interference phenomena are discussed in a later section.

### 2.3.2 Fresnel zone

It is important to have a clear communication zone between transmitting and receiving antennas to maximize transmission distance and speed. This zone is referred to as a Fresnel zone. Obstacles in the Fresnel zone will cause transmitted waves to be reflected towards the receiving antenna causing multipath interference, which is described in the next section.

The transmitted wave can be visualized as a series of concentric ellipsoids that grows wider as transmission distance increase. As it is stated in [17], “The term Fresnel zone defines the shape of these ellipses as a circular zone with a radius such that the distance from a point on this circle to the receiving point is some multiple of a half wavelength longer than the direct path.” The radius of the Fresnel zone for communicating terminals can be calculated at any distance in between. The radius is the highest in the centre and is calculated by the equation [17]:

$$r = 17.32 \sqrt{\frac{D}{4f}} \quad (2.1)$$

where

- $r$  is the radius in meters,
- $D$  is the total distance in km
- $f$  is the carrier frequency in GHz.

Figure 2.4 shows a visual representation of a Fresnel zone.

### 2.3.3 Multipath interference

Obstacles in the Fresnel zone gives rise to a reflection which also propagates towards the receiver. The reflection will take a longer time to reach the receiver because of the longer distance travelled and is therefore out of phase with the original signal. If

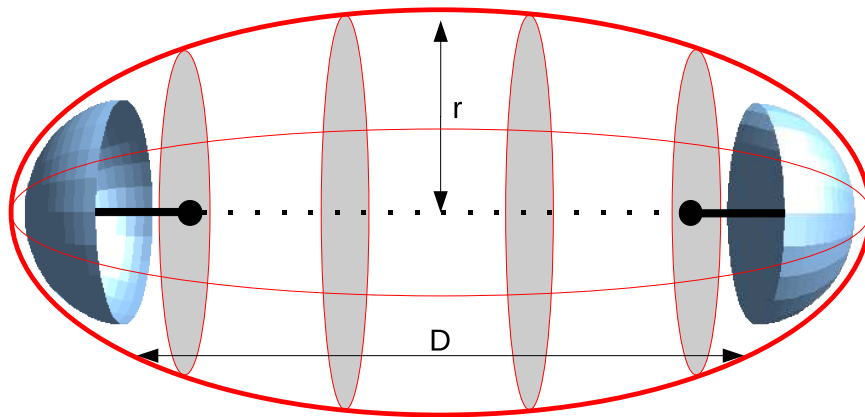


Figure 2.4: Fresnel zone example

the reflection is 180 degrees out of phase at the receiver, the two signals will cancel, otherwise the reflection will result in distortion of the original signal.

Multipath interference will result in signal loss and an increase in error rate due to the increased uncertainty of the actual phase and amplitude state of the signal.[17]

### 2.3.4 Fading

Fading is caused by the movement of obstacles in the Fresnel zone. Such obstacles can include cars, people or trees. This will result in random variations of the frequency response and amplitude of the received signal. Fading is categorized into two groups:

- Log normal or flat fading: Under normal conditions this fading results in the variation in amplitude of the received signal as a constant, while its spectral characteristics remains unchanged.
- Frequency selective fading: This fading causes distortion in the received signal, no longer as an entity, but only certain frequencies are affected. This causes “holes” in the received signal at certain frequencies.

Fading is a localized phenomenon, which means that fading effects differ over distances of about half a wave length. This causes signal strength to be affected by very small variations in antenna placement. Fading is usually caused by multipath interference [17].

### **2.3.5 Shadowing**

The obstruction of a transmitted signal by random obstacles is called shadowing. Obstacles in the transmission path cause large amounts of attenuation through absorption, reflection, scattering and diffraction [18]. This includes objects like steel doors or structural walls which cause "shadows" behind them, resulting in a loss of signal strength or even blockage of the signal. Shadowing is mainly a concern in highly mobile nodes where communication can be lost if a node moves into a communication shadow area [17].

### **2.3.6 Communication transmission rate and distance**

Increasing the communication transmission rate of a Wireless Local Area Network (WLAN) (or any other Radio Frequency (RF)) equipment, causes the receiver's sensitivity and the transmitter's output power to decrease, causing a decrease in the reliable communication distance achievable between nodes. All of these factors must be taken into account when calculating the theoretical communication distance at a certain transfer rate [17].

### **2.3.7 Effects of interference on transmitted signal**

In wireless communication, transmission power is lost as the waves propagate through space. This is a result of a number of phenomena including free-space attenuation, multipath interference, fading and shadowing. Signals are reflected, diffracted and

scattered by objects in the propagation path. These signals arrive at the receiver and are summed with the Line Of Sight (LOS) signal, producing distortion in the received signal relative to the transmitted signal. The LOS signal can also be attenuated by objects located directly within its path [18]. Communication distance and transfer rate are influenced by all of these factors.

As an example of how a transmitted RF signal is influenced by the occurrence of the different interference effects described, consider the case where the achievable communication distance is calculated by various methods. First, only take the regular path loss formula into account. Secondly, include the effect of shadowing into the equation. Lastly, include the effect of multi-path interference as well. To compare the different effects delivered by these interference phenomena, the results of the previous three steps were plotted on the same axis in [18] and are shown in figure 2.5. It can be seen that more random and complex results are obtained when all of these described interference effects are taken into account.

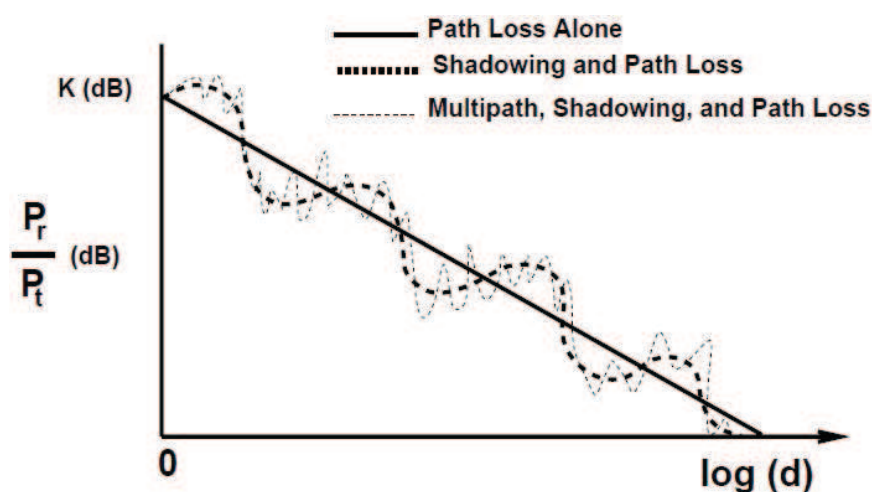


Figure 2.5: Path loss, shadowing and multipath versus distance

Consider the simplest case, where the transmitter and receiver has a LOS communication space with no obstacles in between, the attenuation between the transmitter and receiver can be estimated with the Free-Space Loss (FSL) equation [17]:

$$L_{fs} = 32.44 + 20 \log d + 20 \log f \quad (2.2)$$

where

- $L_{fs}$  is the free-space attenuation in dB,
- $d$  is the communication distance in km and
- $f$  is the carrier frequency in MHz.

The attenuation calculated by this equation is a result of the spreading of radio waves as it propagates away from its source. An increase in attenuation causes reliable communication distance to decrease. As a rule of thumb, an increase of transmit power of 6 dB will double communication distance and vice versa [17]. It should be noted that this equation will give an attenuation value which excludes phenomena like fading, shadowing, multipath effects, etc. There are many path loss models available using empirical or statistical methods to describe and predict path loss more accurately. A brief description of such models follow.

### 2.3.8 Ray tracing

With the use of ray tracing models, it is assumed that there is a finite number of reflectors in the propagation path. It is further assumed that the location and dielectric properties of these objects are known. The details of the multipath effects can then be solved using Maxwell's equations [19]. The effects of reflection, diffraction and scattering are approximated using simple geometric equations [18].

### 2.3.9 Empirical models

Empirical models are based on empirical measurements over a certain distance, within a given frequency range for a particular geographical area for both indoor and outdoor applications. These models include the Okumura model [20] and the Hata model [21].



Both of these models are only valid for transmission in the 150-1500 MHz frequency ranges which are not applicable to this project and thus not discussed [18].

### 2.3.10 Simplified path loss model

If a simplified model, which captures the essence of signal propagation is needed, the following model can be used [18]:

$$P_r = P_t + K - 10\gamma \log_{10} \left[ \frac{d}{d_0} \right] \quad (2.3)$$

where

- $P_r$  and  $P_t$  is the receive and transmit power in dBm respectively,
- $K$  is a unitless constant that depends on antenna characteristics,
- $\gamma$  is the path loss exponent,
- $d_0$  is a reference distance for the antenna far field and
- $d$  is the communication distance.

The values for  $K$ ,  $d_0$  and  $\gamma$  can be obtained analytically or empirically.

### 2.3.11 Combined path loss and shadowing

Random objects in the signal path will give rise to random variations in the received signal power. The location, size and properties of blocking objects are generally unknown, so statistical models must be used to characterize the imposed attenuation. The log-normal shadowing model is used to estimate this variation in signal attenuation and is given by [18]:

$$p(\psi) = \frac{\xi}{\sqrt{2\pi}\sigma_{\psi dB}\psi} \exp \left[ -\frac{(10 \log_{10} \psi - \mu_{\psi dB})^2}{2\sigma_{\psi dB}^2} \right], \psi > 0 \quad (2.4)$$

where

- $\psi = \frac{P_t}{P_r}$ , the ratio of the transmit-to-receive power,
- $\xi = \frac{10}{\ln 10}$ ,
- $\mu_{\psi_{dB}}$  is the mean of  $\psi_{dB} = 10 \log_{10} \psi$  and
- $\sigma_{\psi_{dB}}$  is the standard deviation of  $\psi_{dB}$ .

Equation 2.4 can be combined with equation 2.3 to give [18]:

$$\frac{P_r}{P_t} dB = 10 \log_{10} K - 10\gamma \log_{10} \frac{d}{d_0} - \psi_{dB} \quad (2.5)$$

where

- $\psi_{dB}$  is a Gauss-distributed random variable with mean zero variance  $\sigma_{\psi_{dB}}^2$ .

### 2.3.12 Statistical multipath channel models

Ray-tracing models can be used to characterise multipath effects in deterministic channels but if the number of multipath components is large or if the propagation environment as well as the properties of the objects in the propagation environment are unknown, then statistical multipath models must be used [18]. These models can get quite complex. For an overview of statistical multipath channel models, refer to [18].

### 2.3.13 The hidden node problem

In the physical implementation and topology design of wireless networks, phenomena such as the hidden node problem must be taken into account to ensure maximum throughput is achieved. The hidden node problem occurs when two transmitting nodes are nearby each other, but cannot receive the physical header of transmitted packets, thus they are unaware of each other's transmission. A receiver node can be located between these transmitting nodes receiving simultaneous, and thus interfering, transmissions from the transmitting nodes. The scenario is shown in figure 2.6.

To avoid the occurrence of this situation, the IEEE 802.11 standard relies on a collision avoidance mechanism called the Request To Send (RTS)/Clear To Send (CTS) mechanism. Two communicating nodes can reserve a channel by exchanging two short control packets namely RTS and CTS. Any other node which successfully receives a RTS or CTS packet cannot transmit for a period of time declared in the Physical Layer Convergence Protocol (PLCP) length field. If this handshake between two nodes is successful, all other nodes within range of the transmitting and receiving node will be silent for the duration of the data and acknowledgment transmission.

The exchange of these packets will increase overhead substantially and reduce data throughput. It is thus desirable to design a wireless network topology which does not have any hidden nodes. This will permit the network designer to omit the usage of the RTS/CTS mechanism [22].

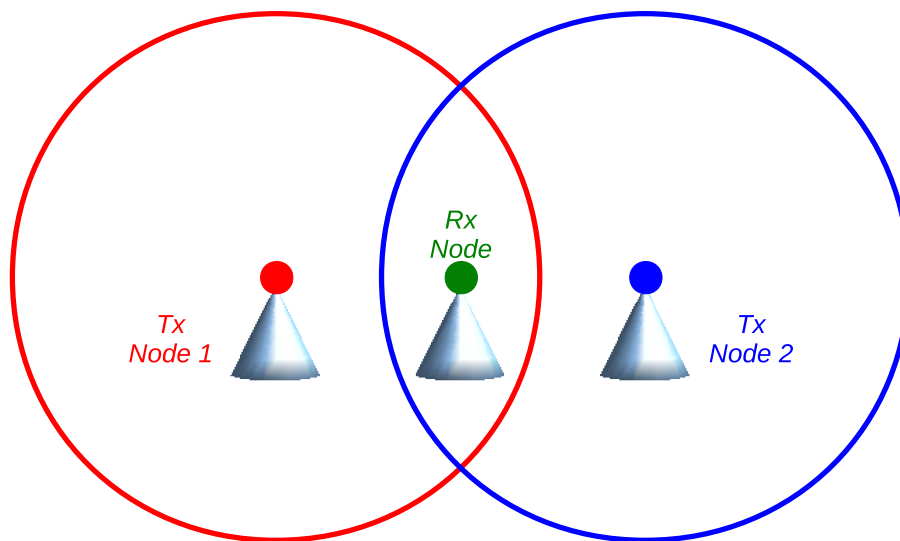


Figure 2.6: The hidden node problem example

## 2.4 Network performance measurements

In order to evaluate any changes made to the standard IEEE 802.11 standard, we need to measure the performance of the network. There are many network performance

measurement methods available. We will mainly focus on the methods discussed in the following subsections:

### 2.4.1 Network throughput, load and end-to-end delay

OPNET modeler will be used for measuring network throughput, load and end-to-end delay. For this reason, we will describe the network performance measurement terms as given in OPNET modeler's documentation [23]. OPNET describes network throughput in the wireless environment as the average number of bits successfully received by the receiver channel per second and forwarded to the higher layers by the WLAN MAC of nodes in the network. The network load is described as the total data traffic, in bits per second, received by the entire network from the higher layers of the MACs that is accepted and queued for transmission. The end-to-end delay measurement includes the medium access delay at the source MAC, reception of all the individual fragments and the transfer of frames via an access point if one is used [23].

### 2.4.2 Jitter or IP packet delay variation

The variation of Internet Protocol (IP) packet delay within a stream of packets, referred to as jitter, can be defined for a selected pair of packets in the stream, going from measurement point MP1 to measurement point MP2. The IP packet delay variation is the difference between the one-way delay of the selected packets, measured at MP1 and MP2 [24].

## 2.5 Other network coding schemes

Some researchers have already successfully implemented network coding in various schemes suitable for certain scenarios. Examples of two such projects are COPE [7] and Avalanche [8].

### 2.5.1 COPE

This research focused on the implementation of network coding for unicast traffic in wireless mesh networks. The researchers developed COPE [7] from theory and implemented the system on a 20-node wireless testbed. Implementation results indicated that COPE increases throughput with gains depending on traffic patterns, congestion levels and the transport protocol used [7].

COPE inserts a new network coding layer between the MAC and IP layers of the Open System Interconnection (OSI) protocol stack. Nodes are set to promiscuous mode, which enables them to forward all overheard packets to the new network coding layer and not only the packets destined for that particular node. Packets are stored for a limited time period by the new network coding layer. Nodes periodically send broadcast reports to neighbours, notifying them of the overheard packets in its buffer. Packets are then network coded together at relay nodes and sent to destination nodes based on COPE's coding algorithm, which identifies "good" coding opportunities [7].

### 2.5.2 Avalanche

This network coding scheme is specifically designed for the distribution of large files over a large, unstructured peer-to-peer network. Large files are split into smaller blocks which are distributed through the network. The distribution of these small blocks creates a perfect opportunity for the implementation of network coding. An example of such an end-system cooperative architecture, is the well-known peer-to-peer network BitTorrent. Through simulations it is shown that file download time is expected to improve by 20% to 30% with network coding implemented throughout the network compared to coding done at the server only and by more than two to three times compared to sending un-encoded information. The robustness of the network is improved and the entering and leaving of nodes is better handled with the system implemented [8].

## 2.6 Simulation software

Software that can be used for simulation purposes includes MATLAB and OPNET modeler wireless suite.

- **MATLAB:** This Mathworks product is a powerful technical computing language which can be used for various simulations and calculations. Theoretical principles of wireless communication can be coded into MATLAB and used to give theoretical results.
- **OPNET:** OPNET is a company providing software for managing applications and networks. OPNET modeler can be used to analyse and design communication networks, devices, protocols, and applications. The OPNET modeler wireless suite edition can be used to simulate mobile devices, including cellular, mobile ad hoc, WLAN, Personal Area Network (PAN) and satellite communication. We can therefore use this software to simulate the implementation of network coding in the different wireless network coding suitable topologies studied in this project.

## 2.7 Implementation software

Click modular router is a toolkit for writing modular network packet processors. Each packet processor or router is highly flexible and configurable [25, 26]. Click modular router can be used on wireless nodes to implement network coding and decoding. Click runs in the Linux environment and replaces or runs alongside the default Linux network stack.

## 2.8 Conclusion

In this chapter, we explained the concept of network coding, described the relevant IEEE 802.11 standards and discussed different wireless communication concepts and performance measurements. We gave a brief overview of other network coding schemes and the software needed to simulate and implement our network coding scheme. Special attention was given to interference types and different models used to predict the effect of interference on wireless communication. In the next chapter, we will describe the different topologies suitable for the implementation of our network coding scheme.

## Chapter 3

# Network coding suitable topology identification

---

*In this chapter, we identify different network coding suitable topologies for wireless ad hoc networks. The layout and workings of the identified topologies are discussed.*

---

### 3.1 Previous work done

In [9], different network coding suitable topologies and methods on finding these topologies in larger mesh networks, are identified.

### 3.2 Network coding suitable topologies

Certain stationary wireless network topologies are more suitable for the implementation of network coding than others. If a node knows where it is located within the



network coding suitable topology, network coding can be implemented without the exchange of control packets.

In the next subsections, network coding suitable topologies will be identified and discussed.

### 3.2.1 Linear topology

The linear topology consists of three nodes. Nodes are arranged in a linear fashion as shown in figure 3.1. Due to the physical constraints on the communication range of the wireless nodes, nodes A and C must communicate with each other through the intermediate node B. When nodes A and C exchange information with each other, node B can monitor the packets that it forwards. When a network coding opportunity arise, node B can combine two packets, one from node A and the other from node C, and send it to both the destination nodes in one transmission. The destination nodes can decode the encoded packet by using their own packet that was sent previously. The topology and flow of information is depicted in figure 3.1.

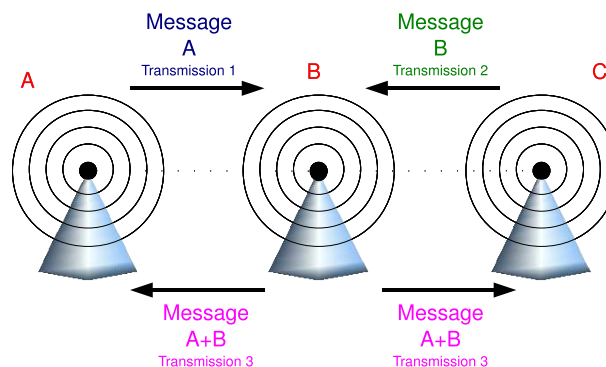


Figure 3.1: Linear network coding topology

### 3.2.2 Bow-tie topology

The bow-tie topology consists of five nodes. The topology and flow of information, is depicted in figure 3.2. Nodes A and D and nodes B and E must be in one another's communication range respectively. Cross communication (between nodes A and E and nodes B and D respectively) uses the intermediate node C as a relay node. This is due to the physical constraints on the communication range of the wireless nodes. Observe the case for the transmission of information from the top nodes (nodes A and B) to the bottom nodes (nodes D and E) in a multicast fashion. Transmission of information is conducted through the intermediate node C. Node C (also referred to as the "smart" or "coding" node) can then monitor the packets sent to it and identify network coding opportunities. Coded packets can then be decoded at the destination nodes by using the packet first received from the respective top node.

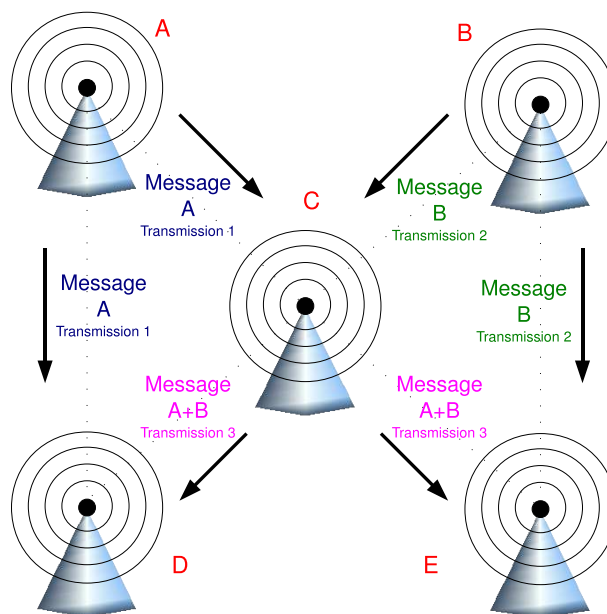


Figure 3.2: Bow-tie network coding topology

### 3.2.3 Butterfly topology

The butterfly topology consists of six nodes. The topology is depicted in figure 3.3. The flow of information in the topology is almost the same as in the bow-tie topology. The only difference is the introduction of another intermediate node below the "smart" or "coding" node. The second intermediate node's sole purpose is to forward the coded packets to the destination nodes.

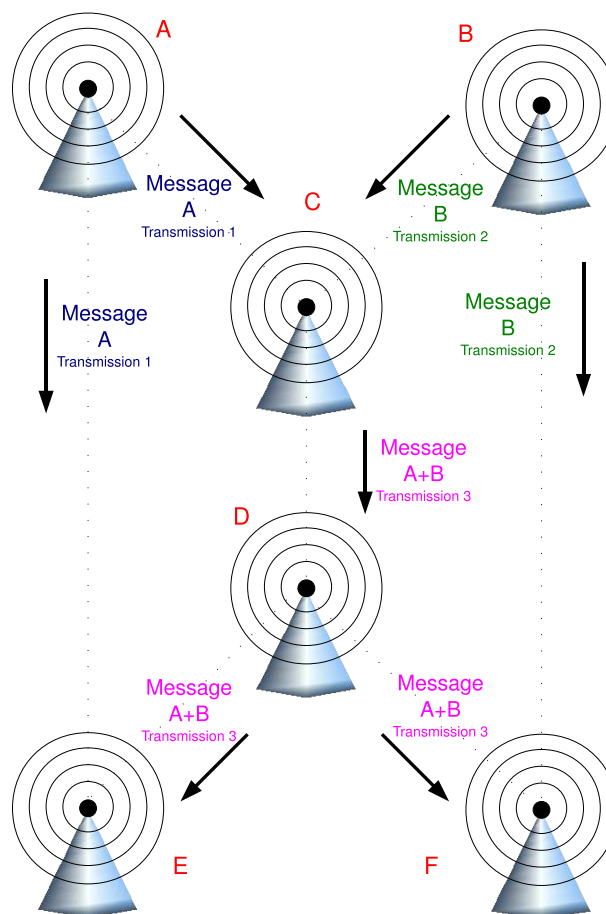


Figure 3.3: Butterfly network coding topology

### 3.2.4 Extended butterfly topology

The extended butterfly topology consists of seven nodes. The flow of information in the topology is almost the same as in the butterfly topology. The only difference is the

introduction of another intermediate node. The third intermediate node's sole purpose is to act as another relay node to forward the coded packets to the destination nodes. The topology and flow of information is depicted in figure 3.4. It will be revealed in a later chapter that the realization of this topology is impractical. The topology is included because of its usage by other researchers in theoretical network coding studies.

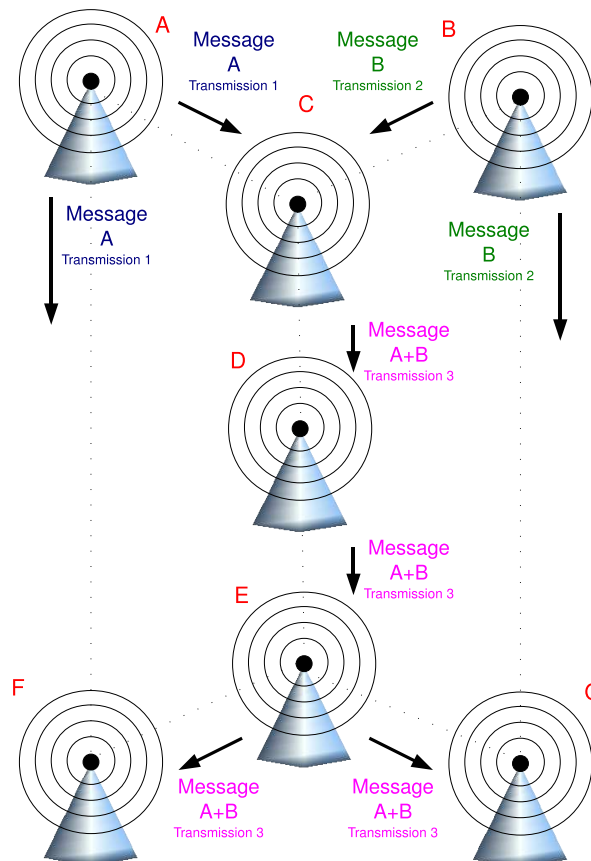


Figure 3.4: Extended butterfly network coding topology

### 3.2.5 Hybrid topologies

Hybrid topologies can consist of any combination of the other identified topologies. The topology discussed here is a combination of the bow-tie and butterfly topologies and consists of six nodes. The flow of information in the topology is almost the same

as in the bow-tie topology. The only two differences is the introduction of another intermediate node below the "smart" or "coding" node and the variation in placement of the two destination nodes. One of the destination nodes is placed where the destination nodes of a butterfly topology would conventionally be located and the other where a bow-tie topology's destination nodes would be located. The topology and flow of information is depicted in figure 3.5.

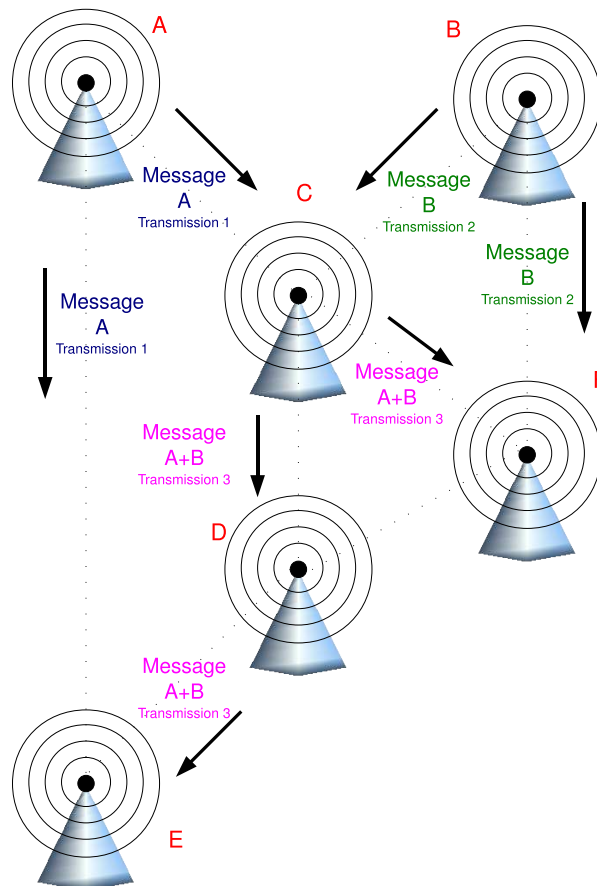


Figure 3.5: Hybrid butterfly network coding topology

### 3.3 Conclusion

In this chapter, we discussed the different wireless network topologies in which network coding can be implemented. We identified the position and function of each node

within a topology. In the next chapter, we will use mathematical modeling to calculate the exact dimensions of these identified topologies under certain circumstances.

# Chapter 4

## Wireless communication rate, distance and area calculations

---

*In this chapter, we discuss the creation of a mathematical model used to calculate the theoretical communication distance achievable between communicating nodes at a certain communication rate. The model produces a graphical output of network coding suitable topologies, which we use to identify possible locations of communicating nodes. We also discuss the hidden node problem and the effects thereof.*

---

### 4.1 Communication rates

Practical wireless communication hardware have different communication rate settings for each of the different available IEEE standards (IEEE 802.11 a/b/g), because the faster information is transmitted, the more likely the wireless channel is to introduce errors during transmission. The error rate is a function of the communication speed, distance and interference present. To achieve the maximum performance in

varying conditions, devices must adapt their transmission rate dynamically. There are a number of algorithms available that calculates the best communication rate for a specific channel dynamically. These algorithms include the Auto Rate Fallback (ARF) and the Receiver Based Auto Rate (RBAR) [27]. The communication rate selection algorithm used on specific hardware is mostly vendor dependent.

When communication rate is increased, the receiver sensitivity decreases as well as the transmitter output power, therefore the reliable communication distance will decrease as the communication rate is increased. The specific sensitivity and power values are hardware dependent. In our calculations, all the initial information used is based on the popular Atheros AR5112 [28] chipset and shown in tables 4.1 and 4.2.

Table 4.1: Atheros AR5112 chipset specifications: Receiver sensitivity

Rate (Mbps):	1	2	5.5	6	9	11	12	18	24	36	48	54
802.11b (dBm):	-95	-94	-92			-90						
802.11a (dBm):				-88	-87		-85	-83	-80	-75	-73	-71
802.11g (dBm):				-90	-89		-87	-85	-82	-79	-76	-74

Table 4.2: Atheros AR5112 chipset specifications: Transmit power

Rate (Mbps):	1	2	5.5	6	9	11	12	18	24	36	48	54
802.11b (dBm):	18	18	18			18						
802.11a (dBm):				17	16		16	15	15	14	14	13
802.11g (dBm):				18	18		18	17	17	17	16	16

## 4.2 Communication distance calculations

For the simplest case, consider the FSL equation [17]:

$$L_{fs} = 32.44 + 20 \log d + 20 \log f \quad (4.1)$$

where

- $L_{fs}$  is the free-space attenuation in dB,



- $d$  is the communication distance in km and
- $f$  is the carrier frequency in MHz.

From this equation we can get an estimate of the theoretically possible communication distance that can be achieved by wireless nodes.

To calculate the power at the receiver we can use the simple formula [17]:

$$P_r = P_t + G_t + G_r - L_{fs} \quad (4.2)$$

where

- $P_r$  is the power at the receiver in dB,
- $P_t$  is the power at the transmitter in dB,
- $G_t$  is the transmitter antenna gain in dB,
- $G_r$  is the receiver antenna gain in dB and
- $L_{fs}$  is the free-space attenuation in dB

When combining the two equations we get:

$$P_r = P_t + G_t + G_r - 32.44 - 20 \log d - 20 \log f \quad (4.3)$$

We need the distance that each node can communicate. Rewrite the above equation:

$$d = 10^{(P_t + G_t + G_r - 32.44 - P_r - 20 \log f) / 20} \quad (4.4)$$

Consider the case where the receiver and transmitter have an antenna gain of unity. Then the above equation simplifies to:

$$d = 10^{(P_t - 32.44 - P_r - 20 \log f) / 20} \quad (4.5)$$

Now we can calculate the theoretical communication distances for nodes communicating with the different IEEE 802.11 standards at various communication rates by substituting the values in tables 4.1 and 4.2 into the above equation.

To calculate the Fresnel radius at the highest point (located at the centre between the transmitter and receiver) we use the equation [17]:

$$r = 17.32 \sqrt{\frac{D}{4f}} \quad (4.6)$$

where

- $r$  is the radius in meters,
- $D$  is the total distance in km
- $f$  is the carrier frequency in GHz.

We can now substitute the calculated communication distance for each IEEE 802.11 standard at the various communication rates into  $D$  in the above equation.

In all the calculations,  $f$  is replaced with 2.4 GHz and 5 GHz, which is the standard communication frequency band for the IEEE 802.11 b/g standards and the IEEE 802.11 a standard respectively.

The MATLAB output of the all the calculations are shown in figures 4.1, 4.2, 4.3, 4.4.

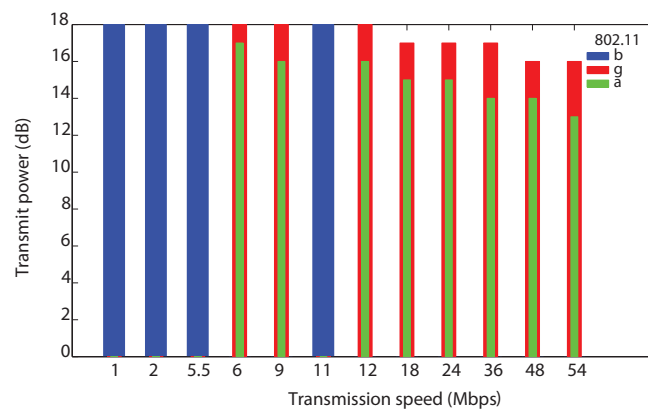


Figure 4.1: Mathematical calculations: Transmit power

Figure 4.1 gives a graphical representation of the maximum transmit power when different communication rates are used for the 802.11 a, b and g standards. The values in the figure are the same as given in table 4.2 and are based on the Atheros AR5112 [28] chipset.

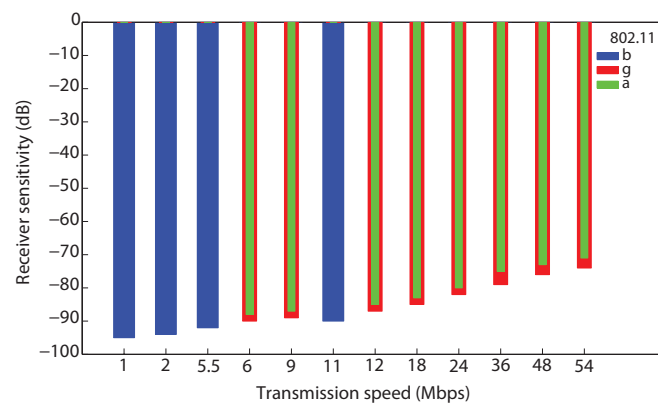


Figure 4.2: Mathematical calculations: Receiver sensitivity

Figure 4.2 gives a graphical representation of the receiver sensitivity when different communication rates are used for the 802.11 a, b and g standards. The values in the figure are the same as given in table 4.1 and are based on the Atheros AR5112 [28] chipset.

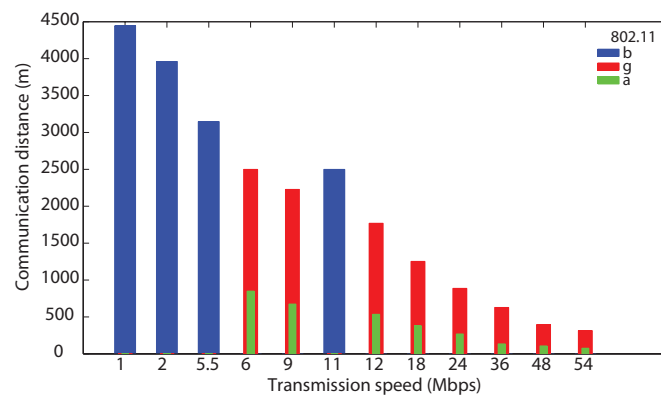


Figure 4.3: Mathematical calculations: Communication distance

Figure 4.3 gives a graphical representation of the maximum theoretical communication distance as calculated by the mathematical model using the FSL equation. The theoretical communication distance are given for each of the possible communication rate settings when the 802.11 a, b or g standards are used.

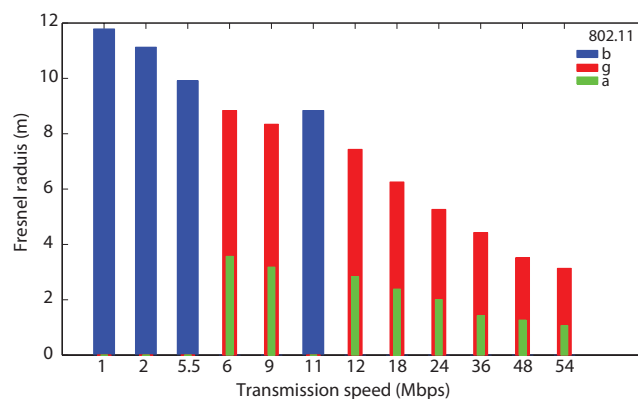


Figure 4.4: Mathematical calculations: Fresnel radius

Figure 4.4 gives a graphical representation of the Fresnel radius when different communication rates are used for the 802.11 a, b and g standards. This output is used to give an estimate of the area between the transmitter and receiver that must be free of obstacles to prevent interference degrading the communication signal. If there are any obstacles in this area between communicating nodes, the calculated communication distances displayed in figure 4.3, would not be achievable.

### 4.3 Communication area calculations

Consider the use of omni-directional antennas with no gain. The results of the above section can be used to determine the communication area for each node in the different network coding suitable topologies. That is, the area in which each node can be located to enable the network coding topology to function successfully. An example of the output figures that can be generated by this developed MATLAB model, is shown in figures 4.5, 4.6, 4.7 and 4.8. In these examples, all nodes used the IEEE 802.11g standard, communicating at a fixed rate of 54Mbps. At this communication rate, the hidden node problem will not have an impact on the performance of the identified topologies. The reason for this will be described later in this chapter.

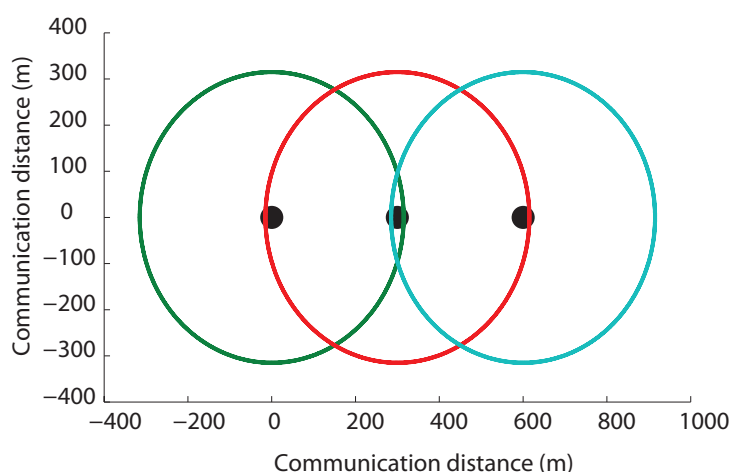


Figure 4.5: Communication area: Linear topology

Figure 4.5 gives the graphical output, as calculated by the mathematical model, for the linear topology. The communication circles of each node are displayed. Communication areas are formed where the relevant circles overlap. These communication areas can be used to determine the possible location of each node within the topology.

Figure 4.6 gives the graphical output, as calculated by the mathematical model, for the bow-tie topology. The communication circles of each node are displayed. As with the linear topology, this output can be used to determine the possible location of each node

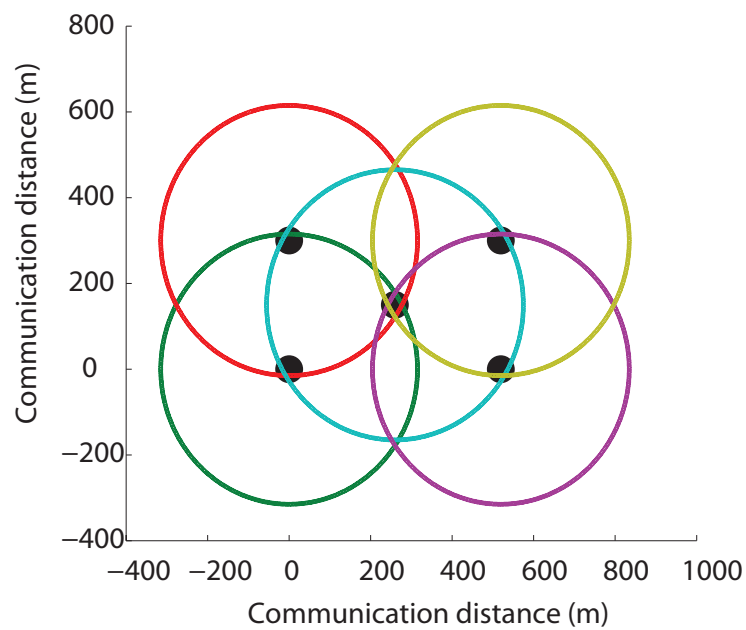


Figure 4.6: Communication area: Bow-tie topology

within the topology.

Figure 4.7 gives the graphical output, as calculated by the mathematical model, for the butterfly topology. The communication circles of each node are displayed. As with the other topologies, this output can be used to determine the possible location of each node within the topology.

Figure 4.8 gives the graphical output, as calculated by the mathematical model, for the hybrid butterfly topology. The communication circles of each node are displayed. As with the other topologies, this output can be used to determine the possible location of each node within the topology.

In these examples, the communication area is not defined as the coverage area only, but also the flexibility of the placement of nodes in an area. Thus, the bow-tie topology covers the same area as the butterfly topology, but the coding node cannot move in the bow-tie topology, whereas in the butterfly topology, they can move.

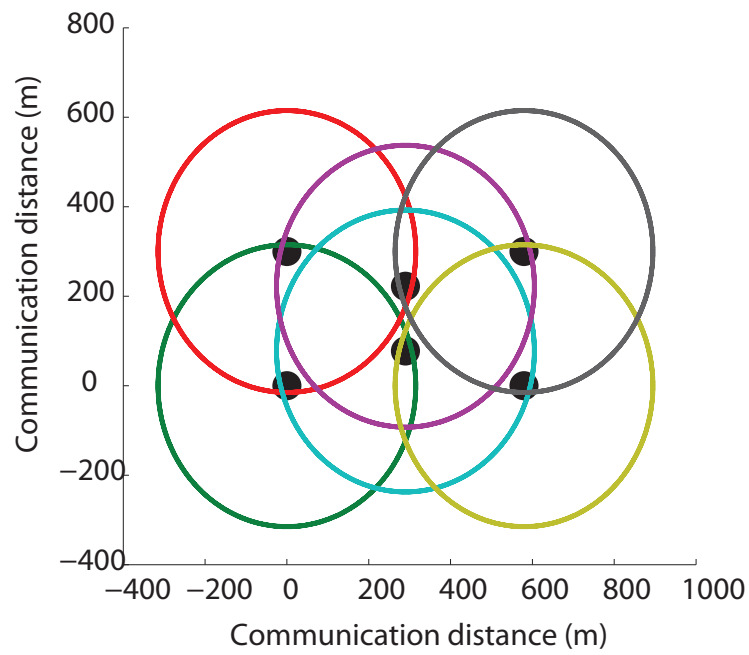


Figure 4.7: Communication area: Butterfly topology

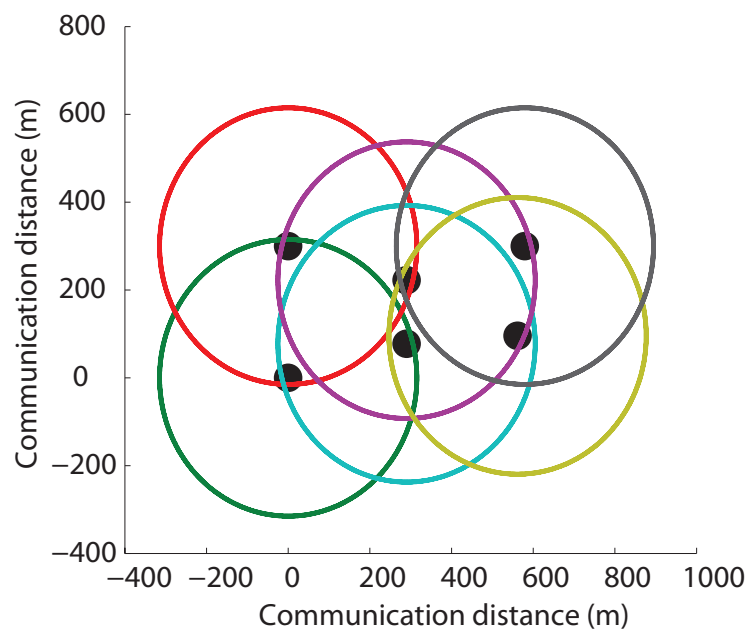


Figure 4.8: Communication area: Hybrid butterfly topology

### 4.3.1 Node locations within identified areas

Nodes must be located within an area where the necessary communication circles overlap. An example of such an identified area is shown in figure 4.9. Consider the bottom right node in a bow-tie topology. This node must be able to communicate with the top right and the centre (intermediate) node. The area in which these conditions are satisfied, is shown in figure 4.9. This area identification can be done for any of the network coding suitable topologies by using figures 4.5, 4.6, 4.7, 4.8.

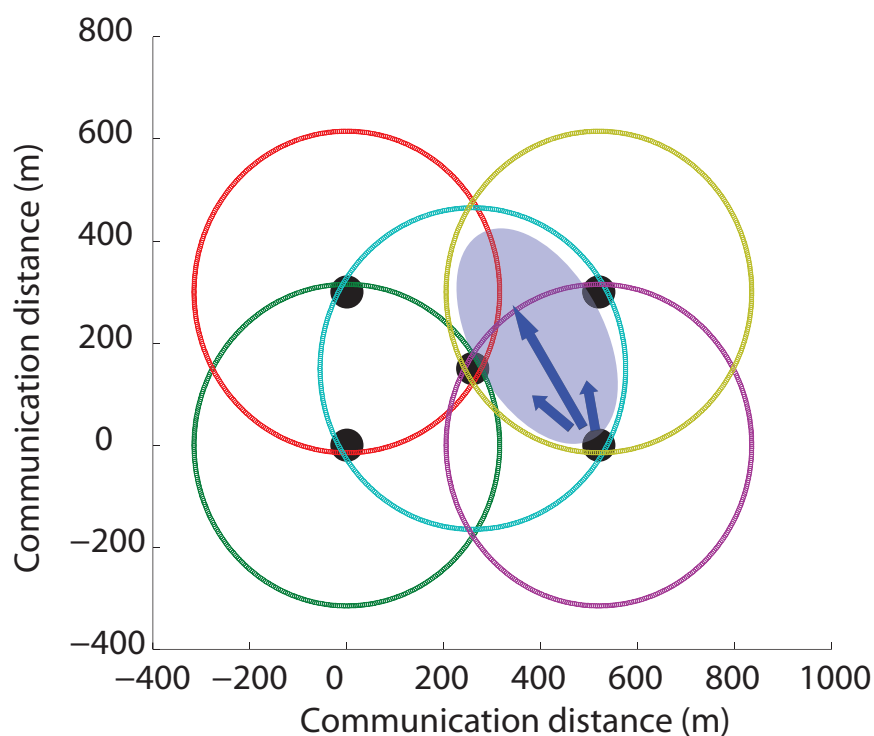


Figure 4.9: Node displacement: Bottom right node

When nodes are displaced from their default location, the communication circles are also displaced. New communication areas are then formed. When more than one node is displaced, the process of finding the new communication areas is least complicated when nodes are moved one at a time. This makes it possible to determine the new areas in which the other nodes can be moved around in. An example of the situation is shown in figure 4.10. The bottom right node is displaced from its default location



to another position within its communication area as was shown in figure 4.9. This displacement caused the top right node's communication area to change as shown in figure 4.10. Now this top node can be moved around within the new formed communication area.

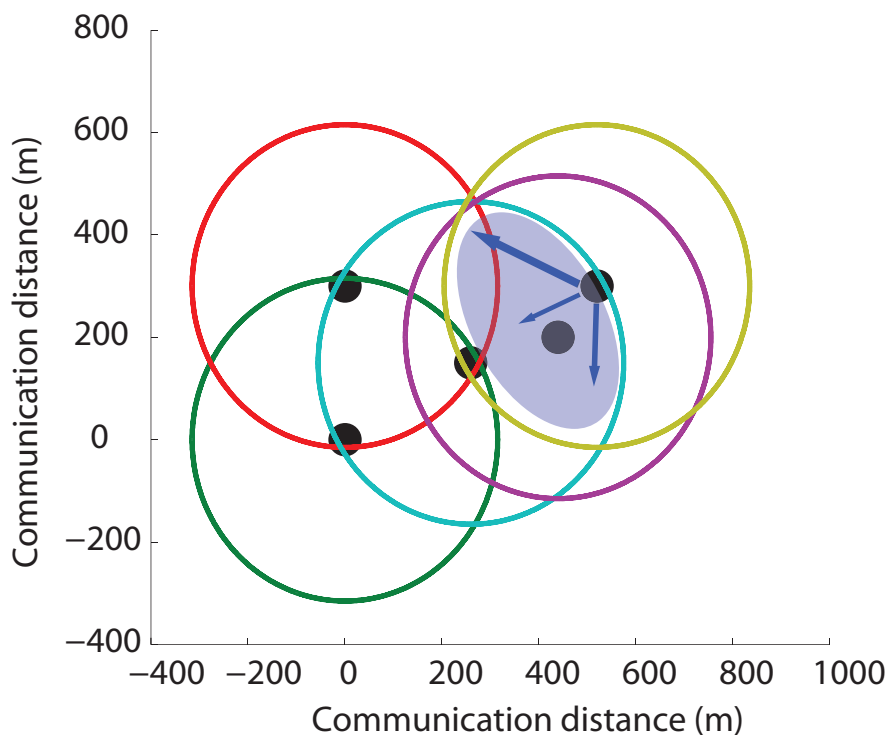


Figure 4.10: Node displacement: Top right node

## 4.4 Hidden nodes

To avoid the occurrence of hidden nodes, all the nodes in the identified topologies must be able to receive the physical header (PLCP header) of a packet being transmitted by any other node. When using the IEEE 802.11 standard, the PLCP header must be transmitted at 1 or 2 Mbps while the rest of the packet can be transmitted at higher transmission rates [29]. The consequence of this property is that the maximum reception range of the PLCP header can be larger than that of the data part.

When nodes are arranged in close proximity to each other in such a way that the transmission of packets can successfully occur at rates of 12 Mbps or above using the IEEE 802.11g standard and 6 Mbps or above using the IEEE 802.11a standard, the hidden node problem will not occur. This can be seen in figure 4.11 where each node in the topology transmit data at a high transmission rate. It can be seen that the packet header can be transmitted successfully to all the other nodes when sent at 1 Mb/s or 2 Mb/s, as indicated by the larger two circles. However, if any of the topologies grow beyond this point and use one of the lower transmission rates, including 11 Mb/s and lower when using the IEEE 802.11b standard or 9 Mb/s and lower using the IEEE 802.11g standard, hidden nodes may arise and the RTS/CTS mechanism must be enabled to avoid unnecessary collisions.

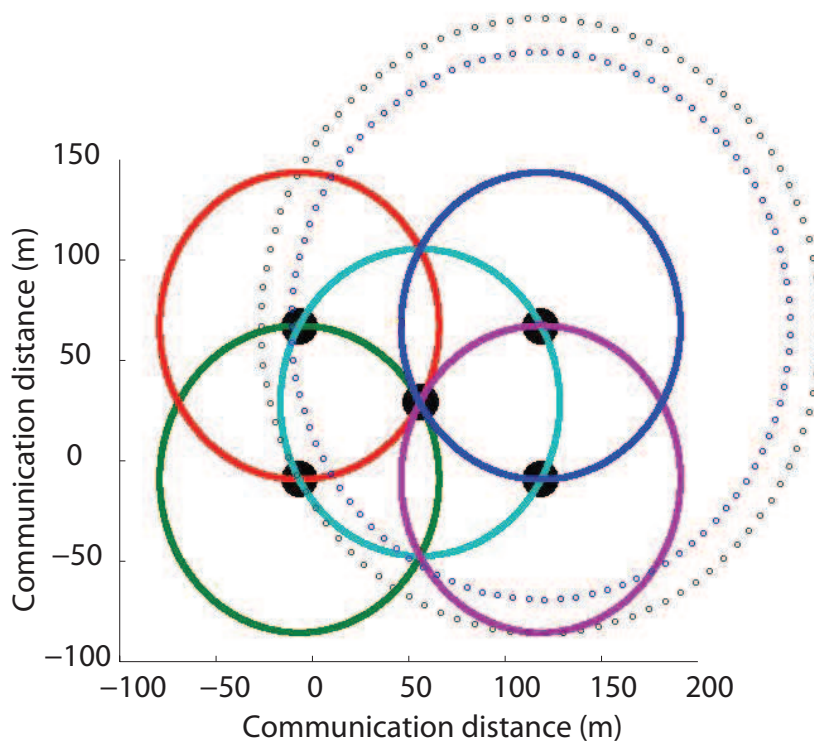


Figure 4.11: Hidden node problem: Bow-tie topology

## **4.5 Conclusion**

In this chapter, we used a mathematical model to calculate the communication distances and possible placement areas of nodes located within a network coding suitable topology. The mathematical model used the popular Atheros AR5112 chipset as a base for calculations and assumed that the transmitter and receiver had no obstructions between them. In the next chapter, we will discuss the effects interference has on the identified network coding suitable topologies.

# Chapter 5

## The effects of interference

---

*In this chapter, we study the effects of interference caused by objects in the transmission signal's propagation path. We modify our mathematical model by replacing the free-space loss equation with a more realistic signal propagation prediction method. We then compare the theoretical communication distance predictions as delivered by the free-space loss equation and the more realistic signal propagation prediction method. Lastly, we look at the effects of different materials on signal propagation.*

---

### 5.1 Introduction

Interference decreases the signal quality of wireless transmissions. This in turn affects the communication range of wireless nodes to such an extent that the communication distances calculated in chapter 4, which only takes free space path loss into consideration, will not be realizable in the real world. The different types of interferences experienced in practical wireless networks includes multipath interference, fading and shadowing. All of these interference effects were described in chapter 2.

---

## 5.2 Interference models

To model the effect of interference on a transmitted signal, different models have been created by various researchers. These models can be divided into different categories, discussed in chapter 2, which includes ray tracing, empirical and statistical models. In this project, we use the log-distance model [30] to predict the transmission distance that can be achieved by commercially available hardware supporting the IEEE 802.11 standard. This model is simplistic and effective in modeling signal propagation over a wide range of frequencies [1]. The model uses empirical measurements and statistical methods to predict the attenuation a communication signal might experience in the real world and takes log-normal shadowing into account [1]. The measurements differ from indoor and outdoor applications due to the increased number of obstacles located between the transmitter and receiver in a typical indoor environment compared to a typical outdoor environment.

### 5.2.1 Log-distance model - Indoor application

The log-distance path loss model for indoor applications is given by [30]:

$$Pr(d) = \overline{Pr}(d) + X_\sigma Pr(d) = Pr_0 - 10\alpha \log(d) + X_\sigma \quad (5.1)$$

where

- $Pr_0$  is the signal strength 1 meter from the transmitter
- $\alpha$  is known as the path loss exponent
- $X_\sigma$  represents a Gaussian random variable with zero mean and standard deviation of  $\sigma$  dB
- $Pr(d)$  represents the mean (expected) signal strength  $d$  meters from the transmitter

In [1], this model was empirically verified with the usage of four Cisco Aironet 1200 wireless access points, operating at 2.4 GHz (IEEE 802.11b). The receiver used was a Cisco Aironet 340 PCMCIA card. Experiments yielded  $\alpha = 4.02$  and  $\sigma = 7.36$  dB. The log-distance model is a simplistic model causing the attenuation induced by obstacles between the receiver and transmitter to be aggregated into  $\alpha$ .

This model is used to determine more realistic communication distances assuming Cisco Aironet 1200 access points are used for mesh communication, operating at 2.4 GHz IEEE 802.11b 1Mbps and by using the above values for  $\alpha$ ,  $\sigma$  and  $Pr_0 = -20$  dB as given in [1]. The result of these calculations are shown in figure 5.1 for repeated invocations of the model.

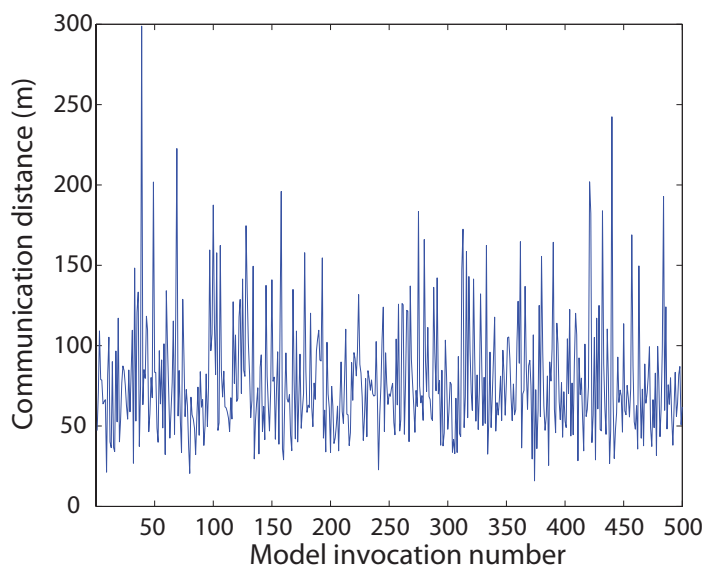


Figure 5.1: Indoor log-distance model communication distance

The log-distance model was invoked 500 times and the average of the statistical communication distance results (70m) was used in the mathematical model created in chapter 4. The output of the mathematical model is shown in figure 5.2 for a bow-tie application.

By comparing figure 5.2 with figure 4.6 from the previous chapter, we can see that the mathematical model now predicts a considerably smaller overall topology. This is be-

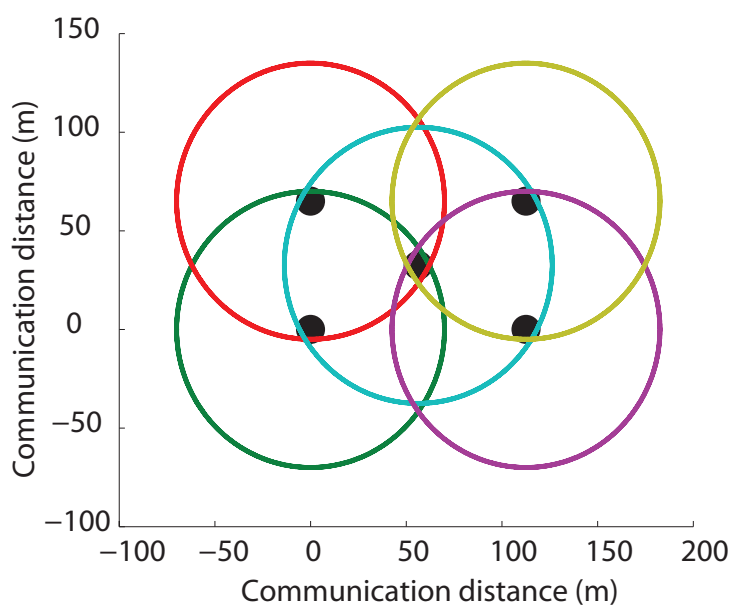


Figure 5.2: Indoor log-distance model communication area for bow-tie topology

cause of the reduction in predicted communication distance between communicating nodes. The butterfly and hybrid butterfly experiences the same reduction in overall topology size.

### 5.2.2 Log-distance model - Outdoor application

The log-distance path loss model for outdoor applications is similar to the model used for indoor applications and is given by [30, 31]:

$$Pr(d) = Pr_0 - 10\alpha \log(d) - W + X_\sigma \quad (5.2)$$

where

- $Pr_0$  is the signal strength 1 meter from the transmitter,
- $\alpha$  is known as the path loss exponent,
- $X_\sigma$  represents a Gaussian random variable with zero mean and standard deviation of  $\sigma$  dB

- $P_r(d)$  represents the mean (expected) signal strength  $d$  meters from the transmitter
- $W$  is the wall attenuation factor in dB

In [1], this model was empirically verified with the usage of four Cisco Aironet 1200 wireless access points, operating at 2.4 GHz (IEEE 802.11b). The receiver used was a Cisco Aironet 340 PCMCIA card. Experiments yielded  $W = 4.8\text{dB}$ ,  $\alpha = 3.32$  and  $\sigma = 3.1$  dB.

This model is used to determine more realistic communication distances assuming Cisco Aironet 1200 access points are used for mesh communication, operating at 2.4 GHz IEEE 802.11b 1Mbps and by using the above values for  $W$ ,  $\alpha$ ,  $\sigma$  and  $P_{r_0} = -20$  dB as given in [1]. The result of these calculations are shown in figure 5.3 for repeated invocations of the model.

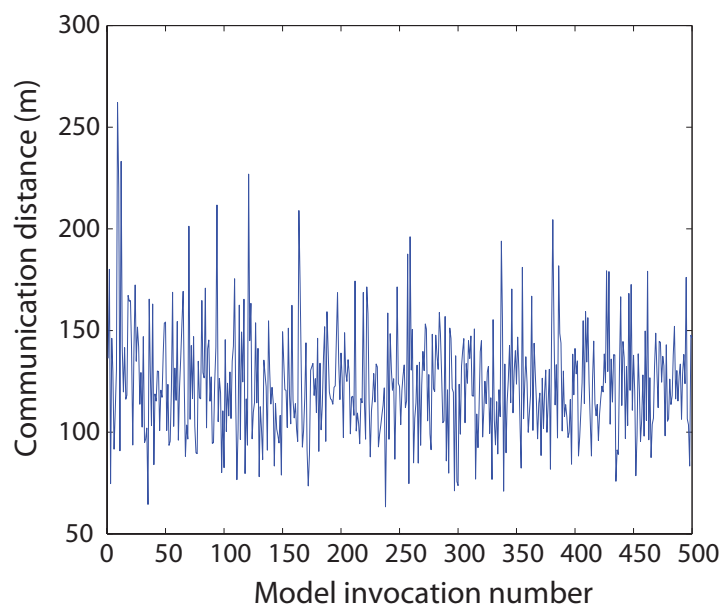


Figure 5.3: Outdoor log-distance model communication distance

The log-distance model was invoked 500 times and the average of the statistical communication distance results (120m) was used in the mathematical model created in



chapter 4. The output of the mathematical model is shown in figure 5.4 for a bow-tie application.

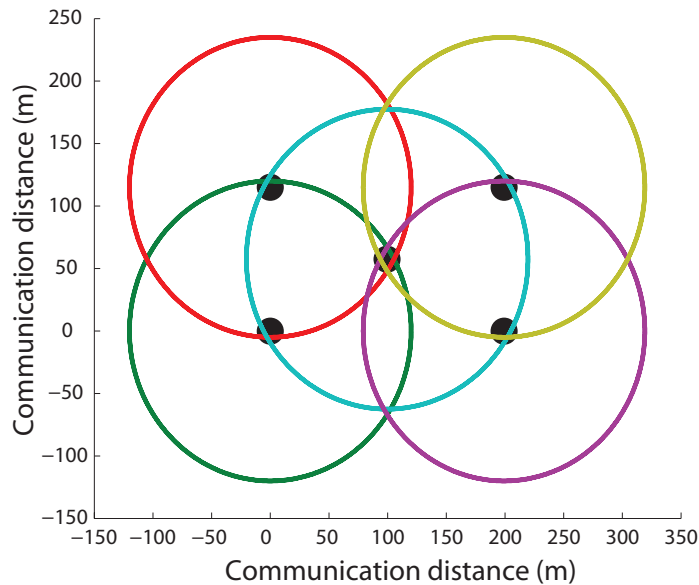


Figure 5.4: Outdoor log-distance model communication area for bow-tie topology

By comparing figure 5.4 with figure 4.6 from the previous chapter, we can see that the mathematical model now predicts a considerably smaller overall topology. This is because of the reduction in predicted communication distance between communicating nodes, which was also seen in section 5.2.1 where the log-distance model for indoor applications were used for communication distance prediction. The butterfly and hybrid butterfly experiences the same reduction in overall topology size.

When we compare the indoor and the outdoor log-distance models, we can see that the outdoor log-distance model predicts a larger communication distance between nodes than the indoor model. In outdoor environments, the amount of objects between communicating nodes are usually less than the amount of objects between communicating nodes in indoor environments, translating into a reduction in interference experienced and therefore also an increase in communication distance.

### 5.3 Comparison of log-distance model and free-space attenuation

In this section, we compare the reliable communication distances as calculated by the log-distance model, used in this chapter, and the free-space attenuation equation, as used in chapter 4. In table 5.1, the transmit power and receiver sensitivity of the Cisco Aironet 1200 access point and Cisco Aironet 340 client device are shown. The maximum achievable uplink and downlink communication distances were calculated with both models. The outdoor log-distance model was used for these calculations and  $W$ , the wall attenuation factor, was taken into account in the free-space attenuation model. The results of these calculations are shown in figure 5.5.

Table 5.1: Cisco Aironet hardware specifications (802.11b, 1Mbps) [1, 2, 3]

	Access Point:	Client Adapter:
	Cisco Aironet 1200	Cisco Aironet 340
Transmit Power:	20 dBm	30 mW = 14.77dBm
Receiver Sensitivity:	-94 dBm	-90dBm

It can be seen that the free-space attenuation model, which does not take any interference effects into account, can not describe the reliable communication distances that can be achieved in practical situations accurately.

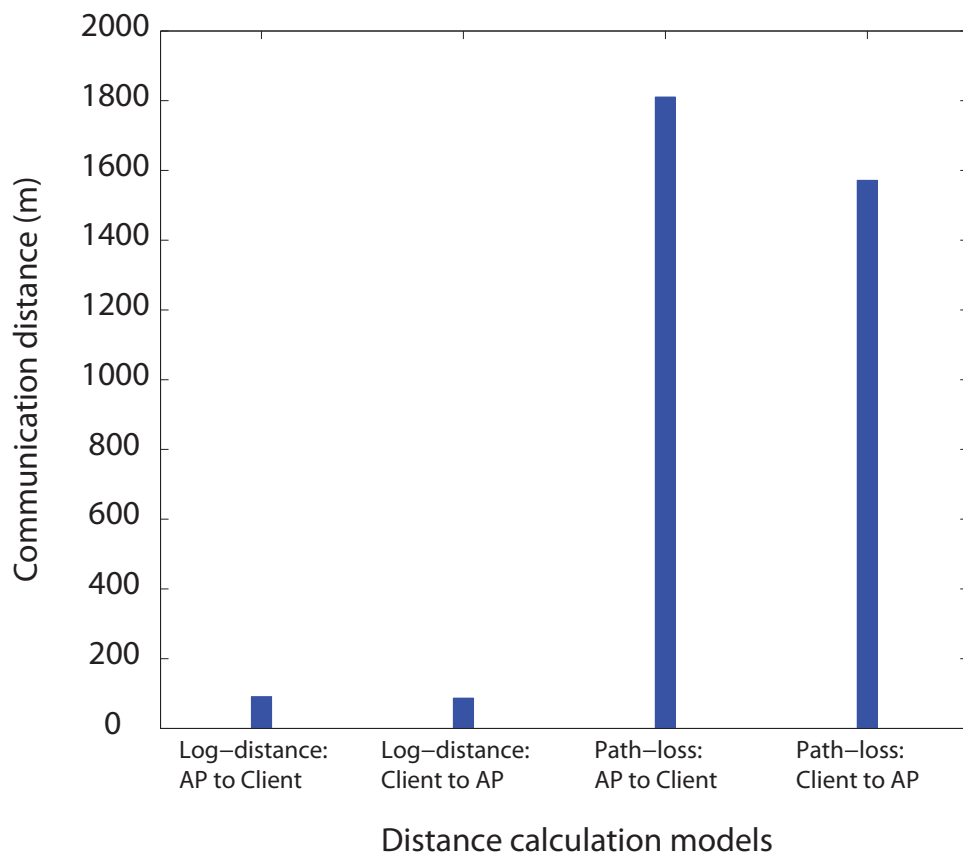


Figure 5.5: Comparing log-distance and free-space signal attenuation models

## 5.4 Signal attenuation caused by objects

In a typical indoor environment, many large objects including walls, doors and windows may be located between a wireless transmitter and receiver. These objects can cause varying amounts of signal attenuation, based on the material from which the object is made and its size. Figure 5.6 shows some common materials and their attenuation properties.

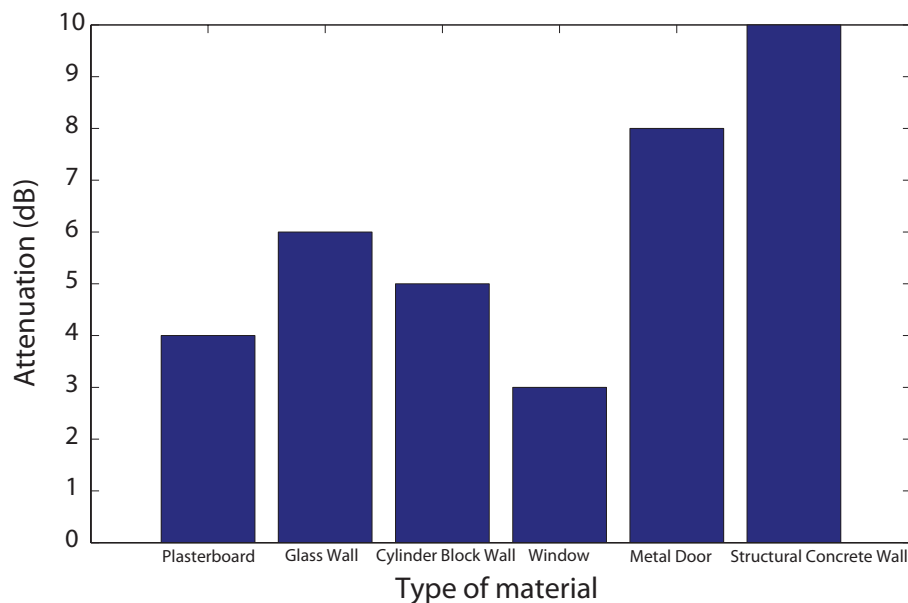


Figure 5.6: Attenuation caused by objects

Figure 5.7 depicts the bow-tie topology with one node located in the corner of an office with walls made of plasterboard. The figure gives an example of how objects in the transmission path influence signal strength and optimal node placement.

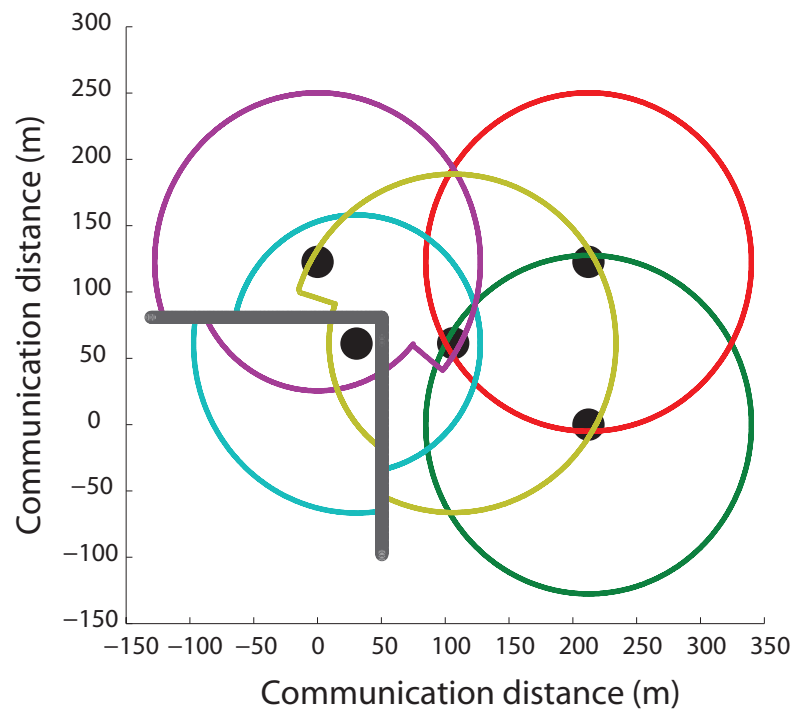


Figure 5.7: Example of a wall in a bow-tie topology

## 5.5 Conclusion

We discussed the effects of interference caused by obstructions in the signal propagation path of communicating nodes in this chapter. The mathematical model, used to predict the size and the possible placement of nodes in network coding suitable topologies, were modified to use a more realistic communication distance prediction method. In the next chapter, we are going to use this modified mathematical model to calculate the optimum node positions within all of the identified network coding suitable topologies. The calculated node co-ordinates are used in the simulations to ensure network coding would be possible in practical situations.

# Chapter 6

## Simulation of identified topologies

---

*In this chapter, we simulate the identified network coding suitable topologies in OPNET modeler to ensure that network coding can successfully be implemented in these topologies using the node placement areas as predicted by our mathematical model. We give an introduction to OPNET modeler and discuss some of the functions and models used. We then discuss the creation and implementation of our network coding model and the algorithm used to code and decode packets.*

---

### 6.1 Introduction to OPNET

OPNET modeler wireless suite is used for the simulation phase of the project. In OPNET modeler's documentation [23], the software is described as follows: "OPNET provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. Both behaviour and performance of modeled systems can be analysed by performing discrete event simulations. The OPNET environment incorporates tools for all phases of a study, including model design,

---

simulation, data collection, and data analysis” [23].

In the following sections, we will describe the essential procedures and models used in our simulations and give an overview of the relevant components of these procedures and models.

### 6.1.1 OPNET radio transceiver pipeline

To understand how OPNET simulates the physical wireless environment, it is essential to understand the radio transceiver pipeline used by OPNET modeler wireless suite to simulate the transmission of packets in a wireless communication environment. This will give an overview of the physical wireless transmission attributes OPNET takes into account and what attributes OPNET neglects during simulations.

An overview of the radio transceiver pipeline used in OPNET modeler is shown in figure 6.1 [23].

A detailed description of each stage is presented in appendix B.

### 6.1.2 ICI packets

Interface Control Information (ICI) packets are simple data objects. They only contain storage for user-defined values and no notation for size or ownership [23]. ICI packets have different applications in OPNET, but in this project we use ICI packets for simple communication between layers. This enables the testing of network coding concepts without changing the structure of transmitted data packets.

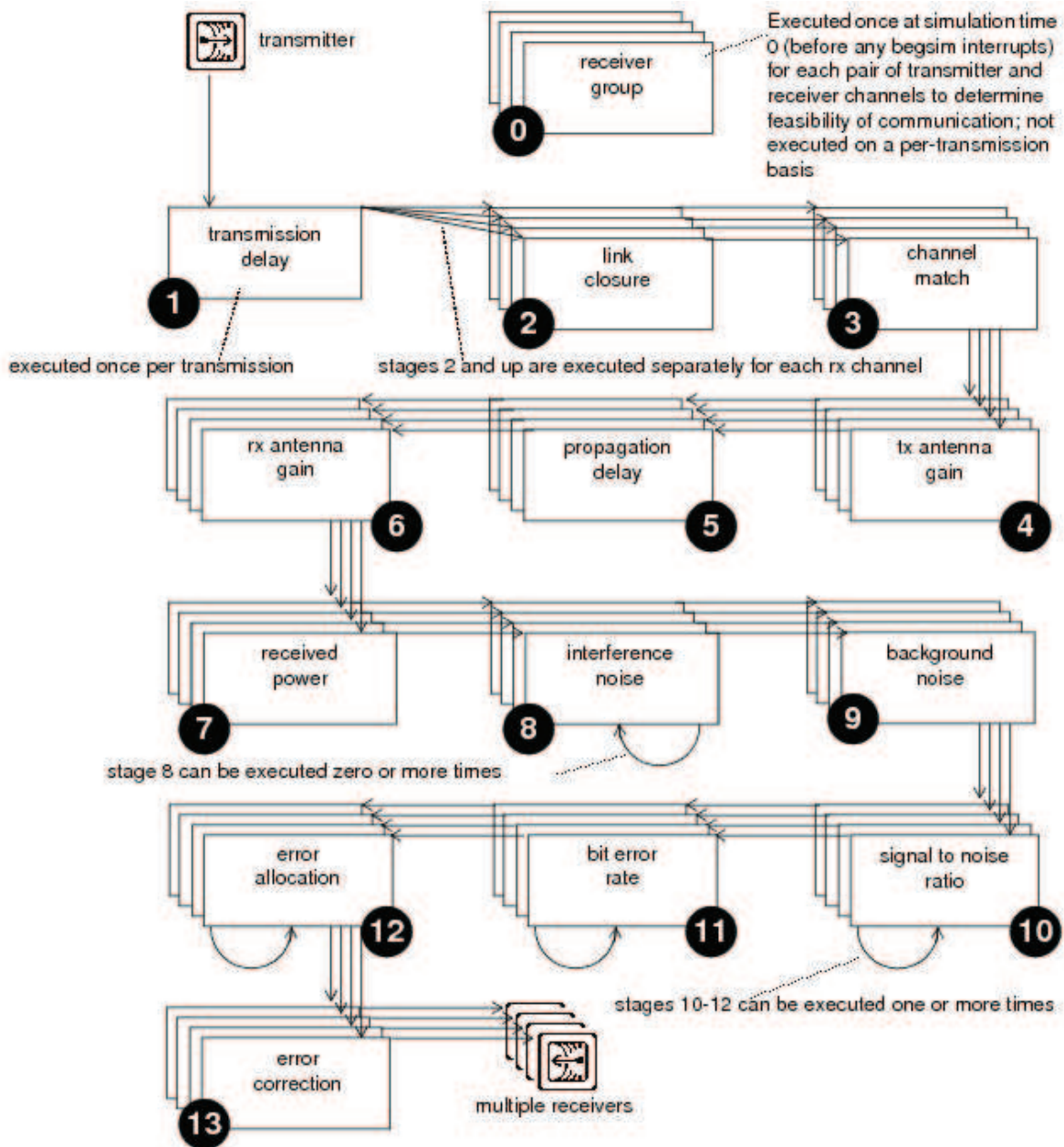


Figure 6.1: OPNET radio transceiver module overview



### 6.1.3 OPNET functions used

OPNET's Simulation Kernel provides services that can be accessed through Kernel Procedures (KPs). KPs can be called from within process models, Transceiver Pipeline stages, C/C++ functions that have been scheduled as interrupts or simply C/C++ functions that are directly or indirectly invoked from one of these contexts [23]. The KPs used in this project are described in appendix C.

## 6.2 OPNET standard models

An example of the workspace created in OPNET, is shown in figure 6.2. In this example, the hybrid butterfly topology is shown. All the different topologies were simulated in a similar fashion with varying amounts of nodes, located at different positions.

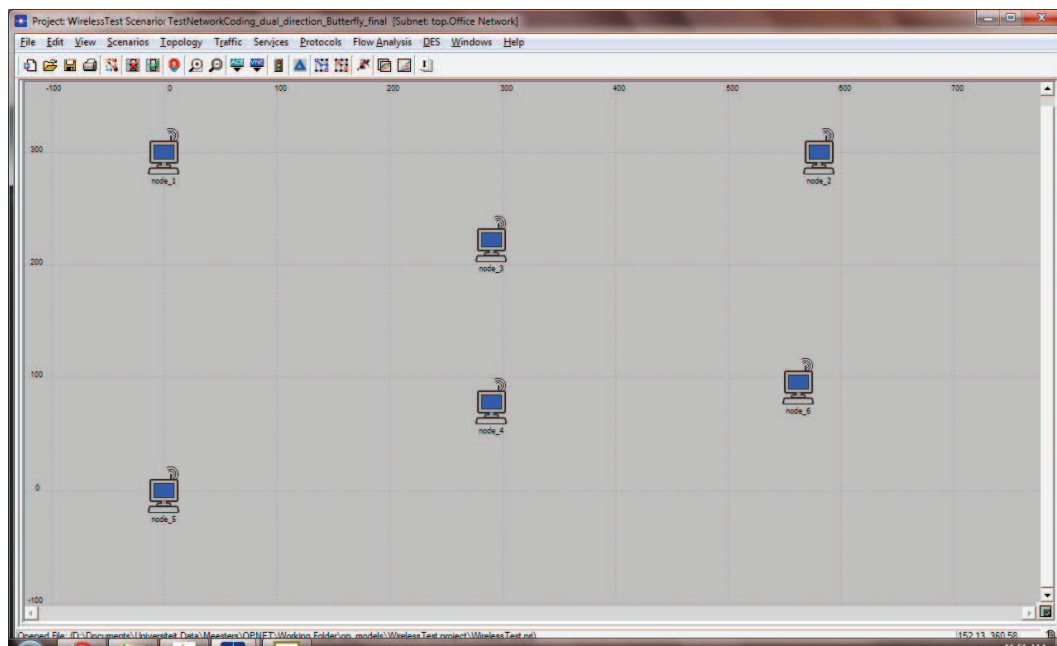


Figure 6.2: OPNET workspace example

In OPNET's wireless database, two different wireless models can be used. The more advanced model, shown in figure 6.4, includes all the layers of the OSI protocol stack

while the basic model, shown in figure 6.3, only includes the physical and data link layers with a packet source and sink combination.

This project aims to implement network coding on a basic level, therefore the simpler of the two models was used.

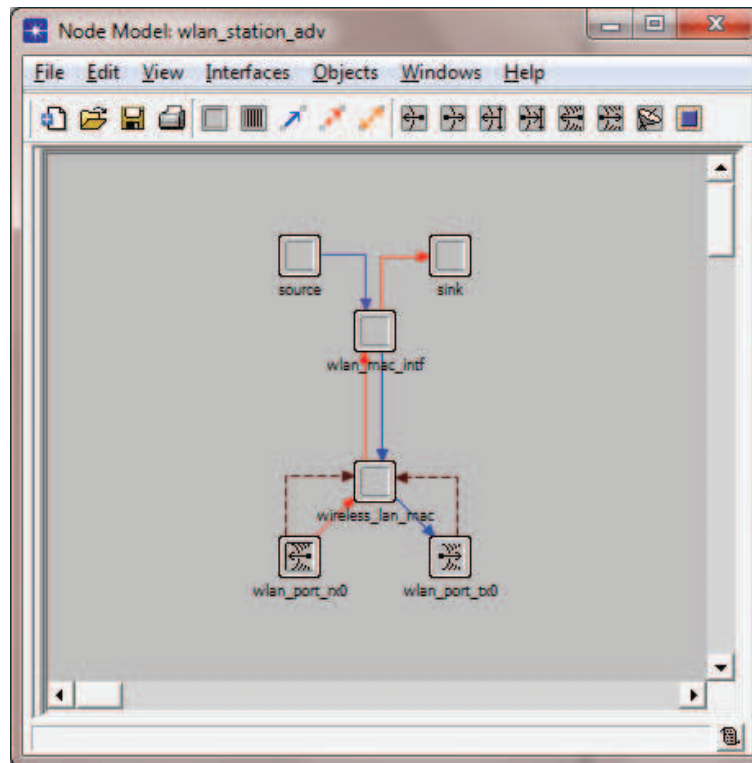


Figure 6.3: OPNET standard WLAN station

### 6.3 OPNET custom model

The creation of the custom OPNET wireless node station used in the simulations and the functions thereof, are discussed in the next subsections. Refer to appendix F for the created OPNET code and project files.

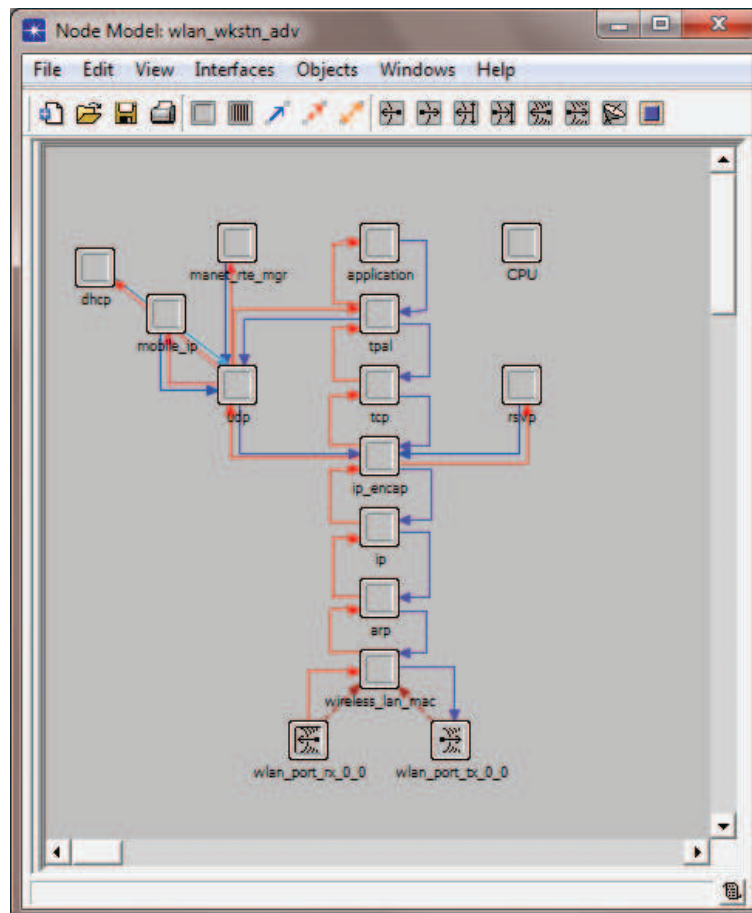


Figure 6.4: OPNET standard WLAN workstation

### 6.3.1 Changes to standard model

The custom OPENT wireless node station model used during simulations, was created by changing and adding functionality to the standard OPNET wireless node station model. The changes to the standard model are described next.

#### Physical layer

No changes were made to the physical layer for our simulations. The IEEE 802.11 a/b/g standards are utilized as supplied with OPNET modeler wireless suite.

#### Data link layer

The data link layer contains the MAC layer. Changes to the standard MAC layer include:

- Accept and forward four "multicast network coding MAC addresses" apart from the node's address and broadcast address.
- After a packet is stripped from the data link layer's headers, the source, destination and the four multicast coding addresses are sent to the higher layers using an ICI packet.

The data link layer also contains the Logical Link Control (LLC) layer. Changes to the standard LLC layer include:

- The verification of destination addresses was disabled.
- ICI packets are sent to the higher layers as packets proceed upwards in the IP stack.

### 6.3.2 New processor: Network coding layer

Figure 6.5 depicts the modified wireless station model. A new network coding layer was created and placed on top of the LLC layer within the data link layer. The created finite state diagram of the network coding layer, is shown in figure 6.6.

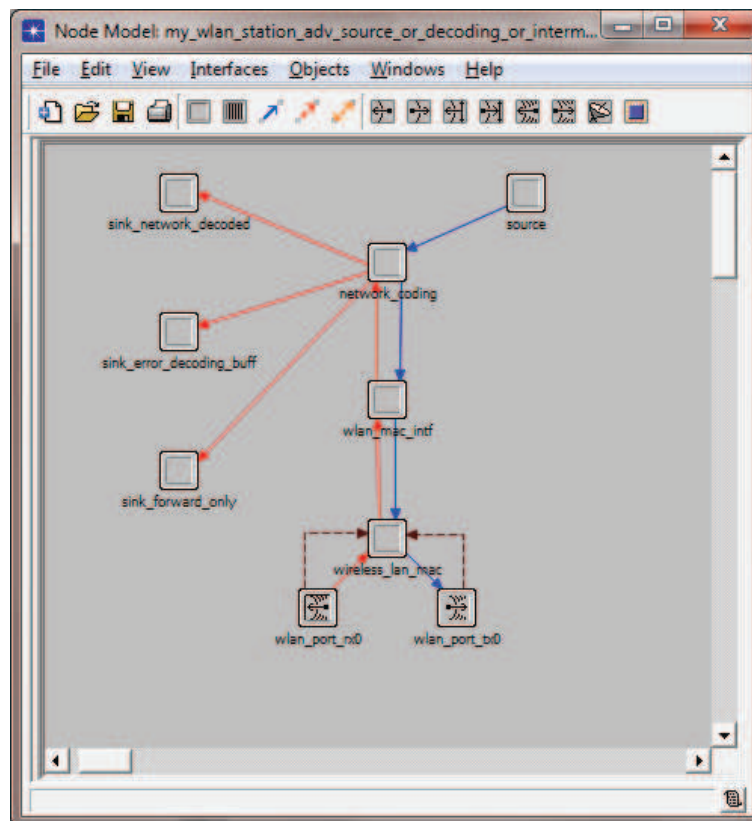


Figure 6.5: OPNET custom WLAN network coding station

Depending on the node's location in the topology, a node implementing this new model can be set to act as a source and/or destination, an intermediate node or a forward only node. This node functions can be set on a per packet basis, but for our simulations, the topologies were chosen to be fixed, thus the node function is set at the beginning of the simulation and maintained for the whole simulation.

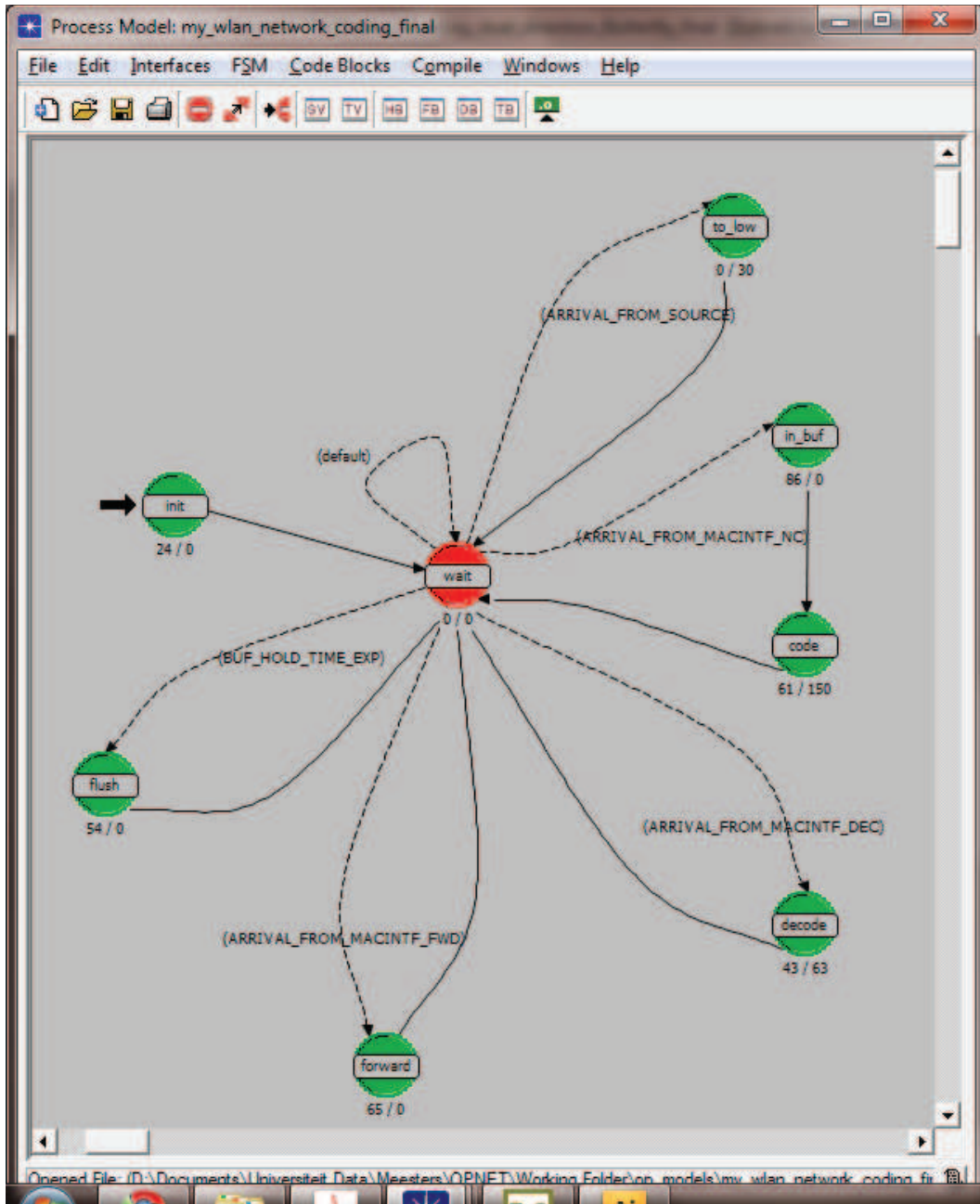


Figure 6.6: OPNET custom WLAN network coding layer

### **Node function set as source or destination**

When a node is set to act as a source or destination, the node can generate packets, it can receive packets, or both, depending on the node parameters.

If the user wants a specific node to act as a source, the parameters listed under the "Source" tab in the node attributes must be specified. This includes the "Direct destination address" (the address of the destination node directly opposite to this source node) and the "Network coding test value" (a test value added as a header to each generated packet to test and verify the network coding and decoding algorithms) parameters. The user must also specify the parameters listed under the "Traffic generation Parameters". These parameters specifies the generated traffic's packet size, transmission rate, shape of traffic, etc. Refer to [23] for more information on these traffic generation parameters.

If the user wants a specific node to act as a destination, the parameters listed under the "Destination" tab in the node attributes must be specified. This includes the "Decode packets?" (to enable or disable the decoding of packets), the "Decoding buffer hold time" (the time packets can be inside the decoding buffer before being "flushed" or forwarded) and the "Direct listen source address" (the address of the source node directly opposite to this destination node) parameters.

### **Node function set as intermediate**

Intermediate nodes or "smart" coding nodes can monitor incoming packets generated by the source nodes and identify coding opportunities. If the user wants a specific node to act as an intermediate coding node, the parameters listed under the "intermediate" tab in the node attributes must be specified. This includes the "Buffer Hold Time" (the time packets can be inside the coding buffer before being "flushed" or forwarded), the "Buffer Hold Time Increment" (if the user wants to study the effect a variation of buffer hold time has on the performance of the network), the "Destination Coding address 1"

(the address that the algorithm uses for coded packets) and the "Destination Coding address 2" (used to enable dual direction coding of packets) parameters.

### **Node function set as forward**

If the user wants a specific node to act as a forwarding node, the only configuration required is the specification of the "Multicast Coding addresses" under the "WLAN" attribute.

### **Configuration required by all nodes**

All nodes must be given an unique name and WLAN MAC address, the correct multicast coding addresses, WLAN parameters (including type of IEEE communication standard to be used) and the correct node function. The attributes that can be adjusted are shown in figure 6.7.

### **6.3.3 New sink processors**

Instead of having the one default sink processor (where the received packets are destroyed), three sink processors were created. This separation enables the user to draw conclusions about the performance of the network coding topology's performance. All the network coded packets that were decoded successfully by a destination node, are sent to the sink processor "sink\_network\_decoded". All the packets that were not network coded but only forwarded and successfully received by a destination node, are sent to the sink processor "sink\_forward\_only". The packets that were received but flushed from the decoder's buffer, are sent to the sink processor "sink\_error\_decoding\_buff".



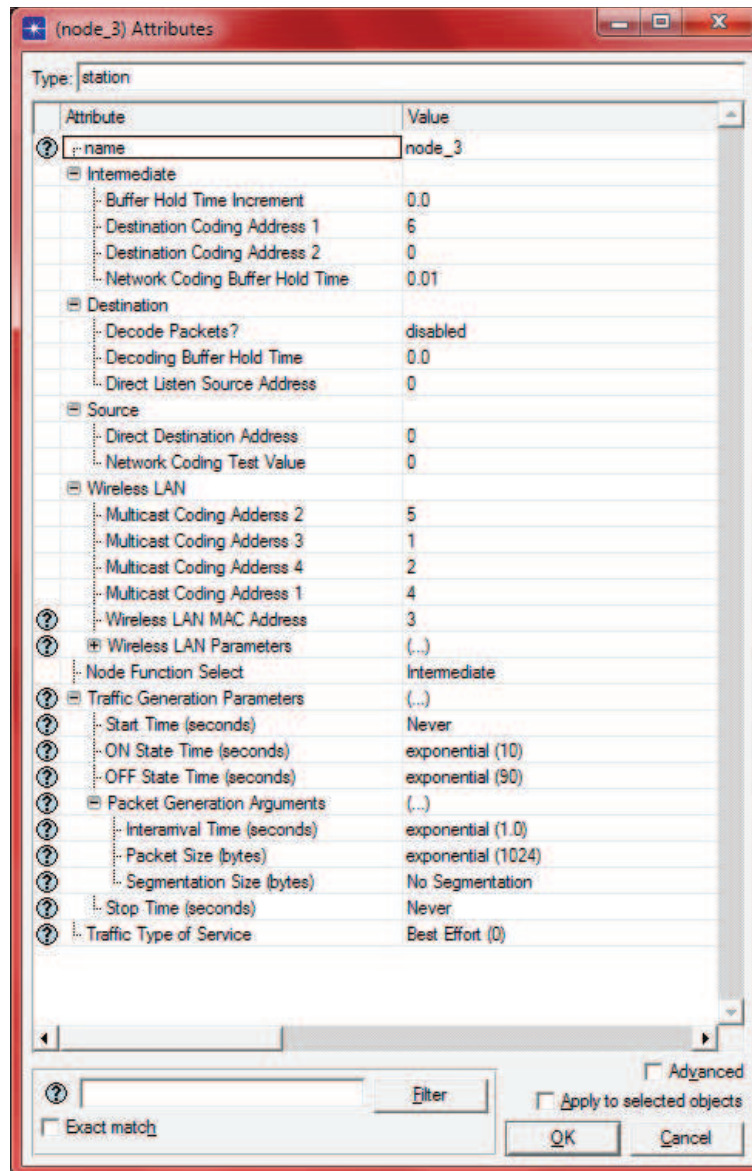
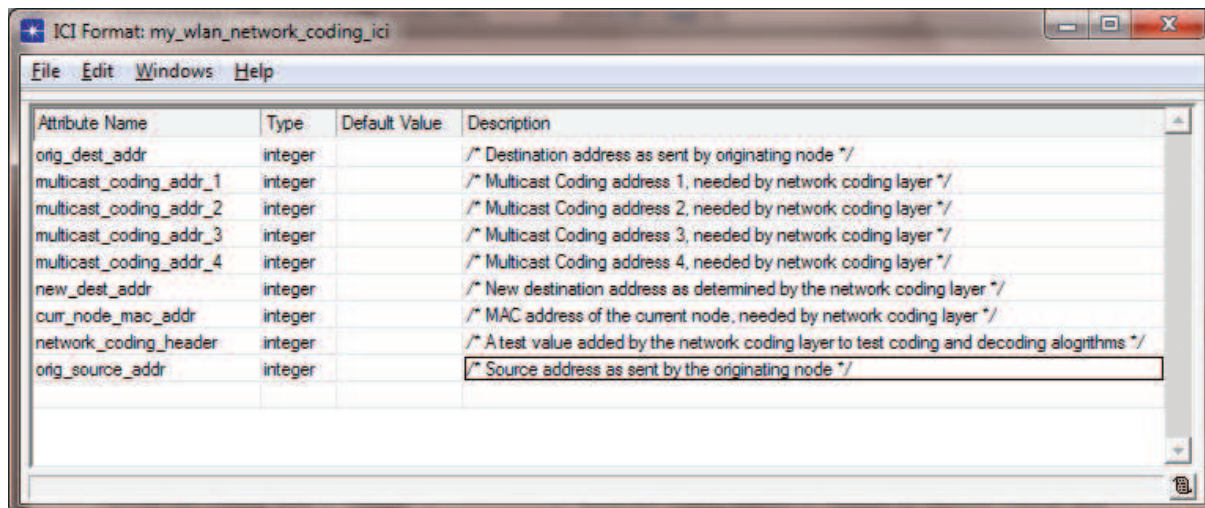


Figure 6.7: Custom network coding node attributes

### 6.3.4 New ICI for inter-layer communication

Figure 6.8 depicts the ICI created and used for interlayer communication in each node. The new network coding layer requires certain information to determine if network coding can take place. A description of exactly how the network coding layer determines whether network coding can take place or not, is discussed in the next section.



The screenshot shows a window titled "ICI Format: my\_wlan\_network\_coding\_ici" with a menu bar (File, Edit, Windows, Help) and a table of attributes. The table has four columns: Attribute Name, Type, Default Value, and Description. The attributes listed are:

Attribute Name	Type	Default Value	Description
orig_dest_addr	integer		/* Destination address as sent by originating node */
multicast_coding_addr_1	integer		/* Multicast Coding address 1, needed by network coding layer */
multicast_coding_addr_2	integer		/* Multicast Coding address 2, needed by network coding layer */
multicast_coding_addr_3	integer		/* Multicast Coding address 3, needed by network coding layer */
multicast_coding_addr_4	integer		/* Multicast Coding address 4, needed by network coding layer */
new_dest_addr	integer		/* New destination address as determined by the network coding layer */
cur_node_mac_addr	integer		/* MAC address of the current node, needed by network coding layer */
network_coding_header	integer		/* A test value added by the network coding layer to test coding and decoding algorithms */
orig_source_addr	integer		/* Source address as sent by the originating node */

Figure 6.8: OPNET custom ICI packet

### 6.3.5 Network coding algorithm

A flow diagram, describing the operation of a coding or intermediate node, is shown in figure 6.9. The operation of the buffer used in a coding or intermediate node, is explained in detail as a flow diagram in figure 6.10.

A flow diagram, describing the operation of a decoding or destination node, is shown in figure 6.11. The operation of the buffer used in a decoding or destination node, is explained in detail as a flow diagram in figure 6.12.

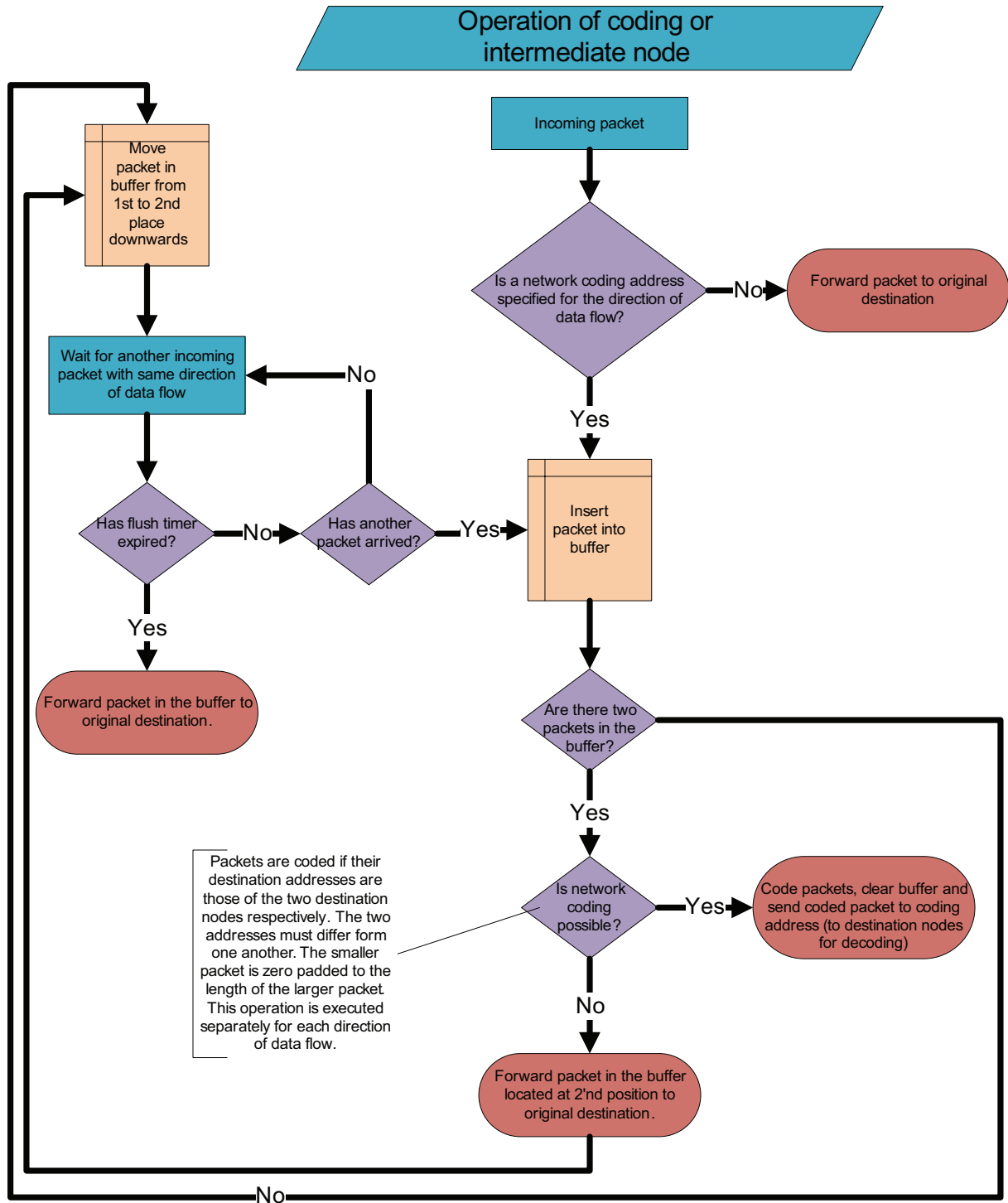


Figure 6.9: Flowchart of coding node

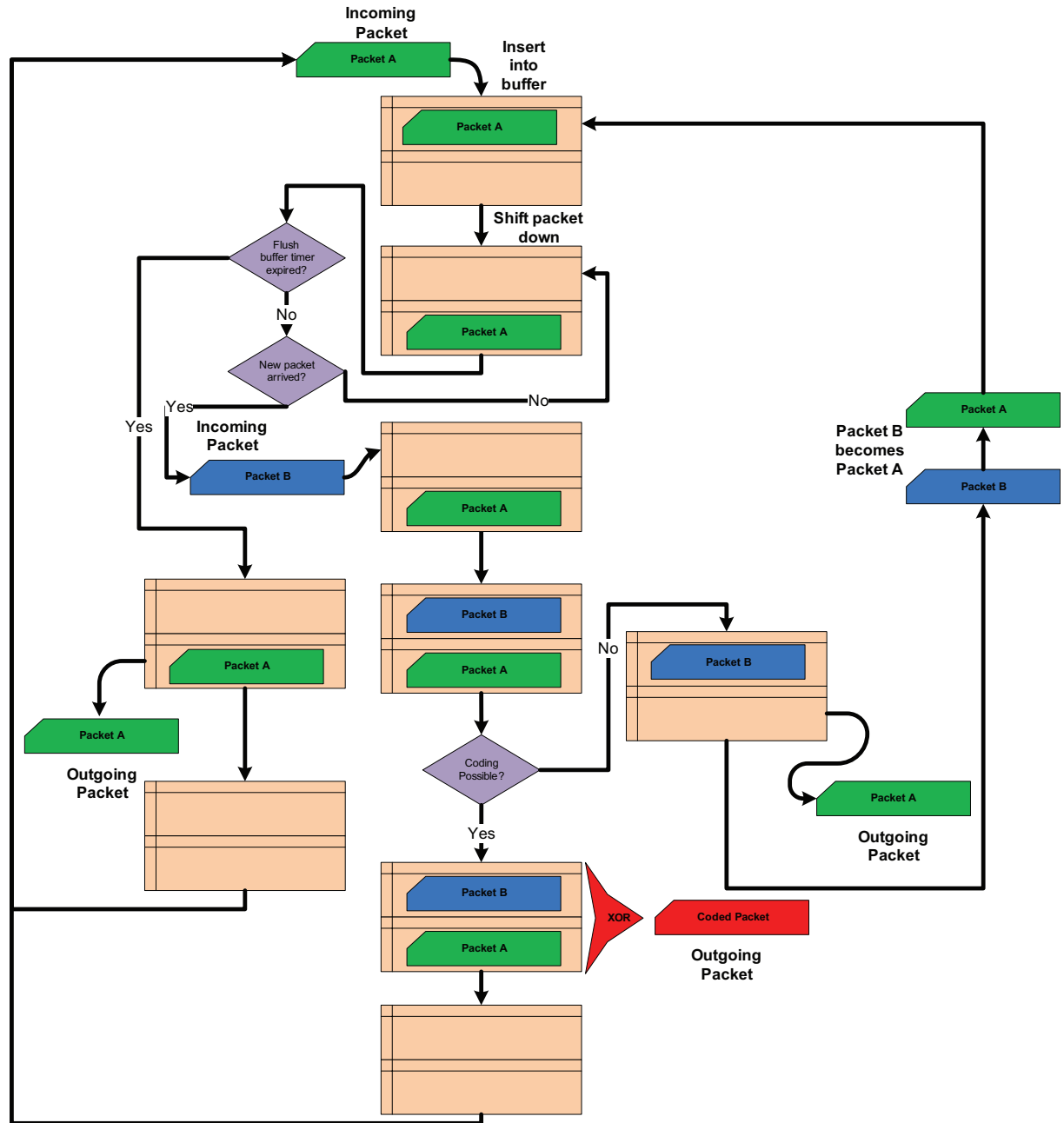


Figure 6.10: Buffer used in coding node

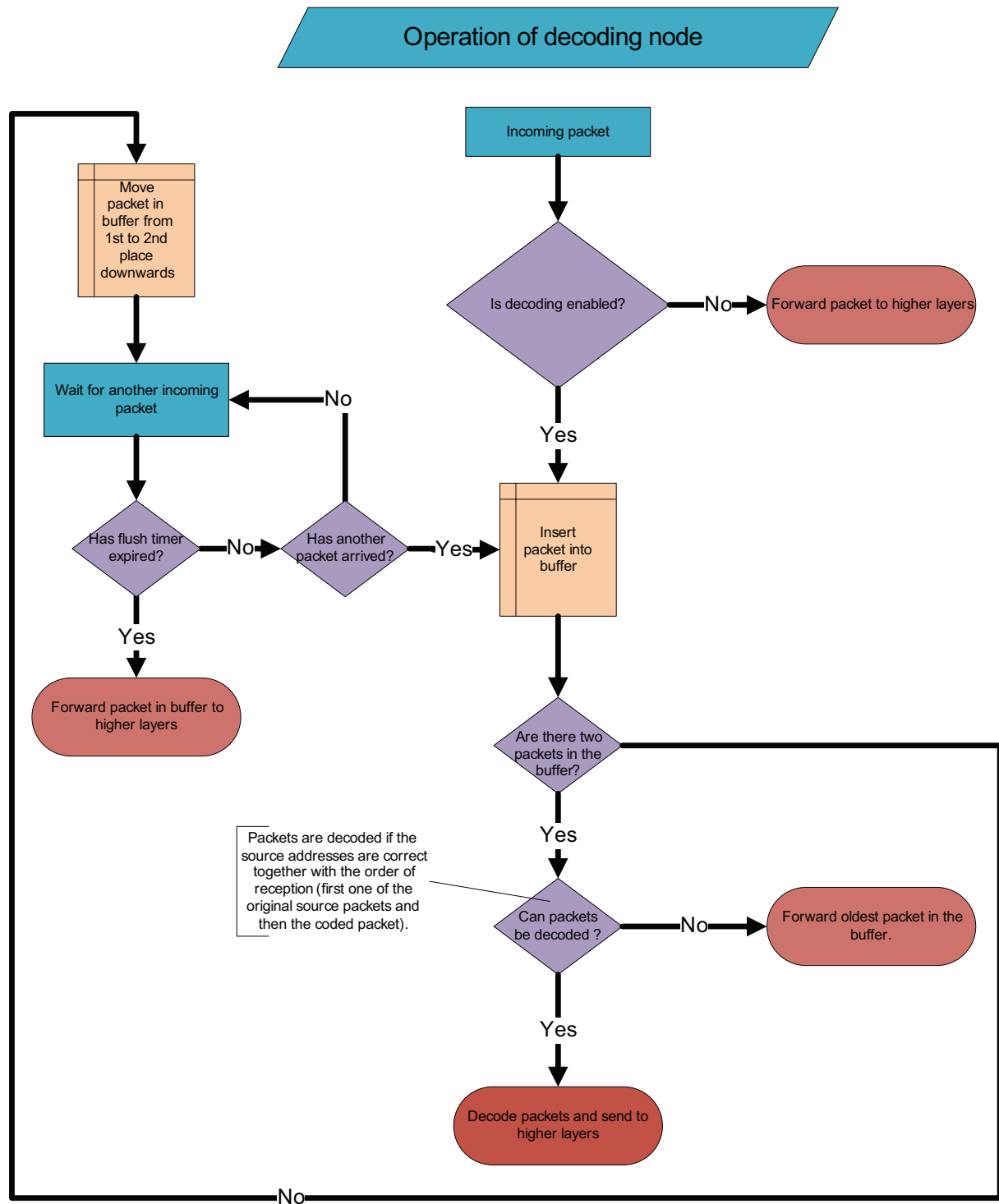


Figure 6.11: Flowchart of decoding node

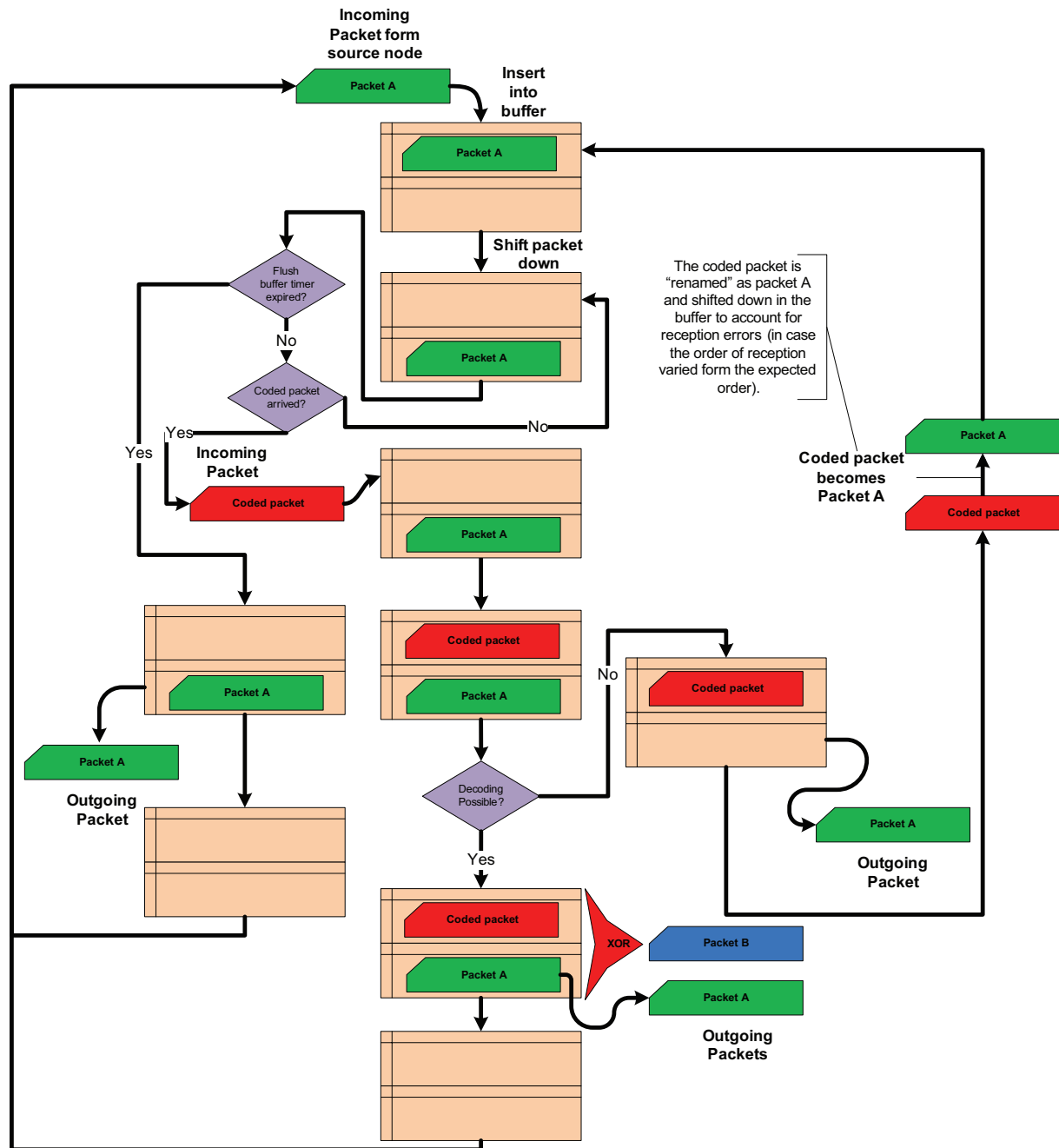


Figure 6.12: Buffer used in decoding node

## 6.4 Impact of algorithm design choices

A current limitation of the designed network coding algorithm is its limited scalability. The algorithm can only be used within the identified network coding suitable topologies discussed in chapter 3 or similar topologies, that is topologies with two source nodes and two destination nodes.

It is a well-known fact that the sequence of the XOR operation does not matter, that is  $A \text{ XOR } B = B \text{ XOR } A$ . When inspecting the decoding operation of a destination node in figure 6.11, one might notice the note indicating that the order of reception must be correct for decoding to take place. That is because of the order in which information propagates through the network. The first packet received by the decoding node will always be from the original source from where the coding node will listen and code the original packet with another. Only then will the coded packet be sent to the destination node. Therefore the decoding node will always receive the original packet and then the coded packet.

## 6.5 Comparison of simulation model to OPNET's standard model

To validate the created OPNET node, we used a very simple simulation to test the operation of the new node and compare it to the standard OPNET node. Two standard OPNET "station" nodes were placed in an interference free environment with 100m of distance separating them and configured to exchange data with one another, using the OPNET's default settings for data generation. The wireless MAC and physical layer parameters were also set to default and the simulation duration to 10 minutes. The workspace is shown in figure 6.13. Next, two of the newly created nodes were placed in the exact same positions and configured with the same default parameters. The same seed for data generation was used in both cases.

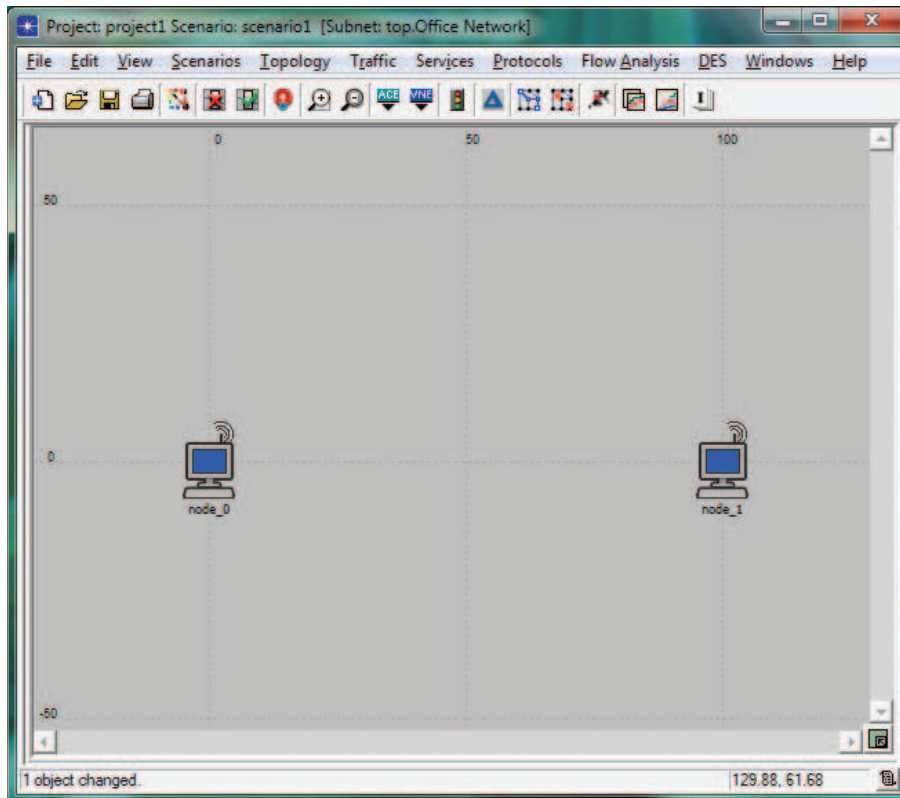


Figure 6.13: Workspace: Comparing models

The results of the two simulations were compared and showed no variation. This confirmed that the operation of the new node's lower layers was not changed in any way.

## 6.6 Conclusion

In this chapter, we discussed the simulation environment and presented the developed network coding algorithm. In the next chapter we will present the simulation results obtained for the bow-tie, butterfly and hybrid butterfly topologies.



# Chapter 7

## Simulation results

---

*In this chapter, we present the simulation results obtained for the bow-tie, butterfly and hybrid butterfly network coding suitable topologies. We create three simulation scenarios, one where the source nodes send data at a constant and synchronized rate through the network, one where the source nodes send data at a variable and unsynchronized rate through the network and one that duplicates the second scenario, but with a simulation runtime of ten hours.*

---

### 7.1 Constant bit stream as source

Results of the simulation with two source nodes, sending data in one direction through the network for a simulation time of 10 minutes at a rate of 100 kbits/sec, for each of the different identified network coding suitable topologies, are presented in this section.

### 7.1.1 Bow-tie topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 24.97% from an average of 408.917 kbits/sec to 306.804 kbits/sec when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.1.

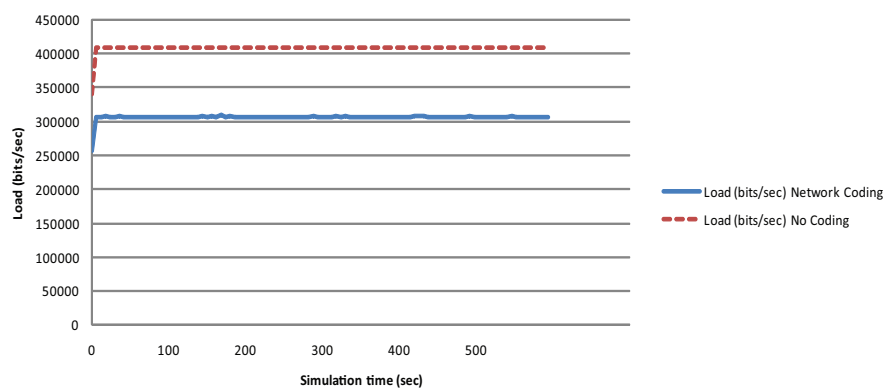


Figure 7.1: Global load, bow-tie simulation

When the two source nodes each transmit a constant bit stream, the global delay is increased by 1.27% from an average of 1.5143 ms to 1.5338 ms when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.2.

When the two source nodes each transmit a constant bit stream, the media access delay is increased by 14.91% from an average of 0.4037 ms to 0.4639 ms when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.3.

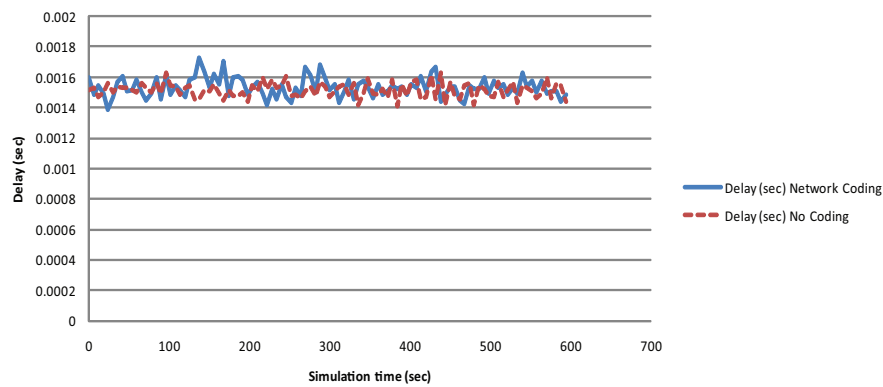


Figure 7.2: Global end-to-end delay, bow-tie simulation

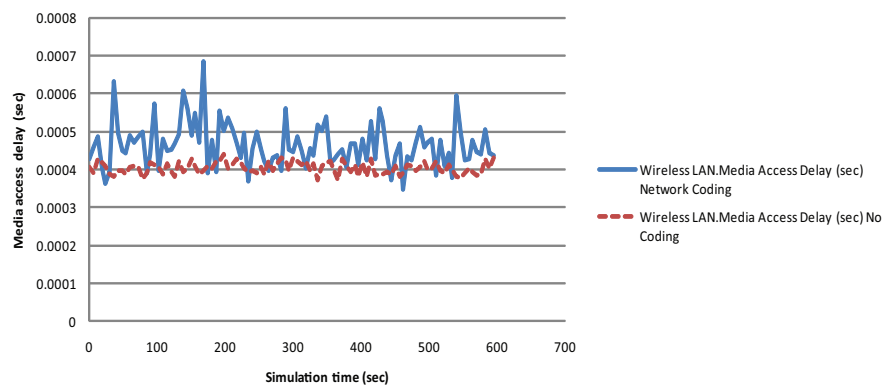


Figure 7.3: Global media access delay, bow-tie simulation

### 7.1.2 Butterfly topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 33.33% from an average of 609.280 kbits/sec to 406.187 kbits/sec when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.4.

When the two source nodes each transmit a constant bit stream, the global delay is de-

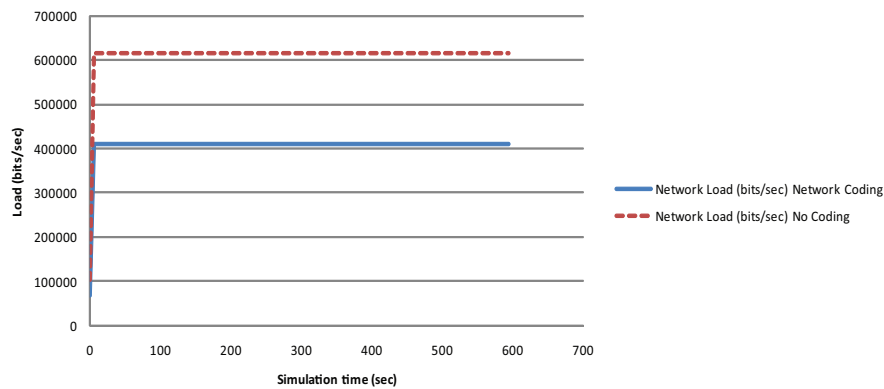


Figure 7.4: Global load, butterfly simulation

creased by 13.09% from an average of 1.4556 ms to 1.2651 ms when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.5.

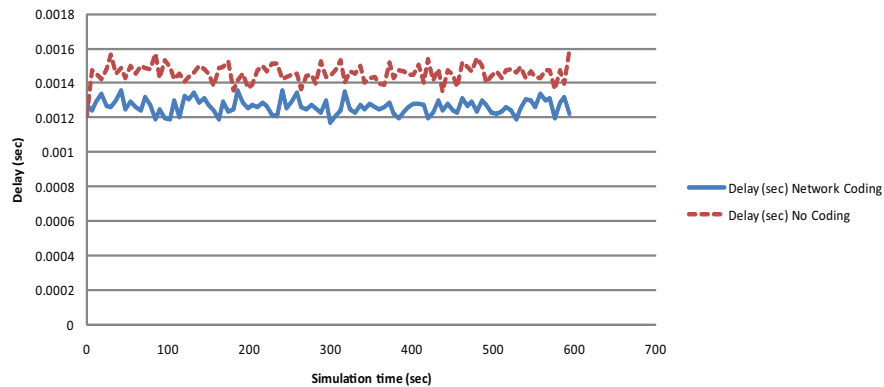


Figure 7.5: Global end-to-end delay, butterfly simulation

When the two source nodes each transmit a constant bit stream, the media access delay is decreased by 15.55% from an average of 0.2379 ms to 0.2009 ms when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.6.

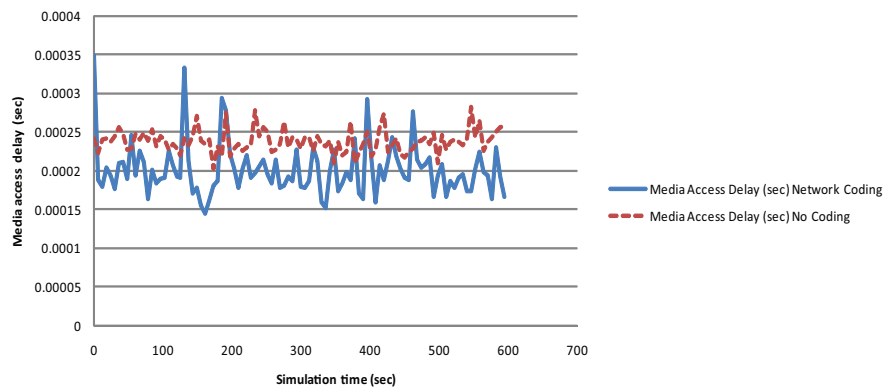


Figure 7.6: Global media access delay, butterfly simulation

### 7.1.3 Hybrid butterfly topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 20% from an average of 507.713 kbits/sec to 406.187 kbits/sec when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.7.

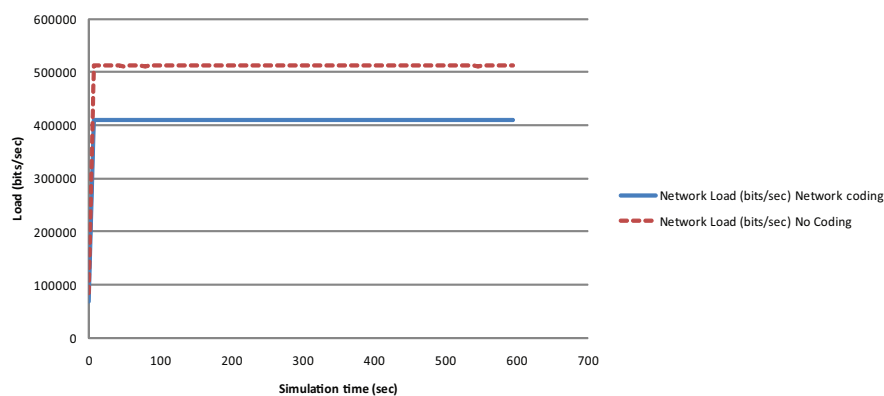


Figure 7.7: Global load, hybrid butterfly simulation

When the two source nodes each transmit a constant bit stream, the global delay is

decreased by 2.54% from an average of 1.4201 ms to 1.3840 ms when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.8.

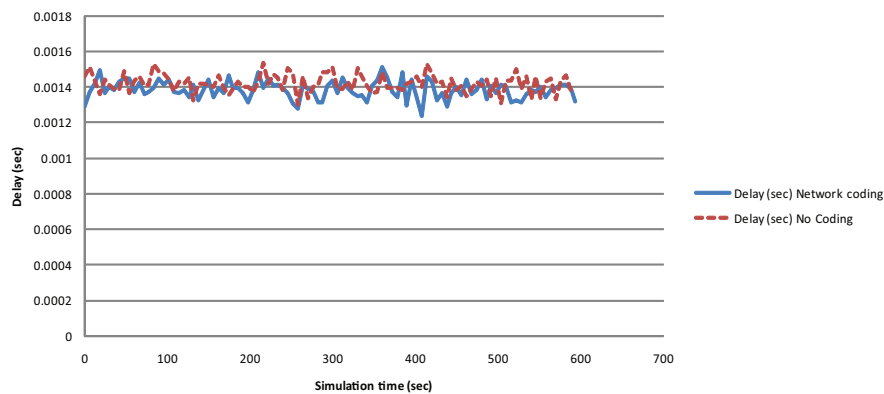


Figure 7.8: Global end-to-end delay, hybrid butterfly simulation

When the two source nodes each transmit a constant bit stream, the media access delay is increased by 55.73% from an average of 0.2661 ms to 0.4144 ms when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.9.

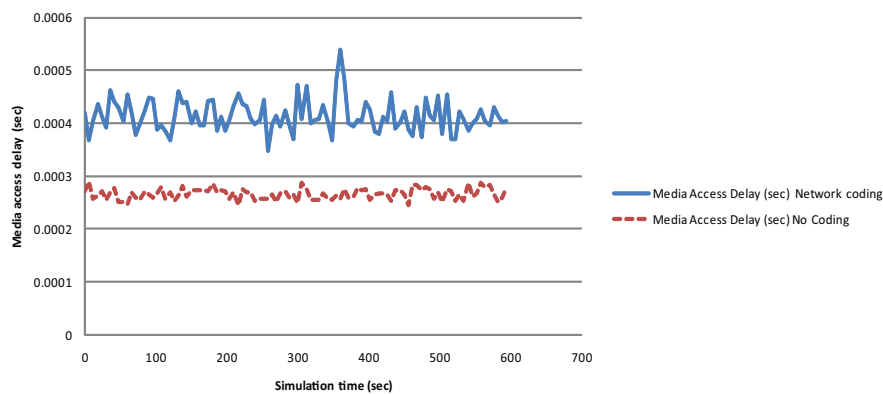


Figure 7.9: Global media access delay, hybrid butterfly simulation

### 7.1.4 Summary of results - Constant bit stream

A summary of the simulation results obtained when a constant bit stream is used as source, are shown in table 7.1.

Table 7.1: Simulation results for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Global Load:	24.97% decrease	33.33% decrease	20% decrease
Global Delay:	1.27% increase	13.09% decrease	2.54% decrease
Media Access Delay:	14.91% increase	15.55% decrease	55.73% increase

### 7.1.5 Variation of packet delay - Constant bit stream

The variation of IP packet delay (jitter) measured for each topology simulated, is shown in table 7.2. The difference in global jitter for when network coding is implemented and when network coding is not implemented, can be compared for the different topologies.

Table 7.2: Variation of IP packet delay for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Network Coding Implemented			
Maximum:	229.68 $\mu$ s	148.78 $\mu$ s	222.43 $\mu$ s
Average:	74.77 $\mu$ s	48.22 $\mu$ s	54.34 $\mu$ s
No Network Coding:			
Maximum:	214.33 $\mu$ s	260.44 $\mu$ s	176.91 $\mu$ s
Average:	64.93 $\mu$ s	58.14 $\mu$ s	60.26 $\mu$ s
Difference in average:	15.15% increase	17.06% decrease	9.82% decrease

## 7.2 Variable bit stream as source

Results of the simulations with two source nodes, sending data in one direction through the network for a simulation time of 10 minutes at a variable rate, for each of the different identified network coding suitable topologies, are presented in this section.

When the two source nodes each transmit a variable bit stream, OPNET generates a random bit stream that is unique to every simulation instance. In order to compare different simulations, we must be aware of any differences in source traffic between the simulations that we compare. Therefore, the average load of each source node is also given.

### 7.2.1 Bow-tie topology

The average load for source node one was 67.406 kbits/sec and 63.529 kbits/sec for the network coding and traditional communication instances respectively. That is a difference of 5.75% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.10.

The average load for source node two was 61.809 kbits/sec and 67.325 kbits/sec for the network coding and traditional communication instances respectively. That is a



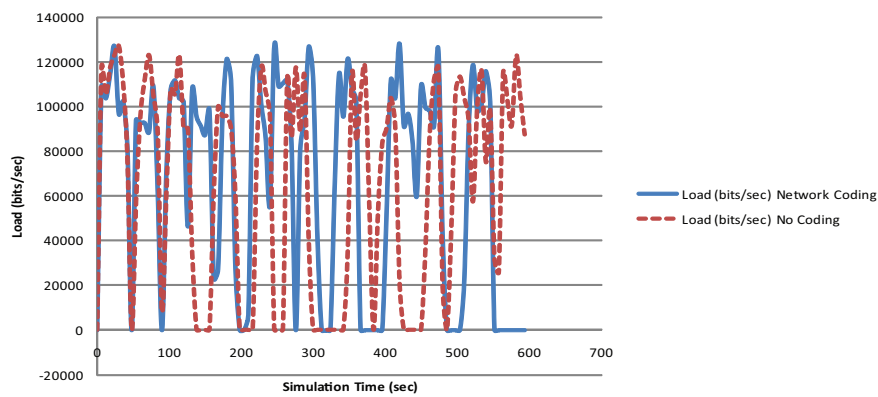


Figure 7.10: Source node one load, bow-tie simulation

difference of 8.92% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.11.

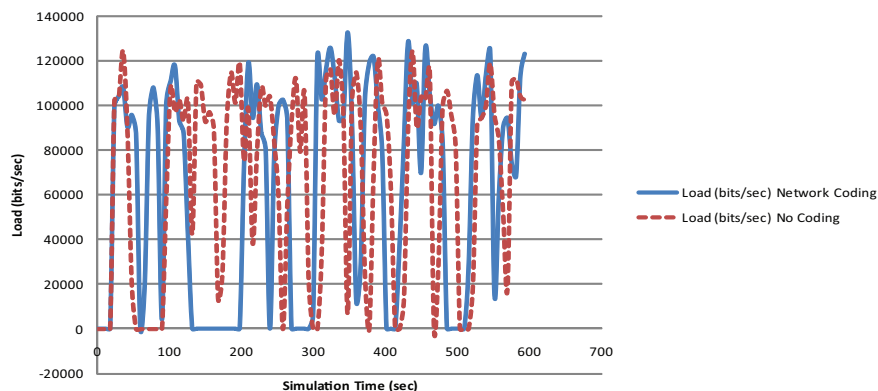


Figure 7.11: Source node two load, bow-tie simulation

When the two source nodes each transmit a variable bit stream, the global load is decreased by 4.967% from an average of 261.707 kbits/sec to 248.709 kbits/sec when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.12.

When the two source nodes each transmit a variable bit stream, the global delay is

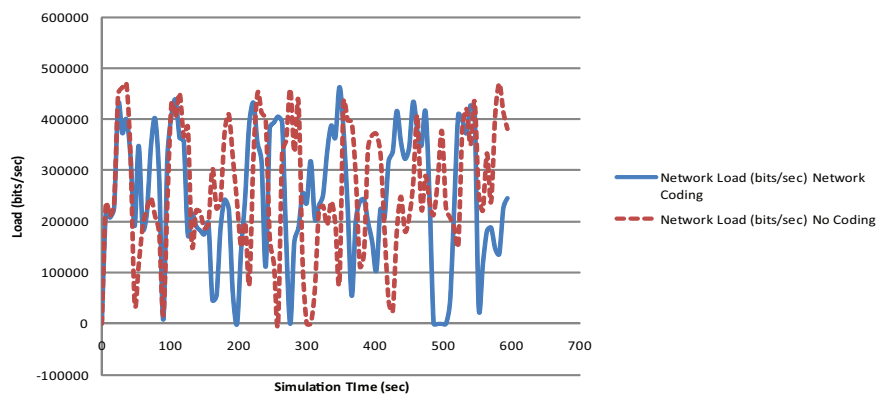


Figure 7.12: Global load, bow-tie simulation

increased by 0.32% from an average of 4.5617 ms to 4.5763 ms when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.13.

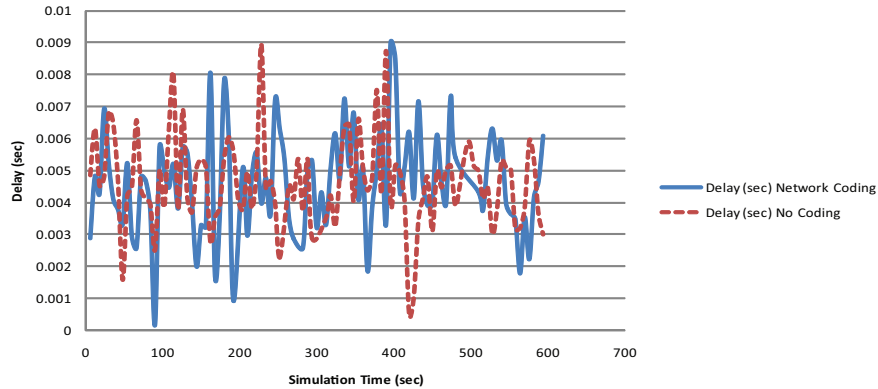


Figure 7.13: Global end-to-end delay, bow-tie simulation

When the two source nodes each transmit a variable bit stream, the media access delay is decreased by 0.102% from an average of 3.4373 ms to 3.4338 ms when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.14.

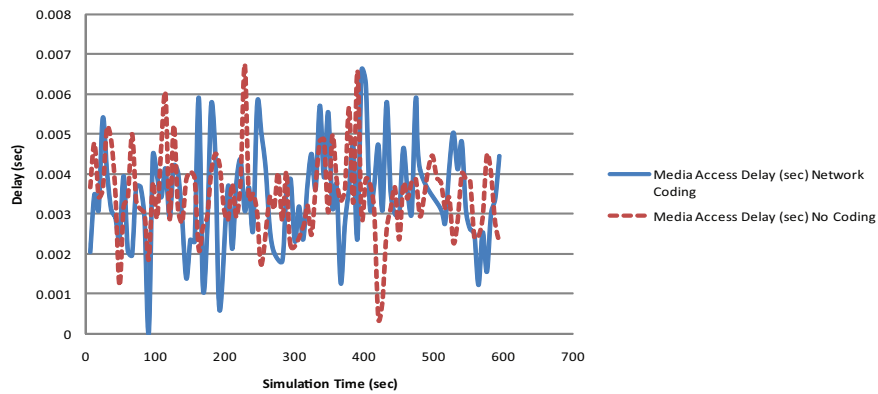


Figure 7.14: Global media access delay, bow-tie simulation

## 7.2.2 Butterfly topology

The average load for source node one was 69.919 kbits/sec and 69.591 kbits/sec for the network coding and traditional communication instances respectively. That is a difference of 0.469% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.15.

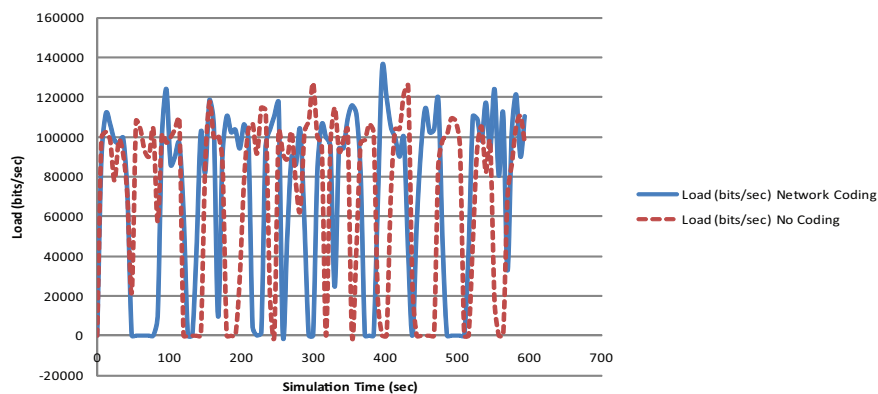


Figure 7.15: Source node one load, butterfly simulation

The average load for source node two was 64.717 kbits/sec and 66.014 kbits/sec for the network coding and traditional communication instances respectively. That is a differ-

ence of 2.004% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.16.

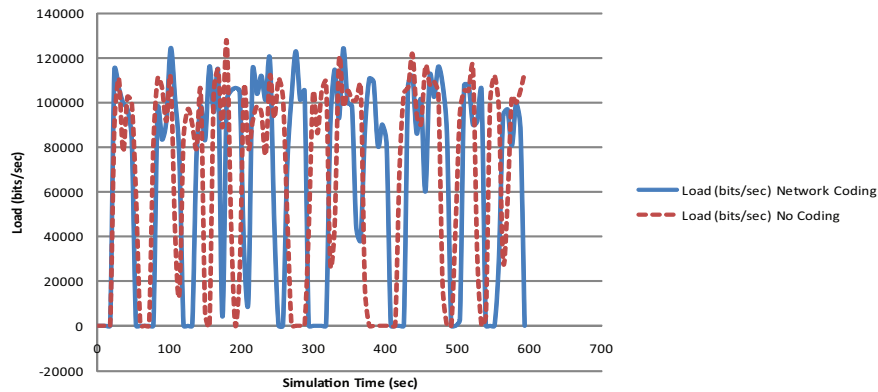


Figure 7.16: Source node two load, butterfly simulation

When the two source nodes each transmit a variable bit stream, the global load is decreased by 33.810% from an average of 406.815 kbits/sec to 269.271 kbits/sec when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.17.

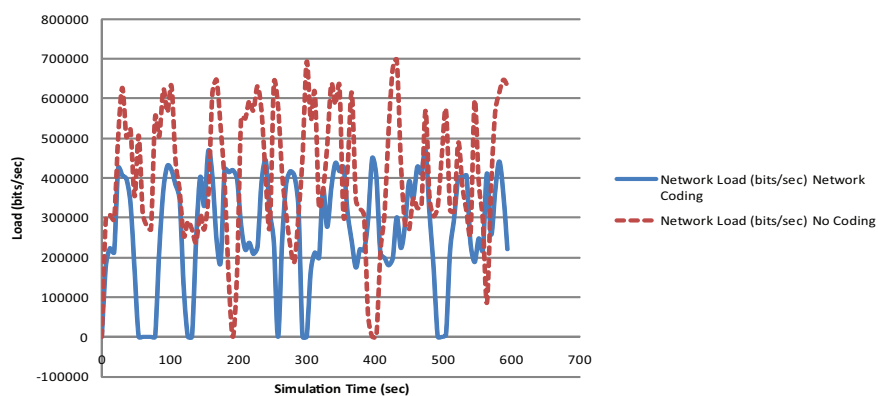


Figure 7.17: Global load, butterfly simulation

When the two source nodes each transmit a variable bit stream, the global delay is in-

creased by 110.774% from an average of 3.3895 ms to 7.1442 ms when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.18.

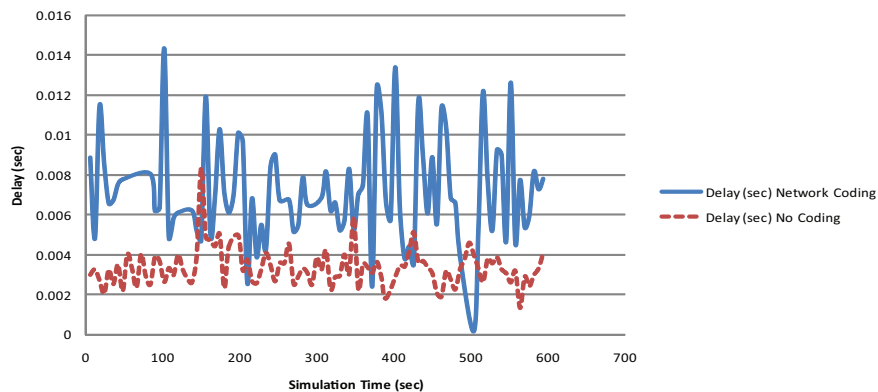


Figure 7.18: Global end-to-end delay, butterfly simulation

When the two source nodes each transmit a variable bit stream, the media access delay is increased by 186.989% from an average of 2.0990 ms to 6.0239 ms when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.19.

### 7.2.3 Hybrid butterfly topology

The average load for source node one was 67.953 kbits/sec and 79.995 kbits/sec for the network coding and traditional communication instances respectively. That is a difference of 17.721% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.20.

The average load for source node two was 67.120 kbits/sec and 71.107 kbits/sec for the network coding and traditional communication instances respectively. That is a differ-

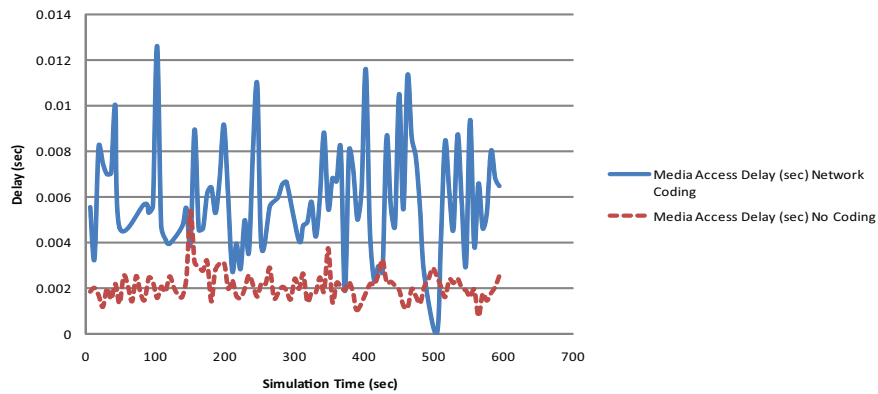


Figure 7.19: Global media access delay, butterfly simulation

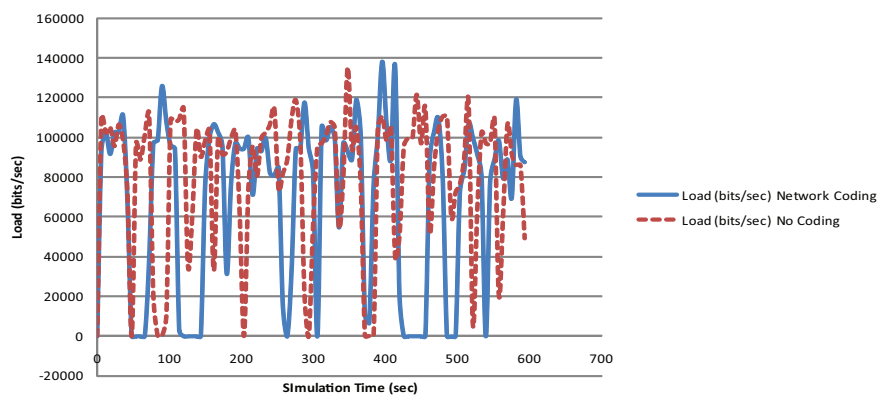


Figure 7.20: Source node one load, hybrid butterfly simulation

ence of 5.940% in source traffic introduced into the topology. The applicable results of the simulation are shown in figure 7.21.

When the two source nodes each transmit a variable bit stream, the global load is decreased by 27.630% from an average of 373.282 kbits/sec to 270.145 kbits/sec when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.22.

When the two source nodes each transmit a variable bit stream, the global delay is in-

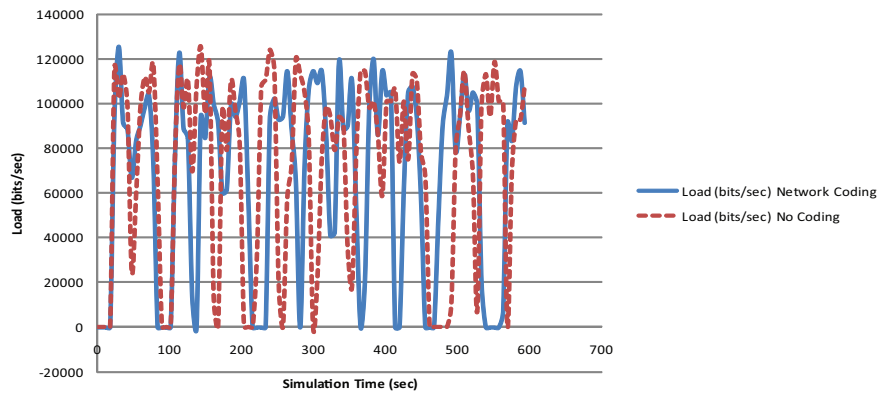


Figure 7.21: Source node two load, hybrid butterfly simulation

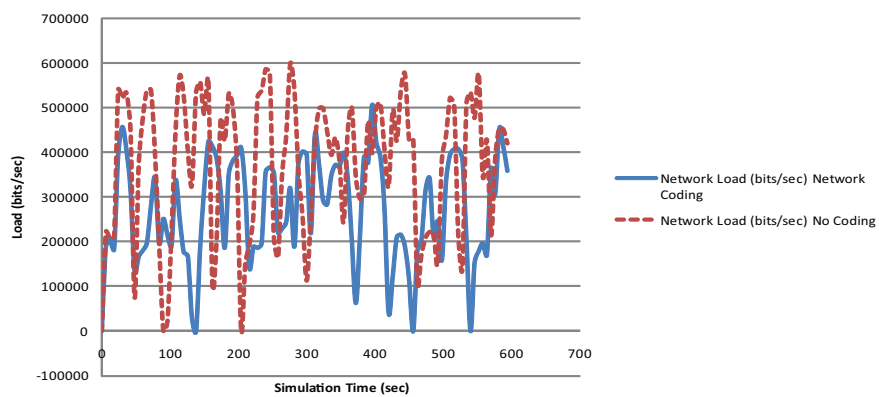


Figure 7.22: Global load, hybrid butterfly simulation

creased by 25.721% from an average of 4.2486 ms to 5.3414 ms when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.23.

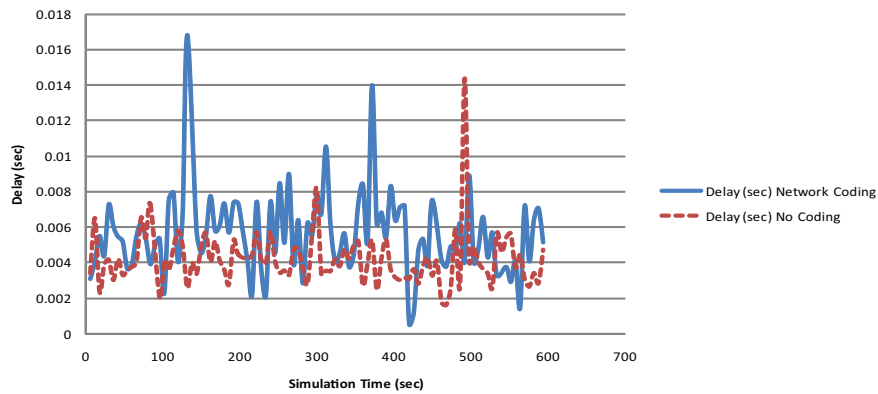


Figure 7.23: Global end-to-end delay, hybrid butterfly simulation

When the two source nodes each transmit a variable bit stream, the media access delay is increased by 41.823% from an average of 2.8984 ms to 4.1106 ms when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 7.24.

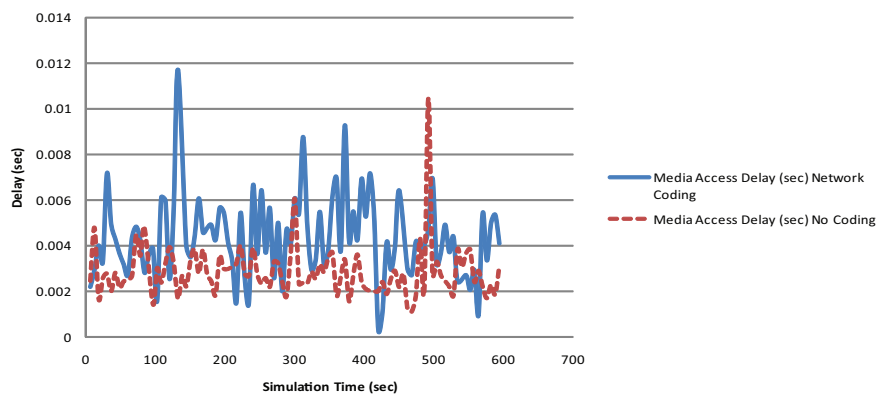


Figure 7.24: Global media access delay, hybrid butterfly simulation



## 7.2.4 Summary of results - Variable bit stream

A summary of the simulation results obtained when a variable bit stream is used as source, are shown in table 7.3.

Table 7.3: Simulation results for a variable bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Difference in Source Load 1:	5.75%	0.47%	17.72%
Difference in Source Load 2:	8.92%	2.004%	5.94%
Global Load:	4.97% decrease	33.81% decrease	27.63% decrease
Global Delay:	0.32% increase	110.77% increase	25.721% increase
Media Access Delay:	0.1% decrease	186.989% increase	41.82% increase

## 7.2.5 Variation of packet delay - Variable bit stream

The variation of IP packet delay (jitter) measured for each topology simulated, is shown in table 7.4. The difference in global jitter for when network coding is implemented and when network coding is not implemented, can be compared for the different topologies.

Table 7.4: Variation of IP packet delay for a variable bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Network Coding Implemented			
Maximum:	6.34 ms	9.96 ms	9.95 ms
Average:	1.74 ms	3.02 ms	2.30 ms
No Network Coding:			
Maximum:	4.80 ms	4.47 ms	11.63 ms
Average:	1.40 ms	0.92 ms	1.39 ms
Difference in average:	24.29% increase	228.26% increase	65.47% increase

## 7.2.6 Longer simulation time

As was mentioned earlier in this section, when the two source nodes each transmit a variable bit stream, OPNET generates a random bit stream that is unique to every simulation instance. This makes comparing the simulations difficult. Referring to table 7.3 it can be seen that the average load introduced into the network by the source nodes varies as much as 15.05% between simulations that we compare.

In order to be able to compare the different simulation results when a variable bit stream is used, we have increased the simulation time to 10 hours to reduce the difference in average load induced into the network between simulations. The results of this simulations are shown in table 7.5.

Table 7.5: Results: Variable bit stream - Longer simulation time

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Difference in Source Load 1:	negligible	negligible	1.43%
Difference in Source Load 2:	negligible	negligible	3.63%
Global Load:	3.79% decrease	33.21% decrease	22.11% decrease
Global Delay:	2.83% increase	100.53% increase	50.76% increase
Media Access Delay:	3.35% increase	172.10% increase	74.75% increase

## 7.2.7 Variation of packet delay - Longer simulation time

The variation of IP packet delay, or jitter, measured for each topology simulated, is shown in table 7.6 for the longer simulation time. The difference in global jitter for when network coding is implemented and when network coding is not implemented, can be compared for the different topologies.

Table 7.6: Variation of packet delay: Variable bit stream - Longer simulation time

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Network Coding Implemented			
Maximum:	785.57 $\mu$ s	1.07 ms	1.01 ms
Average:	235.16 $\mu$ s	355.21 $\mu$ s	303.65 $\mu$ s
No Network Coding:			
Maximum:	748.95 $\mu$ s	785.41 $\mu$	596.06 $\mu$ s
Average:	202.76 $\mu$ s	147.65 $\mu$ s	172.53 $\mu$ s
Difference in average:	15.98% increase	140.58% increase	76% increase

### 7.3 Conclusion

In this chapter, we presented the network coding simulation results for the bow-tie, butterfly and hybrid butterfly topologies. A detailed discussion of the results will follow in chapter 10 where the simulation and implementation results will also be compared. In the next chapter, we will give a detailed explanation of the implementation method and the environment used.

# Chapter 8

## Implementation of identified topologies

---

*In this chapter, we replicate the simulations done in the previous chapter on a six node test bed, using Click modular router in the Linux environment. We give an introduction to Click, describe the wireless testbed setup used and discuss the procedure followed to port the developed network coding algorithm from the OPNET simulation environment to functional code in Click.*

---

### 8.1 Introduction to Click modular router

Click modular router is a toolkit for writing modular packet processors, where a packet is a unit of network data routed between a origin and a destination on any network. Each packet processor or router is highly flexible and configurable [25, 26]. To implement network coding on wireless nodes, we used Click modular router to write a custom network stack to process the received network packets from the wireless interface of a node.

The operation of Click is centered around elements, where an element is a simple software component performing a simple task. These simple tasks include examining or modifying packets in simple ways, as for example packet classification, queuing and interfacing with network devices. Elements are connected by edges and pass packets to one another over links called connections. A collection of connected elements, where each incoming network packet can follow a possible path, is called a router configuration. A router configuration can be run in userspace or in kernelspace [25].

### 8.1.1 Elements and connections

Elements are very simple software components connected by their edges. A connection is formed from an output port to an input port. Input ports are interfaces where packets arrive and are graphically represented by triangles. Output ports are interfaces where packets leave elements and are graphically represented by squares. Input and output ports can be one of three types [32]:

1. Push port - The source must initiate packet transfer. It is graphically represented as a filled square or triangle.
2. Pull port - The destination must initiate packet transfer. It is graphically represented as an empty square or triangle.
3. Agnostic port - This type of port can act as a push or pull port. It is graphically represented as a square in a square or triangle in a triangle.

A push port must be connected to another push or agnostic port and a pull port must be connected to another pull or agnostic port.

An example of an element is given in [25] and shown in figure 8.1.

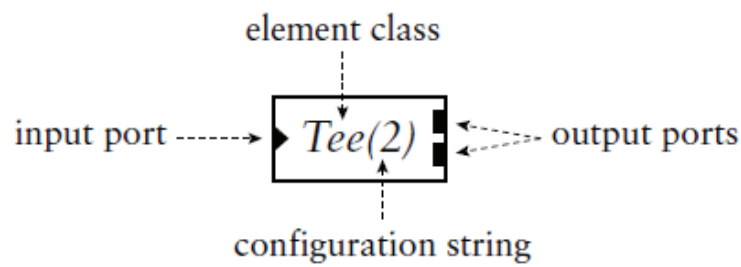


Figure 8.1: Sample element

### 8.1.2 Router configurations

A router configuration is a collection of connected elements. An example of a simple router configuration is given in [25] and shown in figure 8.2. To start Click modular router on a node, Click is initiated with a Click script as input. An example of a Click script is given below. It is a copy of the script that would realize the router configuration shown in figure 8.2.

```
FromDevice(eth0)
-> Counter
-> Discard;
```

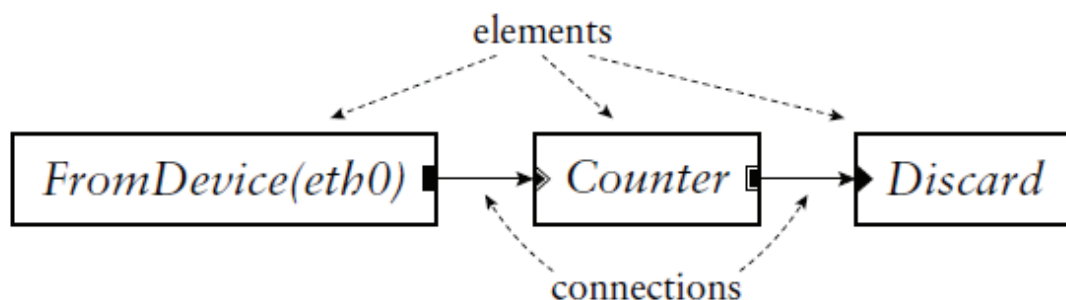


Figure 8.2: Sample router configuration

## 8.2 Wireless testbed setup

A four node wireless testbed was available in our laboratory for the implementation phase of this project. The testbed was previously created and verified by an engineering student at the North-West University [33]. The testbed was expanded to a total of six nodes by making two clones of one of the already existing nodes, creating a six node wireless testbed with nodes identical in all hardware and software aspects. An overview of the physical layout of the testbed is provided in figure 8.3.

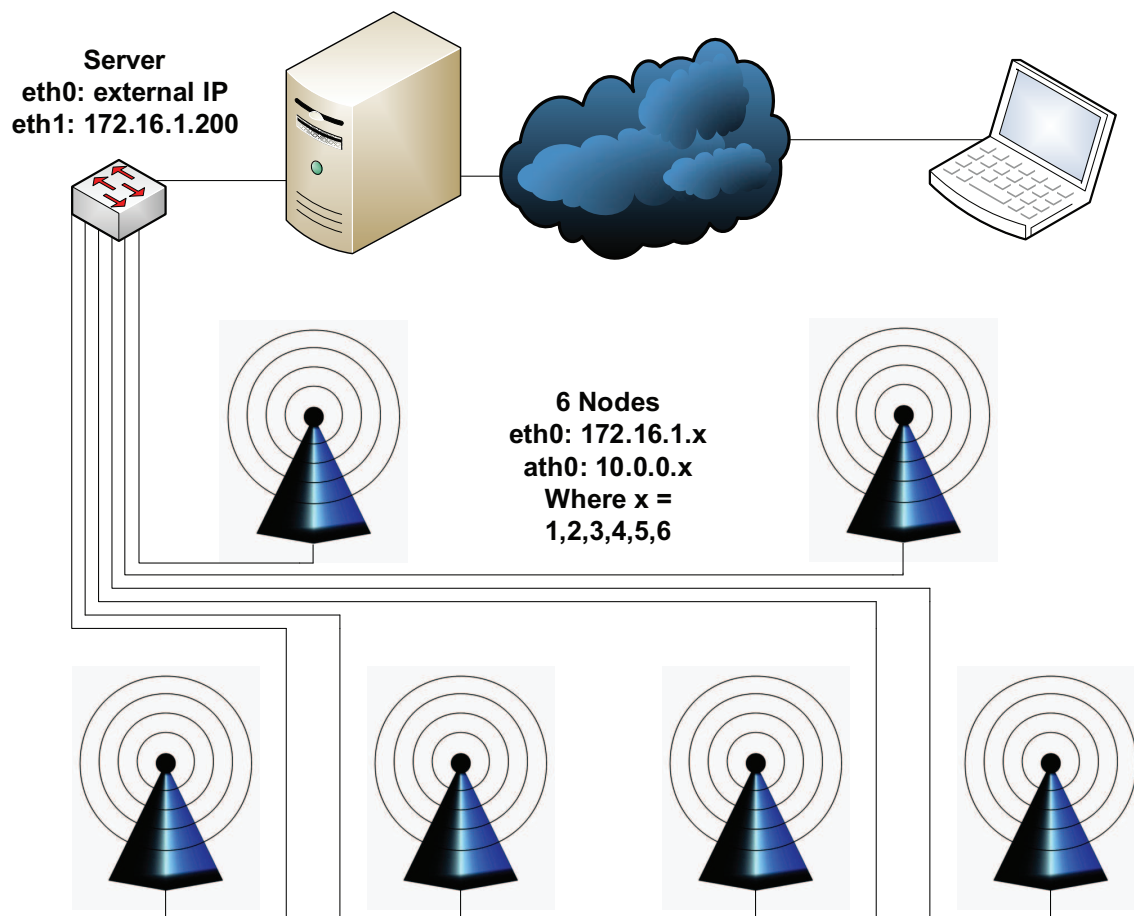


Figure 8.3: Testbed layout

### 8.2.1 Testbed hardware architecture

All the nodes in the testbed are constructed with the following hardware:

- Pentium 4 1.8 GHz Central Processing Unit (CPU)
- Intel 845G chipset
- 512 MB Double Data Rate (DDR) Random Access Memory (RAM)
- 20 GB hard drive
- 3Com 100 Mbps Ethernet interface
- Wistron NeWeb CM9-GP mini-Peripheral Component Interconnect (PCI) 802.11 a/b/g module
- MikroTik RouterBOARD RB11 PCI to mini-PCI adapter
- Aeroflex Inmet 6AH-30, 30 dB attenuator
- Planet HPA 110 4 dBi half wave dipole antenna

### 8.2.2 Testbed software architecture

All the nodes in the testbed used the Fedora Core 8 Linux distribution as an operating system. Each node can boot to two different kernel versions:

- Kernel version 2.6.23
- Kernel version 2.6.24.7, compiled with kernel-space support for the Click modular router

In this project, we decided to use the user space version of Click modular router to decrease the complexity of coding and implementing new Click elements. We therefore only used kernel version 2.6.23 during our implementation experiments.



Click modular router version 1.6, using the user space driver, was used on all the wireless nodes. The driver used to communicate with the WLAN hardware was the MAD-WIFI driver version 0.9.4 revision 3941.

The testbed nodes are remotely configurable and controllable through a server connected to all the nodes via an Ethernet backbone as shown in figure 8.3. To administer the nodes remotely, the user can use *netcat* and *ssh* to log in to each node and execute commands. By using bash scripting, the user can automate the setup and data collection of each experiment.

### 8.2.3 Testbed experimental parameters

The hardware and software parameters of each node as used in all experiments are shown in table 8.1.

Table 8.1: Experimental parameters

Parameter	Value
Antenna Diversity	Off
TX/RX Antenna	1
TX Power	13dBm
Unicast Rate	Locked to 11 Mbps
Broadcast Rate	Locked to 11 Mbps
MTU	1500 bytes
Fragmentation	Off
RTS/CTS Mechanism	Off
Mode of operation	Monitor mode

Antenna diversity is used when the network interface card has two physically separated antennas. This increases the successful transmission and reception rate. Each node in the testbed was fitted with one antenna, thus this option was disabled. To avoid the hidden node problem and ensure that all the nodes in the topology can receive each other's transmissions, the transmission power of all the nodes was set to 13dBm.

The unicast and broadcast rate of all the nodes were locked to 11 Mbps to eliminate the effect that the rate control mechanism of the MADWIFI driver would have on the measured implementation results.

The Maximum Transmission Unit (MTU) was set to the Ethernet default of 1500 bytes. Packet fragmentation and the RTS/CTS mechanism were switched off to ensure the maximum performance of the network coding algorithm on a per packet basis.

All the nodes were set to monitor mode. This enabled each node to receive all the packets transmitted by the other nodes in the topology. This enabled the use of MAC filtering to impose the different network coding suitable topologies in the network.

### 8.3 Created Click elements

To successfully implement the developed network coding scheme as used in the OP-NET simulations described in chapters 6 and 7, we had to create two new click elements. The first element created was the "NetworkCoding" element, used to enable the coding and decoding of incoming packets. The second created element was the "HostEtherFilter2Addr" element, used to filter incoming packets according to their MAC addresses.

The implementation code for these two elements is important for future development and expansion of our network coding scheme, therefore we included the most important code contained in the "NetworkCoding.cc" and "HostEtherFilter2Addr.cc" files in appendix E. Refer to appendix F for the complete code files, their headers and the Click scripts used.

### 8.3.1 The "NetworkCoding" element

The "NetworkCoding" element we created, has three primary functions: Code incoming packets, decode incoming packets and forward packets without alteration. The "NetworkCoding" element must be placed on top of the LLC layer within the data link layer to enable the successful operation thereof. This is the same position in the stack as where the new network coding layer was placed during the simulations as described in chapter 6.

When a node contains the "NetworkCoding" element set to encode incoming packets, network coding will take place according to the same algorithm used during the simulations. This algorithm is described in depth in figures 6.9 and 6.10. When a node contains the "NetworkCoding" element set to decode incoming packets, network decoding will take place according to the same algorithm used during the simulations. This algorithm is described in depth in figures 6.11 and 6.12.

In some situations it is necessary for a node to forward packets to other nodes without changing the contents, for example the bottom middle node in the butterfly topology. The "NetworkCoding" element can also be used to accomplish this function in the correct setting. The "NetworkCoding" element is shown below and graphically depicted in figure 8.4.

```
NetworkCoding(NODEFUNCTION 2, THISNODEADDR $NODE_MAC,  
CODINGDESTADDR $CODE, CODINGFWDDESTADDR $CODEFWD,  
MULTICASTCODINGADDR1 $MCA1, MULTICASTCODINGADDR2 $MCA2)
```

where

- NODEFUNCTION can be:
  - 1 = Decode packets
  - 2 = Encode packets
  - 3 = Forward packets

- THISNODEADDR is the MAC address of the current node.
- CODINGDESTADDR is the MAC address of the coded packets.
- CODINGFWDDESTADDR is the MAC address of coded packets which must only be forwarded (used in the butterfly topology).
- MULTICASTCODINGADDR1 is the first multicast coding address as used in the simulations.
- MULTICASTCODINGADDR2 is the second multicast coding address as used in the simulations.

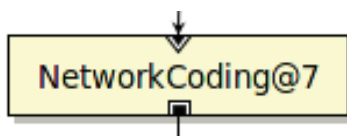


Figure 8.4: Network coding element

### 8.3.2 The "HostEtherFilter2Addr" element

The "HostEtherFilter2Addr" element we created, has two primary functions: Filter incoming packets according to their source MAC addresses and filter incoming packets according to their destination MAC addresses. Due to spacial constraints in the laboratory, we could not position the wireless nodes in such a manner as to impose the different network coding suitable topologies needed to complete the implementation phase of this project. We therefore made sure that the transmission power of all the nodes were sufficient to enable all nodes to communicate, switched all nodes to monitor mode and used MAC filtering to impose the different topologies. The "HostEtherFilter2Addr" element was needed to accomplish this task of filtering incoming wireless network packets. The "HostEtherFilter2Addr" element is shown below and graphically depicted in figure 8.5.

```
HostEtherFilter2Addr(SOURCE_DEST 1, ETHER1 $NODE1MAC,  
ETHER2 $NODE2MAC)
```

where

- SOURCE\_DEST can be:
  - 1 = Filter by source MAC addresses
  - 2 = Filter by destination MAC addresses
- ETHER1 is the first source/destination MAC address to filter by.
- ETHER2 is the second source/destination MAC address to filter by.

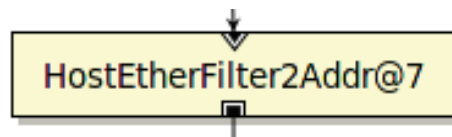


Figure 8.5: MAC address filter element

## 8.4 Click setup for each node

To automate the distribution of new Click scripts to the wireless nodes, a bash script was written which gathers information of each node and inserts this information, along with some static information, at the top of each Click script used in the testbed. A summary of the information inserted into each Click script, is shown in appendix D.1.

### 8.4.1 Source nodes

The Click script for the two source nodes (nodes one and two in all topologies), is shown in appendix D.2 and depicted in figure 8.6.

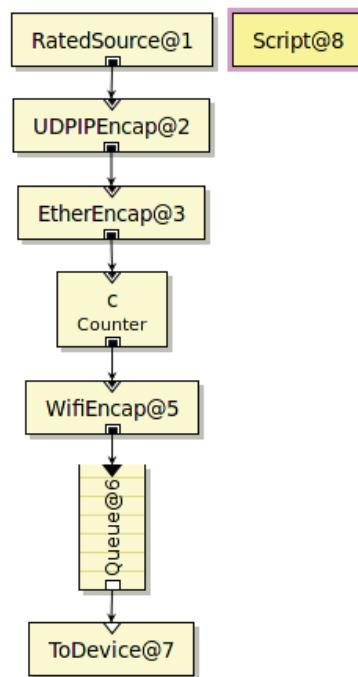


Figure 8.6: Source node

### 8.4.2 Coding node

The Click script for the coding node (node five in all topologies), is shown in appendix D.3 and depicted in figure 8.7(a).

### 8.4.3 Forwarding node

The Click script for the forwarding node (node six in the butterfly and hybrid butterfly topologies), is shown in appendix D.4 and depicted in figure 8.7(b).

### 8.4.4 Receiving nodes

The Click script for the two receiving nodes (nodes three and four in all topologies), is shown appendix D.5 and depicted in figure 8.7(c).

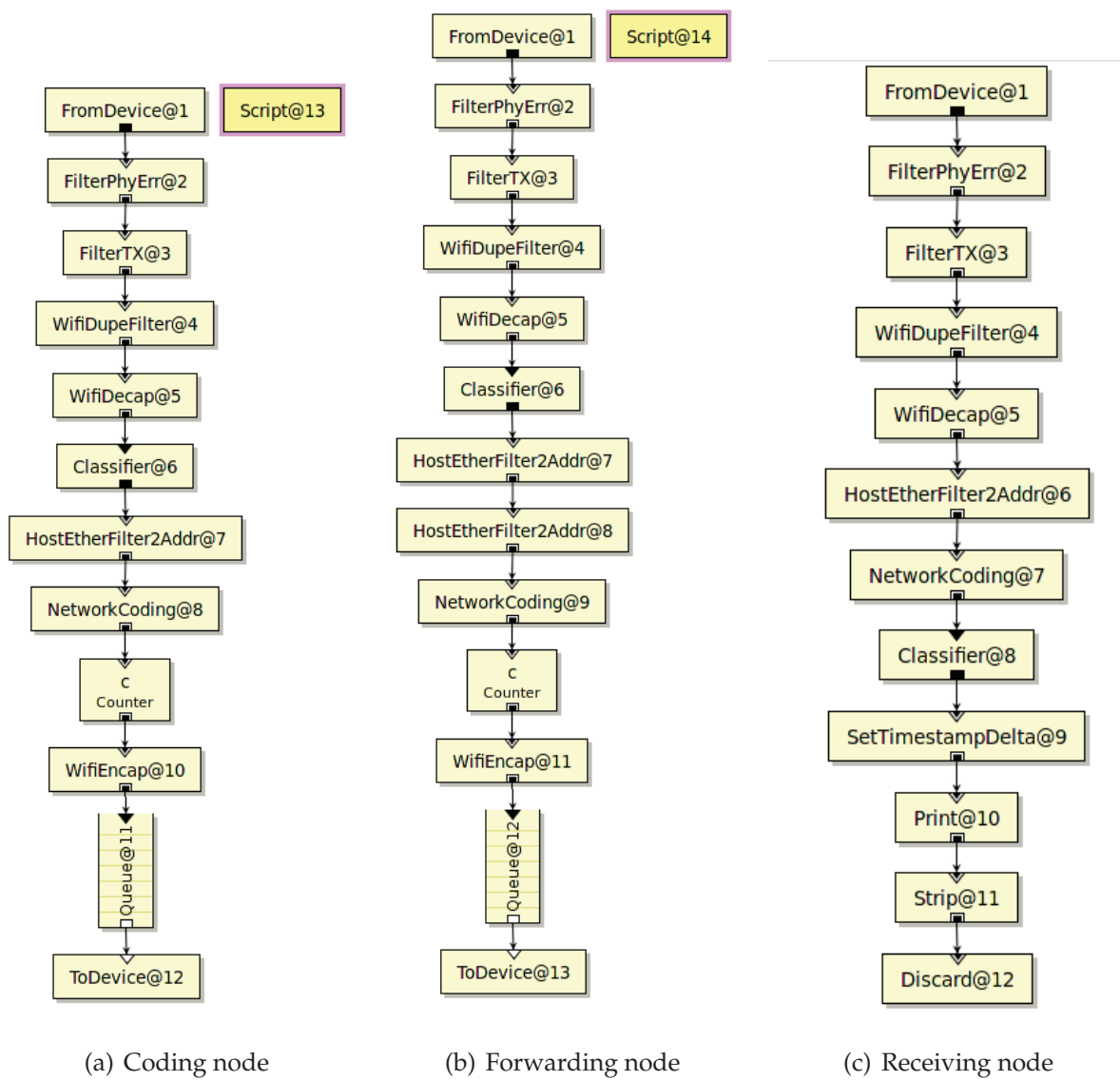


Figure 8.7: Click setup for each node

## **8.5 Conclusion**

In this chapter, we presented an introduction to Click modular router and the physical testbed used for the implementation phase. Special attention was given to the procedure followed to implement the developed network coding algorithm in Click modular router. The results of the implementation phase are given in the next chapter.



# Chapter 9

## Implementation results

---

*In this chapter, we present the implementation results obtained for the bow-tie, butterfly and hybrid butterfly network coding suitable topologies. We use a six node testbed with MAC filtering to force the creation of the different topologies and create one simulation scenario where the source nodes transmit data at a constant and synchronized rate to the destination nodes.*

---

### 9.1 Constant bit stream as source

Measurement results of implementing our developed network coding algorithm with two source nodes, sending 100 packets per second in one direction through the network for a time of 60 seconds, are presented in the next subsections.

### 9.1.1 Bow-tie topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 25.65% from an average of 216.200 kbits/sec to 160.741 kbits/sec when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.1.

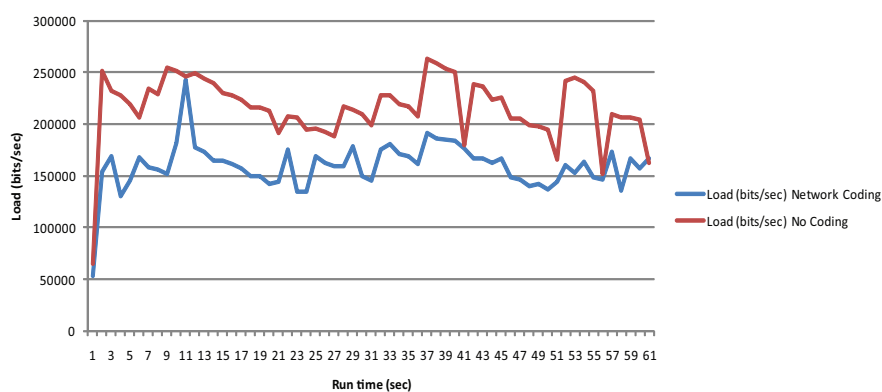


Figure 9.1: Global load, bow-tie implementation

When the two source nodes each transmit a constant bit stream, the global delay is increased by 3.4769 ms from an average of 0.0231 ms to 3.5000 ms when network coding is implemented in the bow-tie topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.2 where network coding is implemented and figure 9.3 where network coding is not implemented.

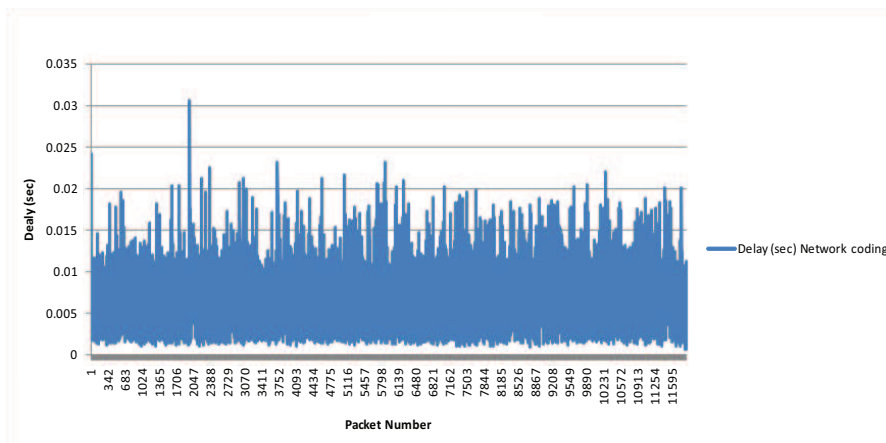


Figure 9.2: Global end-to-end delay, bow-tie implementation: NC

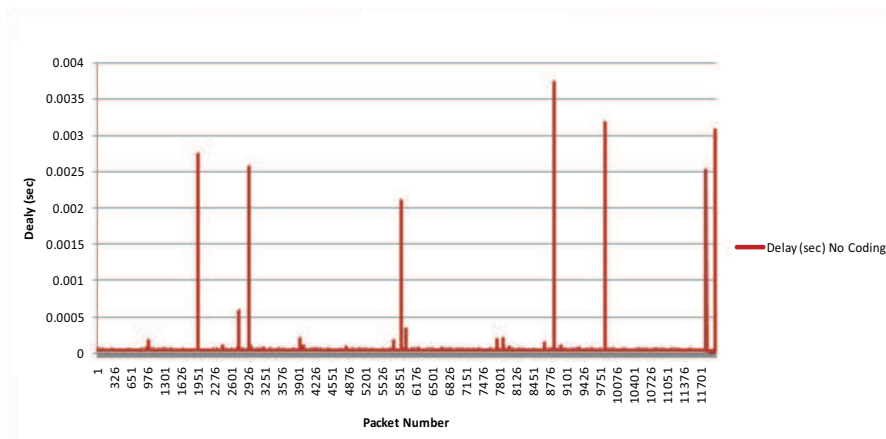


Figure 9.3: Global end-to-end delay, bow-tie implementation: No NC

### 9.1.2 Butterfly topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 36.44% from an average of 316.303 kbits/sec to 201.046 kbits/sec when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.4.

When the two source nodes each transmit a constant bit stream, the global delay is

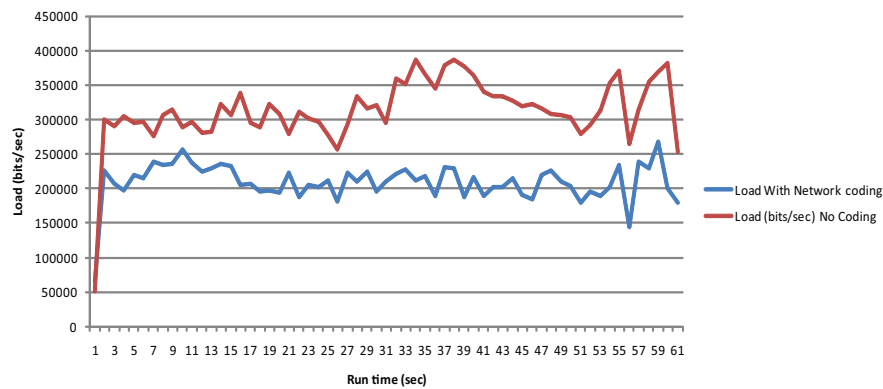


Figure 9.4: Global load, butterfly implementation

increased by 11.0356 ms from an average of 0.0455 ms to 11.0811 ms when network coding is implemented in the butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.5 where network coding is implemented and figure 9.6 where network coding is not implemented.

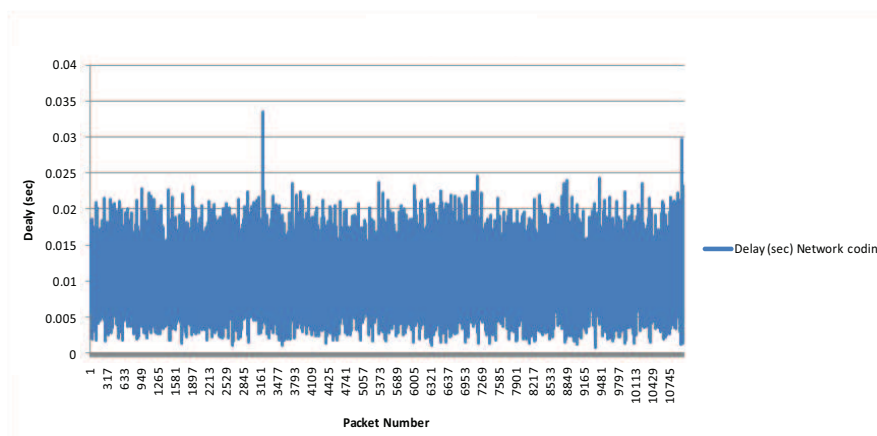


Figure 9.5: Global end-to-end delay, butterfly implementation: NC

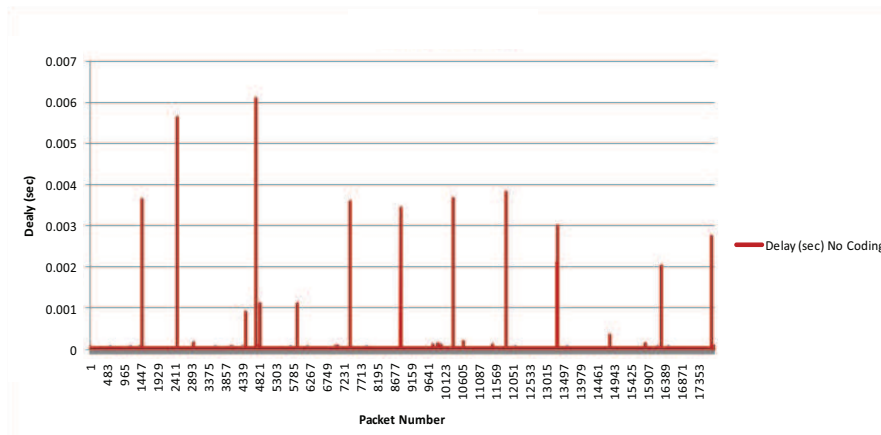


Figure 9.6: Global end-to-end delay, butterfly implementation: No NC

### 9.1.3 Hybrid butterfly topology

When the two source nodes each transmit a constant bit stream, the global load is decreased by 25.77% from an average of 244.149 kbits/sec to 181.243 kbits/sec when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.7.

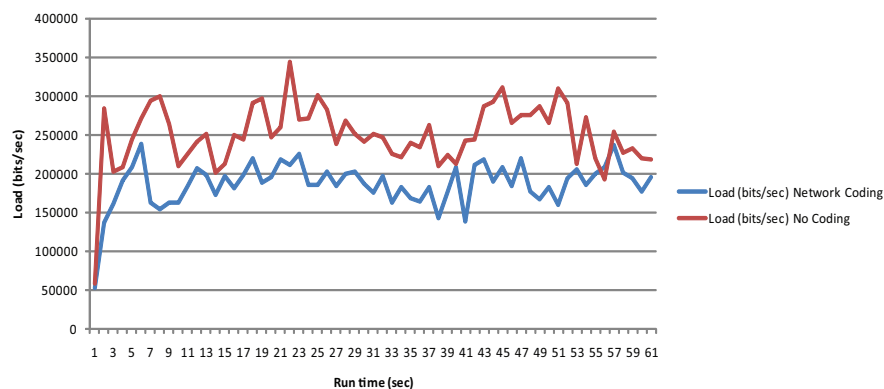


Figure 9.7: Global load, hybrid butterfly implementation

When the two source nodes each transmit a constant bit stream, the global delay is

increased by 21.3669 ms from an average of 0.0324 ms to 21.3933 ms when network coding is implemented in the hybrid butterfly topology compared to when traditional communication methods are used. The applicable results of the simulation are shown in figure 9.8 where network coding is implemented and figure 9.9 where network coding is not implemented.

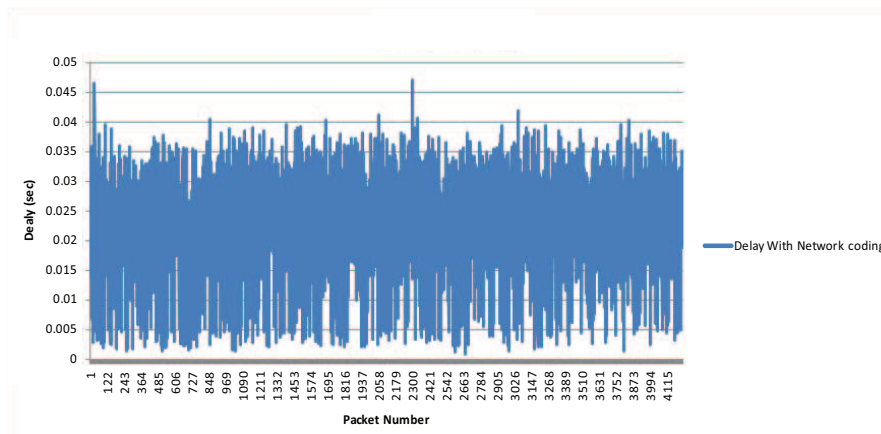


Figure 9.8: Global end-to-end delay, hybrid butterfly implementation: NC

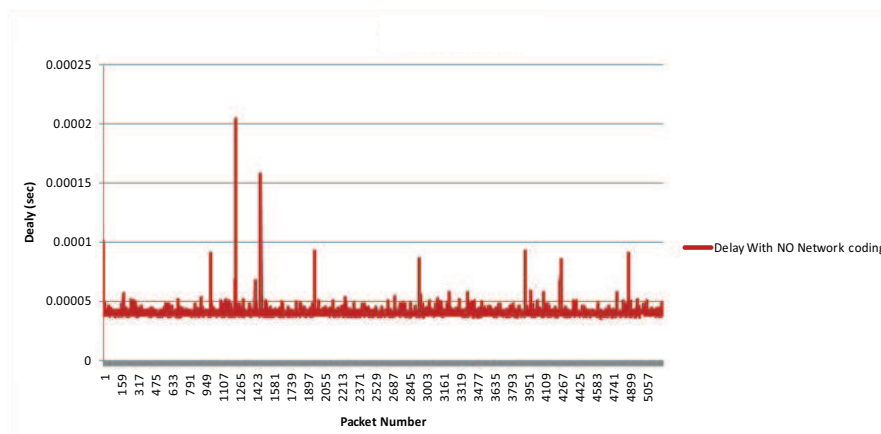


Figure 9.9: Global end-to-end delay, hybrid butterfly implementation: No NC

### 9.1.4 Summary of results - Constant bit stream

A summary of the implementation results obtained when a constant bit stream is used as source, are shown in table 9.1.

Table 9.1: Implementation results for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Global Load:	25.65% decrease	36.44% decrease	25.77% decrease
Global Delay:	3.4769 ms increase from 0.0231 ms to 3.5000 ms	11.0356 ms increase from 0.0455 ms to 11.0811 ms	21.3669 ms increase from 0.0324 ms to 21.3933 ms

### 9.1.5 Variation of packet delay

The variation of IP packet delay (jitter) measured for each topology, is shown in table 9.2. The difference in global jitter for when network coding is implemented and when network coding is not implemented, can be compared for the different topologies.

Table 9.2: Variation of IP packet delay for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Network Coding Implemented Average:	28.92 ms	31.01 ms	43.72 ms
No Network Coding Average:	13.09 ms	13.99 ms	4.91 ms
Difference in average:	120.93% increase	121.66% increase	790.43% increase

## 9.2 Conclusion

In this chapter, we presented the results of the implementation phase. In the next chapter, the results are discussed in detail and are compared to the simulation results

obtained in chapter 7.



# Chapter 10

## Discussion of results

---

*In this chapter, we present a detailed discussion and comparison of the theoretical, simulation and implementation results presented in previous chapters.*

---

### 10.1 Discussion of node placement calculation results

In chapter 3, we identified different topologies suitable for the implementation of network coding. In chapter 4, we used the free space attenuation equation to calculate the theoretical communication distance of nodes communicating at different transmission rates. We used these distances to form communication areas in which each node must be located to enable the successful implementation of network coding. In chapter 5, we modified these communication areas to be of practical use by substituting the free space attenuation equation in our calculations, with a practical path-loss model. We then discussed how nodes can be moved around within these communication areas and how this movement of nodes creates new communication areas for the rest of the nodes within the topology.

By using physical node positions as input for the mathematical model, we can determine whether a practical network, containing one or more of the identified network coding suitable topologies, can implement network coding and reduce the total load in the network successfully.

We used the optimal node positions, as calculated with the mathematical model, in the simulations where each of the identified topologies were reconstructed accordingly and network coding simulated. The results obtained in the simulation phase are discussed in the next section.

## 10.2 Discussion of simulation results

### 10.2.1 Constant bit stream

We simulated three different network coding suitable topologies to test and evaluate the created network coding scheme. We initially subjected the different topologies to a constant bit stream generated by two source nodes. Consequently, traffic aggregated at a constant rate through the network. The intermediate or coding nodes determined whether network coding was possible or not by evaluating the source and destination MAC addresses of the different network packets that accumulated in the node buffers. The most significant results collected from these simulations are shown in table 10.1.

Table 10.1: Simulation results for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Global Load:	24.97% decrease	33.33% decrease	20% decrease
Global Delay:	1.27% increase	13.09% decrease	2.54% decrease
Media Access Delay:	14.91% increase	15.55% decrease	55.73% increase

### **Global load**

The decrease of the global load is quite significant for all the topologies. This is because of the constant and synchronized manner in which data arrived at the coding and destination nodes when a constant bit stream is used as source. The amount of transmissions were reduced, translating into a reduction in global network load for the same throughput.

The decrease in global load is the most when network coding is implemented in the butterfly topology. This can be explained by taking into account that the total number of transmissions is reduced by two when network coding is implemented in the butterfly topology. The total number of transmissions is only reduced by one in the bow-tie and hybrid butterfly topologies, resulting in a similar decrease in global load in these two topologies of about 20%.

### **Global delay and media access delay**

In the bow-tie topology, the global delay of packets successfully received by nodes and the media access delay of packets waiting to be transmitted by nodes, were higher with network coding enabled than with it disabled. Data packets must wait in a buffer at the smart node for another network coding suitable packet to arrive before the network coding algorithm can decide whether network coding can take place or not. The delay increase is however lower than what is expected because of the reduced number of packets that must be sent to the receiver nodes, translating into reduced contention for the shared communication medium.

For the butterfly topology, the global delay of packets successfully received by nodes and the media access delay of packets waiting to be transmitted by nodes, were lower with network coding enabled than with it disabled. This can be explained by taking into account that the total number of transmissions is reduced by two, thereby also reducing the time packets wait to access the shared communication medium and coun-

tering the effect the buffers at the coding and decoding nodes have on the end-to-end delay of transmitted packets.

For the hybrid butterfly topology, the global delay of packets successfully received by nodes, were only slightly lower with network coding enabled than with it disabled. The media access delay increase was much higher than what was experienced in the bow-tie topology. As with the bow-tie topology, the number of transmissions is reduced only by one, thus the delay decrease will not be as significant as in the butterfly topology.

### **Variation of IP packet delay**

The usage of network coding did not affect the variation of IP packet delay or jitter considerably. The average jitter did not vary by more than 10  $\mu$ s for each of the simulated topologies.

## **10.2.2 Variable bit stream**

In the second set of simulations, different topologies were subjected to a variable bit stream generated by the two source nodes, to evaluate the degree of network coding that takes place in situations where the possibility for network coding opportunities to arise are extremely limited. Traffic arrived unsynchronized and in random order at the intermediate node, which then determined when network coding can take place by evaluating the source and destination addresses of the different network packets located in the node's buffer.

The random generation of variable traffic makes it extremely difficult to compare different simulations. This is because of the random load induced by the source nodes. For example, the average load on the network incurred by one source node in the hybrid butterfly topology, differed by 15% from one simulation to the next. Therefore it will not be possible compare different simulation results and deduce valuable and

accurate conclusions. The main reason for the usage of a variable bit stream as source in our simulations, was to evaluate the coding scheme's performance in more realistic environments where networks relay bursty traffic.

The same trends of a reduced global load and increased global and media access delays were seen when a variable bit stream was used in comparison to the usage of a constant bit stream as source. We can therefore deduce that the usage of our network coding scheme can be of value when implemented in networks where very random bursts of data at different rates occur. The only drawback is that the usage of this scheme in such network environments will induce extra delay.

### 10.2.3 Variable bit stream - Longer simulation time

In order to compare the different simulation results when a variable bit stream is used as source, we have increased the simulation time to 10 hours, reducing the difference in average load induced into the network between simulations. The most significant results collected from these simulations are shown in table 10.2.

Table 10.2: Results for a variable bit stream - Longer simulation time

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Difference in Source Load 1:	negligible	negligible	1.43%
Difference in Source Load 2:	negligible	negligible	3.63%
Global Load:	3.79% decrease	33.21% decrease	22.11% decrease
Global Delay:	2.83% increase	100.53% increase	50.76% increase
Media Access Delay:	3.35% increase	172.10% increase	74.75% increase

#### Global load

The decrease of the global load is quite significant for all of the topologies, except for the bow-tie topology. The amount of transmissions is reduced translating to a reduc-

tion in global network load for the same throughput. It seems that the bow-tie topology is more sensitive to the synchronization of incoming packets regarding global load. The decrease in global load for the other topologies compares to the reduction of global load seen when a constant bit stream was used.

### **Global delay and media access delay**

The global load of the bow-tie topology decreased only by 3.79%, which means that network coding did not take place on a regular basis, therefore the global delay and media access delay only varied by 2.83% and 3.35%, meaning that the small amount of network coding did not affect the delays induced on packets significantly.

In the butterfly and hybrid butterfly topologies, there were a significant reduction in global load, indicating that network packets were coded together on a regular basis. The variable and unsynchronized way in which data arrived at the coding and destination nodes, caused packets to wait for suitable coding and decoding packets on some occasions, translating into the large increase in delays induced on the packets. The time packets wait for suitable packets to arrive is limited by the buffer "flush" time which was set to ensure the maximum amount of packets were network coded. The buffer "flush" time can be reduced to limit the impact of the buffers on the global delay, but this will reduce the network coding opportunities found.

### **Variation of IP packet delay**

The variation of IP packet delay increased with the usage of network coding in all of the topologies. The butterfly and hybrid butterfly topologies experienced a larger increase in jitter compared to the bow-tie topology, caused by longer buffer waiting periods induced by the network coding algorithm.

### 10.3 Discussion of implementation results

In order to determine whether the developed network coding scheme will function in reality and produce desirable results, we implemented the network coding algorithm on a six node testbed. The results of the implementation phase give a more realistic estimation of how network coding will affect real life networks.

We forced the creation of the different topologies by using MAC filtering. The created topologies were then subjected to a constant bit stream, generated by two source nodes. The network coding algorithm was implemented exactly the same as in the simulations. The most significant results collected from these simulations are shown in table 10.3.

Table 10.3: Implementation results for a constant bit stream as source

Topology Type:	Bow-tie	Butterfly	Hybrid Butterfly
Global Load:	25.65% decrease	36.44% decrease	25.77% decrease
Global Delay:	3.4769 ms increase from 0.0231 ms to 3.5000 ms	11.0356 ms increase from 0.0455 ms to 11.0811 ms	21.3669 ms increase from 0.0324 ms to 21.3933 ms

#### Global load

The decrease of the global load is quite significant for all the topologies. The decrease in global load is the most when network coding is implemented in the butterfly topology. The bow-tie and hybrid butterfly experienced a similar decrease in global load of about 25%. These results correlate with the simulations in section 10.2.1, where the explanations are given.

## Global delay

The global end-to-end delay of all the topologies were significantly larger than what was predicted by the simulations. This can be because of a number of reasons. It may be that the processing delay of the algorithm is larger in reality than what is predicted by the simulation software. This includes the time packets have to wait in a buffer for a suitable coding or decoding packet to arrive and for coding or decoding to take place. Another explanation for the large increase in global delay, may be because of the MAC filtering used to impose the different network coding suitable topologies. Extra unused packets were received and processed by the lower layers and dropped by the MAC filter located higher up in the network stack. Besides this extra processing imposed on nodes, all the packets received by each node had to be tested by this MAC filter layer, increasing packet processing times.

The number of nodes and therefore also the number of packet "hops" in the topology, seems to have a large impact on the end-to-end delay of the network coded packets. The bow-tie topology, with the least amount of nodes, has the smallest increase in delay. The butterfly topology and the hybrid butterfly topology have the same amount of nodes and therefore a larger increase in end-to-end delay.

The number of transmissions are reduced by two in the butterfly topology, translating into reduced contention for the shared communication medium than what is experienced in the hybrid butterfly topology. The number of transmissions are only reduced by one in the hybrid butterfly topology, explaining why the increase in delay is more in the hybrid butterfly than the butterfly topology.

## Variation of IP packet delay

The variation of IP packet delay increased significantly with the usage of network coding in all of the topologies. This large increase in jitter is caused by longer buffer waiting periods induced by the network coding algorithm than what was seen in the pre-



vious section.

### 10.3.1 Comparing simulation and implementation results

The implementation and simulation results for a constant bit stream as source, as discussed in sections 10.2.1 and 10.3, show similarities in the reduction of the global load of the network. A comparison of the load reduction predicted by the simulations and what was experienced during the implementation experiments, is shown in figure 10.1.

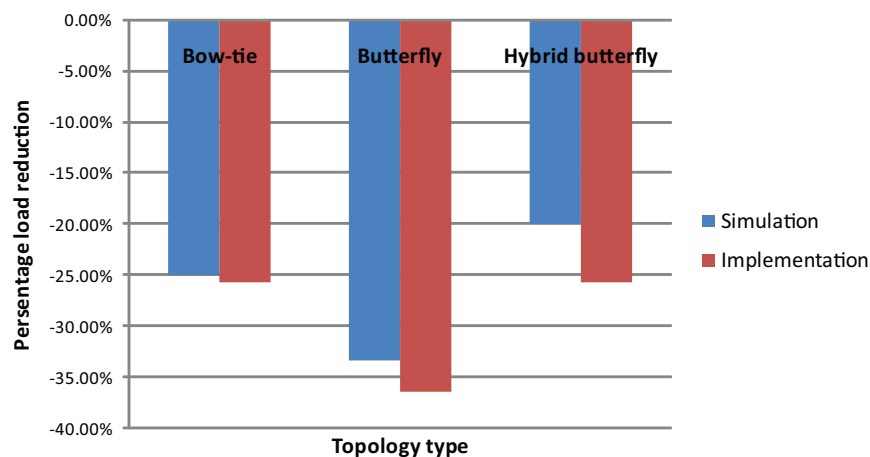


Figure 10.1: Comparison of load reduction

The increase in the end-to-end delay of network packets when network coding is used, is considerably higher during implementation as what was predicted by the simulations. For example, in the bow-tie topology, the simulation results suggested a global delay increase of 1.013 times while the implementation results suggested a global delay increase of 150 times when a constant bit stream is used as source. As described in section 10.3, the reason for this large variation in the prediction, may be because of the simulation software that underestimates the processing delay of the network coding algorithm or that the usage of MAC filtering during implementation, increases the delay imposed on packets significantly.

To our knowledge, the elements that produces variable bit rate traffic in Click modular router version 1.6, were unavailable at the time of writing, therefore we could not create an implementation scenario to compare to the simulation results obtained for a variable bit stream.

## 10.4 Conclusion

In this chapter, we discussed and compared the theoretical, simulation and implementation results. The next chapter will provide a conclusion and final remarks on the work done in this project and the results obtained.

# Chapter 11

## Conclusion

---

*In this chapter, we give an overview of the research work we presented in this dissertation. We present the overall conclusion, discuss the validation and verification of the work done and present possible future work.*

---

### 11.1 Conclusion

In this document, we presented work on network coding in wireless ad hoc networks:

- We identified network topologies that are suitable for the implementation of network coding.
- A mathematical model was created and used to calculate the boundaries of the network coding suitable topologies, the area in which each node in the topology must be located and the maximum size of the topology in which hidden nodes will not occur.

- We used the results from the mathematical model as input for our simulations and created a network coding scheme to evaluate the practicality of implementing network coding in the identified topologies.
- The simulations were recreated and implemented on a six node testbed to ensure that the simulation results served as an accurate representation as to what will happen when the developed network coding scheme is implemented in reality.

The results of our simulations, presented in chapter 10, suggests that the implementation of network coding in suitable topologies, will decrease overall network load at the cost of having nodes capable of coding and decoding packets (the XOR operation is simple but will require some increased computation time) and having a higher end-to-end delay induced on network packets. Network coding is more suitable and will produce better results when implemented in networks where the flow of data is constant and synchronized.

Comparing the results of the implementation phase to the simulation results, suggests that the OPNET simulation environment underestimates the network coding processing delay induced on packets or that the usage of MAC filtering during implementation, increases the delay imposed on packets significantly.

We can derive that the developed network coding scheme will produce better overall performance when implemented in sensor networks, highly congested ad hoc networks, delay-tolerant wireless networks or applications where the advantage of decreasing the global load is more significant than the disadvantage of the higher end-to-end delay induced on the network packets.

By reducing the number of transmissions and therefore the global load of a network, congestion can be reduced, energy can be saved and therefore battery life of sensor nodes increased. Another advantage of implementing network coding, is that the available bandwidth is used more efficiently.

## 11.2 Validation and verification

The verification of the results obtained in this dissertation was done by comparing the theoretical and practical results with one another and other coding schemes.

Validation of the results was done as described in chapter 1. The publication of papers are especially important for validation.

### 11.2.1 Comparing results to theory

The implementation of network coding in the bow-tie topology reduces the total number of transmissions from 4 to 3 resulting in a maximum theoretical reduction in global load of 25%. In the butterfly topology, the total number of transmissions is reduced from 6 to 4 resulting in a 33.33% reduction and in the hybrid butterfly topology, 5 transmissions are reduced to 4, resulting in a 20% theoretical reduction. These theoretical results almost exactly correlate with the simulation results obtained for a constant bit stream as source and is comparable to implementation results, proving that the developed network coding algorithm functions as expected.

### 11.2.2 Comparison of coding scheme to other schemes

It is difficult to compare different network coding schemes that were not developed for the same purpose. Published measurements and results should also be comparable.

COPE was developed for unicast traffic and not multicast traffic as in our experiments. The X topology used in COPE compares to bow-tie topology and reports a coding gain slightly lower than theoretically possible [7] which compares to the load reduction seen in our implementations. However, no published results indicate how the implementation of cope affects the end-to-end delay of packets.

### **11.2.3 Published papers**

Refer to appendix A. Three papers were submitted and accepted at conferences, the first being a work-in-progress and the last two full papers. The feedback received from reviewers and delegates served as valuable input and aided in the production of relevant and quality research.

## **11.3 Future work**

Future work includes the development of a routing protocol, enabling the dynamic identification of network coding suitable topologies within larger ad hoc networks and the determination of the position and function of each node in these topologies.

Further research on the impact of synchronization on this network coding scheme is necessary to enable the deployment of the scheme in real life situations.

# Bibliography

- [1] D. Faria, "Modeling signal attenuation in IEEE 802.11 wireless lans-vol. 1," *Computer Science Department, Stanford University*.
- [2] *Cisco Aironet 1200 series access point*, Cisco Systems, Inc., 2004.
- [3] *Cisco Aironet 340 Series Client Adapters and Access Points In-Building Wireless Solutions*, Cisco Systems, Inc., 2000.
- [4] E. Alban and M. Magana, *Network Coding in Relay Networks*. VDM Verlag Dr. Müller, 2008.
- [5] J. Donner, "Research approaches to mobile use in the developing world: A review of the literature," *The Information Society*, vol. 24, no. 3, pp. 140–159, 2008.
- [6] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 3, pp. 497–510, 2008.
- [8] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, pp. 2235–2245, 2005.
- [9] M. Grobler, "Using topological information in opportunistic network coding," Master's thesis, Department of Engineering, North-West University, 2008.

- 
- [10] L. Grobler, A. Helberg, H. Marais, and C. Naude, "An application of deterministic network coding in manets," *Southern African Telecommunications and Networks Access Conference*, 2008.
- [11] *RESEARCH METHODOLOGY Study Guide NVMI 874*, Department of Engineering, North-West University, 2008.
- [12] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: an instant primer," *Computer Communication Review*, vol. 36, no. 1, p. 63, 2006.
- [13] S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1028–1036, 2007.
- [14] "White paper ieee 802.11g: The new mainstream wireless lan standard," Broadcom Corporation, Tech. Rep., 2003.
- [15] (2009, March) Next generation wireless. 802.11n wireless technology overview. Cisco. [web:] <http://www.cisco.com/en/US/prod/collateral/wireless/>.
- [16] C. Cheng, P. Hsiao, H. Kung, and D. Vlah, "Parallel use of multiple channels in multi-hop 802.11 wireless networks," 2006.
- [17] R. Olexa, *Implementing 802.11, 802.16 and 802.20 wireless networks: Planning, troubleshooting and operations*. Newnes, Elsevier, 2005.
- [18] A. Goldsmith, *Wireless Communications*. Cambridge University press, 2005.
- [19] D. Griffiths, *Introduction to Electrodynamics*, 3rd ed. Prentice Hall, 1999.
- [20] Y. Okumura, E. Ohmori, T. Kawano, and K. Fukuda, "Field strength and its variability in vhf and uhf land-mobile radio service," *Land-mobile communications engineering*, p. 19, 1984.
- [21] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *Vehicular Technology, IEEE Transactions on*, vol. 29, no. 3, pp. 317–325, 2006.



- 
- [22] M. Borgo, A. Zanella, P. Bisaglia, and S. Merlin, "Analysis of the hidden terminal effect in multi-rate ieee 802.11 b networks," *Proc. WPMC*, vol. 3, pp. 6–10, 2004.
- [23] *OPNET Modeler Documentation*, OPNET Technologies, 2009.
- [24] C. Demichelis and P. Chimento, *IP packet delay variation metric for IP performance metrics (IPPM)*. RFC Editor , United States, November 2002.
- [25] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.
- [26] R. Morris, E. Kohler, J. Jannotti, and M. Kaashoek, "The click modular router," *Proceedings of the seventeenth ACM symposium on Operating systems principles*, pp. 217–231, 1999.
- [27] M. Lacage, M. Manshaei, and T. Turletti, "Ieee 802.11 rate adaptation: a practical approach," *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 126–134, 2004.
- [28] *CM9 2.4/5Ghz 802.11A+B+G Wireless miniPCI Card*, DisWire, [www.diswire.com](http://www.diswire.com).
- [29] *IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Computer Society Std., 2007.
- [30] T. Rappaport, *Wireless Communications - Principles and Practice*. Prentice Hall PTR, 1996.
- [31] G. Durgin, T. Rappaport, and H. Xu, "Measurements and models for radio path loss and penetration loss in and around homes and trees at 5.85 ghz," *IEEE Transactions on Communications*, vol. 46(11), pp. 1484–1496, 1998.
- [32] B. . V. M. Braem, "Click modular router, concepts," *University of Antwerp*, 2008.

- 
- [33] M. Ferreira, "Testbed implementation of qos routing enhancements for wireless ad hoc networks," Master's thesis, Department of Engineering, North-West University, 2009.

# Appendix A

## Conference contributions from dissertation

- F.J. Böning, A.S.J. Helberg, M.J. Grobler, "Topological Arrangement of Nodes in Wireless Networks Suitable for the Implementation of Network Coding," *Southern African Telecommunications and Networks Access Conference*, September 2010
- F.J. Böning, A.S.J. Helberg, M.J. Grobler, "A Comparison of Wireless Node Topologies for Network Coding using Practical Path-loss Models," *GlobeNet International Conference on Networks, Menuires, France*, April 2010
- F.J. Böning, A.S.J. Helberg, M.J. Grobler, Work-in-progress, "Topological arrangement of nodes in wireless networks suitable for the implementation of network coding," *Southern African Telecommunications and Networks Access Conference*, September 2009

Refer to the data CD in appendix F for the pdf file of each contribution.

# Appendix B

## OPNET's radio transceiver pipeline

A detailed description of the radio transceiver pipeline used in OPNET modeler [23] is presented in this appendix. An overview of this pipeline is shown in figure 6.1 [23].

- Stage 0: At the beginning of the simulation, the kernel determines a receiver group for each transmitter. The receiver group contains all the receiver modules that are possible candidates for receiving transmissions from a specific transmitter. This stage is invoked in the beginning of a simulation and not on the transmission of each packet.
- Stage 1: This stage calculates the amount of time required for each packet to complete transmission. The result of this calculation is used to schedule an end-of-transmission event which enables the transmitter to begin the transmission of the next packet in queue, if one exists. The result of this stage and the propagation delay stage are both used to calculate the time at which a packet completes reception at the receiver node.
- Stage 2: This stage determines whether a particular receiver channel can be affected by a transmission. It is invoked once for each receiver in a receiver set identified for a transmitter (stage 0). The stage determines if the transmitted signal can reach the candidate receiver and affect it in any way. This includes desired

transmissions as well as interfering ones (such as jamming).

- Stage 3: The purpose of this stage is to classify every transmission with respect to its receiver channel. Each packet is then assigned to one of three possible categories:
  - Valid: A packet is compatible with the receiver module.
  - Noise: A packet can not be received by the receiver, but have an impact on the receiver channel's performance.
  - Ignored: A packet has no impact on a receiver channel's state or performance.
- Stage 4: The purpose of this stage is to calculate the transmitter's antenna gain based on the direction of the vector leading from the transmitter to the receiver. The result of this stage is typically used in stage 7, the computation of the received power for a transmission.
- Stage 5: In this stage, the amount of time required for a packet's signal to travel from the transmitter to the receiver. The result is generally dependent on the distance between a transmitter and a receiver node.
- Stage 6: In this stage, the receiver antenna gain is calculated identically to the calculation of the transmitter antenna gain in stage four. The result of this stage is typically used in stage 7, the computation of the received power for a transmission.
- Stage 7: The purpose of this stage is to compute the signal power of an arriving packet's signal in watts. For valid packets, the result of this stage is used to determine if the receiver is able to correctly capture the information contained in a packet. For noise, the result is used to calculate the relative strengths of valid and noise packets.
- Stage 8: This stage accounts for transmissions that arrive concurrently at the same receiver channel. The stage is invoked if a valid packet arrives at a destination

node while that node is busy receiving another packet or, vice versa, if a packet is valid and is already being received when another valid or invalid packet arrives.

- Stage 9: This stage calculates the effect of all other noise sources that is not accounted for by stage 8. These background noise sources include thermal or galactic noise, emissions from neighbouring electronics, and other unmodeled radio transmissions.
- Stage 10: This stage is invoked when a packet arrives at its destination channel and when one of the two conditions of stage 8 occurs during packet reception. The purpose of this stage is to calculate the current average power Signal to Noise Ratio (SNR) result for an arriving packet. Calculations include the results of stages 7, 8, and 9.
- Stage 11: This stage is invoked under circumstances that corresponds to those of stage 10. The purpose of this stage is to derive the expected rate of bit errors based on SNR.
- Stage 12: This stage estimates the number of bit errors in a packet segment where the bit error probability is a constant. This estimation is based on the result of stage 11 and the length of the effected segment.
- Stage 13: This stage determines if a transmitted packet can be accepted by a receiving node and forwarded to neighbouring modules or if the packet must be destroyed (dropped). This depends on whether the packet has experienced collisions, the result of stage 12 and the ability of the node to correct the induced errors.

# Appendix C

## OPNET's Kernel Procedures used

The KPs used in this project are listed below with a brief description of each as given in [23]:

- Packet functions:
  - `op_pk_create()`: "Creates a new unformatted packet with the specified bulk data size. The packet does not have any of its fields set."
  - `op_pk_send()`: "Forwards the specified packet through an output packet stream, schedules the packet's arrival at a destination module for the current simulation time and releases ownership of the packet by the invoking process."
  - `op_pk_get()`: "Obtains a pointer to a packet that has arrived on an input packet stream and removes the packet from the stream."
  - `op_pk_fd_set()`: "Assigns the value of a structure field in the specified packet as well as the type, size, and procedures for structure copying and deallocation. The field-of-interest is specified by a numeric index."
  - `op_pk_fd_get()`: "Obtains the value of a field in the specified packet. The field-of-interest is specified by a numeric index."
  - `op_pk_total_size_get()`: "Obtains the total size of the specified packet."

- ICI functions
  - `op_ici_create()`: "Creates a new ICI with a predefined structure as described by the specified ICI format."
  - `op_ici_attr_set()`: "Assigns the value of an attribute in the specified ICI."
  - `op_ici_attr_get()`: "Obtains the value of an attribute in the specified ICI."
  - `op_ici_install()`: "Establishes the specified ICI as the installed ICI, which is automatically associated with outgoing interrupts scheduled by the invoking process."
- Interrupt functions
  - `op_intrpt_strm()`: "Obtains the stream index associated with the current interrupt of the invoking process."
  - `op_intrpt_ici()`: "Obtains the ICI associated with the current interrupt."
  - `op_intrpt_schedule_self()`: "Schedules a self-interrupt for the invoking process."
  - `op_intrpt_code()`: "Obtains the numeric code associated with the current interrupt of the invoking process."
  - `op_intrpt_type()`: "Obtains the type of the current interrupt of the invoking process."
- Statistical functions
  - `op_stat_reg()`: "Returns a handle that can be used to reference a node or module statistic (local or global) from within a process model."
  - `op_stat_write()`: "Writes a (time,value) pair to the specified statistic. The value is specified as an argument to this KP and the time is the current simulation time."
- Internal Model Access functions
  - `op_ima_obj_attr_get()`: "Obtains the value of an attribute in the specified object."



- General functions
  - `op_sim_time()`: "Obtains the current simulation time."
  - `op_ev_cancel()`: "Revokes an event that has been previously scheduled."
  - `op_id_self()`: "Obtains the object ID of the surrounding processor or queue."

# Appendix D

## Click modular router scripts

### D.1 General Click script information

A summary of the general information inserted into each new Click script, is shown below:

```
define($DEV $iface)
define($BSSID 00:00:00:00:00:00)
define($NODE_IP $clickpre$x)
define($NODE_MAC $mac)
define($SOURCE_PORT 1234)
define($CODE 06:0B:6B:DB:AD:77)
define($CODEFWD 06:0B:6B:DB:AD:78)
define($NODE1MAC 06:0B:6B:DB:AD:41)
define($NODE2MAC 06:0B:6B:DB:AD:1C)
define($NODE3MAC 06:0B:6B:DB:AD:9E)
define($NODE4MAC 06:0B:6B:DB:AD:CA)
define($NODE5MAC 06:1B:B1:02:B7:70)
define($NODE6MAC 06:1B:B1:02:B7:5F)
```

if the current node number is node 1, insert the following as well:

```
define($DEST_IP $clickpre3)
define($DEST_MAC 06:0B:6B:DB:AD:9E)
define($DEST_PORT 1234)
```

if the current node number is node 2, insert the following as well:

```
define($DEST_IP $clickpre4)
define($DEST_MAC 06:0B:6B:DB:AD:CA)
define($DEST_PORT 1234)
```

where

- iface = wireless interface of node
- mac = MAC address of the current node
- clickpre = "10.0.0."

## D.2 Source nodes

The Click script for the two source nodes, is shown below and depicted in figure 8.6.

```
RatedSource("Hallo!!! - from source node", 100, -1)
-> UDPIPEncap($NODE_IP, $SOURCE_PORT, $DEST_IP, $DEST_PORT)
-> EtherEncap(0x0800, $NODE_MAC, $DEST_MAC)
-> c::Counter
-> WifiEncap(0, $BSSID)
-> Queue(10000)
-> ToDevice($DEV); Script(read c.bit_rate, wait 1, loop)
```

## D.3 Coding node

The Click script for the coding node, is shown below and depicted in figure 8.7(a).

```
FromDevice($DEV, SNIFFER false)
-> FilterPhyErr()
-> FilterTX()
-> WifiDupeFilter()
-> WifiDecap()
-> Classifier(12/0800)
-> HostEtherFilter2Addr(SOURCE_DEST 1, ETHER1 $NODE1MAC,
ETHER2 $NODE2MAC)
-> NetworkCoding(NODEFUNCTION 2, THISNODEADDR $NODE_MAC,
CODINGDESTADDR $CODE, CODINGFWDDDESTADDR $CODEFWD,
MULTICASTCODINGADDR1 $NODE3MAC,
MULTICASTCODINGADDR2 $NODE4MAC)
-> c::Counter
-> WifiEncap(0, $BSSID)
-> Queue(10000)
-> ToDevice($DEV); Script(read c.bit_rate, wait 1, loop)
```

## D.4 Forwarding node

The Click script for the forwarding node, is shown below and depicted in figure 8.7(b).

```
FromDevice($DEV, SNIFFER false)
-> FilterPhyErr()
-> FilterTX()
-> WifiDupeFilter()
```

```
-> WifiDecap()
-> Classifier(12/0800)
-> HostEtherFilter2Addr(SOURCE_DEST 1, ETHER1 $NODE5MAC,
ETHER2 $NODE5MAC)
-> HostEtherFilter2Addr(SOURCE_DEST 2, ETHER1 $NODE3MAC,
ETHER2 $NODE4MAC)
-> NetworkCoding(NODEFUNCTION 3, THISNODEADDR $NODE_MAC,
CODINGDESTADDR $CODE, CODINGFWDDDESTADDR $CODEFWD,
MULTICASTCODINGADDR1 $NODE3MAC,
MULTICASTCODINGADDR2 $NODE4MAC)
-> c::Counter
-> WifiEncap(0, $BSSID)
-> Queue(10000)
-> ToDevice($DEV); Script(read c.bit_rate, wait 1, loop)
```

## D.5 Receiving nodes

The Click script for the two receiving nodes, is shown below and depicted in figure 8.7(c).

```
FromDevice($DEV, SNIFFER false)
-> FilterPhyErr()
-> FilterTX()
-> WifiDupeFilter()
-> WifiDecap()
-> HostEtherFilter2Addr(SOURCE_DEST 1, ETHER1 $NODE1MAC,
ETHER2 $NODE6MAC)
-> NetworkCoding(NODEFUNCTION 1, THISNODEADDR $NODE_MAC,
CODINGDESTADDR $CODE, CODINGFWDDDESTADDR $CODEFWD,
```

```
MULTICASTCODINGADDR1 $NODE3MAC,  
MULTICASTCODINGADDR2 $NODE4MAC)  
-> Classifier(12/0800)  
-> SetTimestampDelta(TYPE NOW)  
-> Print(CONTENTS NONE, TIMESTAMP true)  
-> Strip(14)  
-> CheckIPHeader(VERBOSE true)  
-> IPPrint(ip, PAYLOAD ASCII)  
-> Discard;
```

# Appendix E

## Click modular router implementation code

### NetworkCoding.cc

```
#include <click/config.h>
#include <click/confparse.hh>
#include <click/error.hh>
#include <clicknet/ether.h>
#include "networkcoding.hh"

CLICK_DECLS
NetworkCoding::NetworkCoding() {}
NetworkCoding::~NetworkCoding() {}

int NetworkCoding::configure(Vector<String> &conf, ErrorHandler *errh)
{
    if (cp_varkparse(conf, this, errh,
        "NODEFUNCTION", cpkM, cpInteger, &node_function,
        "THISNODEADDR", cpkM, cpEthernetAddress, &node_addr,
        "CODINGDESTADDR", cpkM, cpEthernetAddress, &coding_dest_addr,
        "CODINGFWDDESTADDR", cpkM, cpEthernetAddress, &coding_fwd_dest_addr,
        "MULTICASTCODINGADDR1", cpkM, cpEthernetAddress, &multicast_coding_addr_1,
        "MULTICASTCODINGADDR2", cpkM, cpEthernetAddress, &multicast_coding_addr_2,
        cpEnd) < 0)
        return -1;
    if (node_function > 4)
        return errh->error("nodefunction must be 1, 2, 3 or 4");

    queue_size = 2;
    queue_occup[0] = 0;
    queue_occup[1] = 0;
    return 0;
}

int NetworkCoding::isCodingPossible(void)
{
    uint8_t orig_dest_addr[2][6];
    uint8_t orig_source_addr[2][6];
    uint8_t nc_possible = 0;

    for (int i=0; i<2; i++)
    {
        if (WritablePacket *q = queue[i]->uniqueify())
        {
            click_ether *ethh = reinterpret_cast<click_ether*>(q->data());
            memcpy(orig_dest_addr[i], ethh->ether_dhost, 6);
            memcpy(orig_source_addr[i], ethh->ether_shost, 6);
        }
    }
    for (int j=0; j<6; j++)
    {
```

```

    if ((orig_dest_addr[0][j] == multicast_coding_addr_1[j] || orig_dest_addr[0][j] == multicast_coding_addr_2[j]) &&
        (orig_dest_addr[1][j] == multicast_coding_addr_1[j] || orig_dest_addr[1][j] == multicast_coding_addr_2[j]))
    {
        if (orig_dest_addr[0][j] != orig_dest_addr[1][j])
        {
            nc_possible = 1;
        }
    }
    else
    {
        nc_possible = 0;
    }
}
return nc_possible;
}

int NetworkCoding::isDecodingPossible(void)
{
    uint8_t orig_dest_addr[2][6];
    uint8_t orig_source_addr[2][6];
    uint8_t dec_possible = 0;

    for (int i=0; i<2; i++)
    {
        if (WritablePacket *q = queue[i]->uniqueify())
        {
            click_ether *ethh = reinterpret_cast<click_ether *>(q->data());
            memcpy(orig_dest_addr[i], ethh->ether_dhost, 6);
            memcpy(orig_source_addr[i], ethh->ether_shost, 6);
        }
    }
    for (int j=0; j<6; j++)
    {
        if ((orig_dest_addr[0][j] == coding_dest_addr[j] || orig_dest_addr[0][j] == coding_fwd_dest_addr[j]) &&
            (orig_dest_addr[1][j] == node_addr[j]))
        {
            dec_possible = 1;
        }
        else
        {
            dec_possible = 0;
        }
    }
    return dec_possible;
}

int NetworkCoding::isForwardPossible(Packet *p)
{
    uint8_t orig_dest_addr[6];
    uint8_t orig_source_addr[6];
    uint8_t fwd_possible = 0;

    if (WritablePacket *q = p->uniqueify())
    {
        click_ether *ethh = reinterpret_cast<click_ether *>(q->data());
        memcpy(orig_dest_addr, ethh->ether_dhost, 6);
        memcpy(orig_source_addr, ethh->ether_shost, 6);
    }
    for (int j=0; j<6; j++)
    {
        if ((orig_dest_addr[j] == multicast_coding_addr_1[j]) ||
            (orig_dest_addr[j] == multicast_coding_addr_2[j]) || (orig_dest_addr[j] == coding_dest_addr[j]))
        {
            fwd_possible = 1;
        }
        else
        {
            fwd_possible = 0;
        }
    }
    return fwd_possible;
}

Packet * NetworkCoding::codeQueue(void)
{
    uint8_t inhou0[128] = {0};
    uint32_t lengte0 = queue[0]->length();
    uint8_t inhou1[128] = {0};
    uint32_t lengte1 = queue[1]->length();
    uint8_t kodeer[128] = {0};
    uint32_t lengte_kodeer = 0;
    Packet * out;

    for (uint32_t i=0; i<lengte0; i++)
    {
        inhou0[i] = *(queue[0]->data() + i);
    }

    for (uint32_t i=0; i<lengte1; i++)
    {
        inhou1[i] = *(queue[1]->data() + i);
    }

    if (lengte0 >= lengte1)

```



```

    {
        for (uint32_t i=0; i<length0; i++)
        {
            kodeer[i] = inhou0[i] inhou1[i];
        }
        lengte_kodeer = length0;
    }
    else if (length0 < length1)
    {
        for (uint32_t i=0; i<length1; i++)
        {
            kodeer[i] = inhou0[i] inhou1[i];
        }
        lengte_kodeer = length1;
    }
    out = Packet::make(kodeer, lengte_kodeer);
    out->set_timestamp_anno(queue[1]->timestamp_anno());
    return setEtherAddr(out, node_addr, coding_dest_addr);
}

Packet * NetworkCoding::changeForwardAddrXover(Packet *p)
{
    uint8_t orig_dest_addr[6];
    uint8_t addr = 0;

    if (WritablePacket *q = p->uniqueify())
    {
        click_ether *ethh = reinterpret_cast<click_ether *>(q->data());
        memcpy(orig_dest_addr, ethh->ether_dhost, 6);
    }
    for (int j=0; j<6; j++)
    {
        if (orig_dest_addr[j] == multicast_coding_addr_1[j])
        {
            addr = 1;
        }
        else if (orig_dest_addr[j] == multicast_coding_addr_2[j])
        {
            addr = 2;
        }
        else if (orig_dest_addr[j] == coding_dest_addr[j])
        {
            addr = 3;
        }
    }
    if (addr == 1)
    {
        return setEtherAddr(p, node_addr, multicast_coding_addr_2);
    }
    else if (addr == 2)
    {
        return setEtherAddr(p, node_addr, multicast_coding_addr_1);
    }
    else if (addr == 3)
    {
        return setEtherAddr(p, node_addr, coding_fwd_dest_addr);
    }
    else
    {
        return 0;
    }
}

Packet * NetworkCoding::changeForwardAddr(Packet *p)
{
    uint8_t orig_dest_addr[6];
    uint8_t addr = 0;

    if (WritablePacket *q = p->uniqueify())
    {
        click_ether *ethh = reinterpret_cast<click_ether *>(q->data());
        memcpy(orig_dest_addr, ethh->ether_dhost, 6);
    }
    for (int j=0; j<6; j++)
    {
        if (orig_dest_addr[j] == multicast_coding_addr_1[j])
        {
            addr = 1;
        }
        else if (orig_dest_addr[j] == multicast_coding_addr_2[j])
        {
            addr = 2;
        }
        else if (orig_dest_addr[j] == coding_dest_addr[j])
        {
            addr = 3;
        }
    }
    if (addr == 1)
    {
        return setEtherAddr(p, node_addr, multicast_coding_addr_1);
    }
    else if (addr == 2)
    {

```

```

        return setEtherAddr(p, node_addr, multicast_coding_addr_2);
    }
    else if (addr == 3)
    {
        return setEtherAddr(p, node_addr, coding_fwd_dest_addr);
    }
    else
    {
        return 0;
    }
}

Packet * NetworkCoding::decodeQueue(void)
{
    uint8_t inhou0[128] = {0};
    uint32_t lengte0 = queue[0]->length();
    uint8_t inhou1[128] = {0};
    uint32_t lengte1 = queue[1]->length();
    uint8_t dekodeer[128] = {0};
    uint32_t lengte_dekodeer = 0;
    Packet * out;

    for (uint32_t i=0; i<lengte0; i++)
    {
        inhou0[i] = *(queue[0]->data() + i);
    }
    for (uint32_t i=0; i<lengte1; i++)
    {
        inhou1[i] = *(queue[1]->data() + i);
    }
    if (lengte0 >= lengte1)
    {
        for (uint32_t i=0; i<lengte0; i++)
        {
            dekodeer[i] = inhou0[i] ^ inhou1[i];
        }
        lengte_dekodeer = lengte0;
    }
    else if (lengte0 < lengte1)
    {
        for (uint32_t i=0; i<lengte1; i++)
        {
            dekodeer[i] = inhou0[i] ^ inhou1[i];
        }
        lengte_dekodeer = lengte1;
    }
    out = Packet::make(dekodeer, lengte_dekodeer);
    out->set_timestamp_anno(queue[1]->timestamp_anno());
    return setEtherAddr(out, coding_dest_addr, node_addr);
}

Packet * NetworkCoding::setEtherAddr(Packet *p, uint8_t source[6], uint8_t dest[6])
{
    click_ether_ethh;
    uint16_t etht = 0x0800;

    ethh.ether_type = htons(etht);
    for (int i=0; i<6; i++)
    {
        ethh.ether_shost[i] = source[i];
        ethh.ether_dhost[i] = dest[i];
    }
    if (WritablePacket *q = p->uniqueify())
    {
        memcpy(q->data(), &ethh, 14);
        return q;
    }
    else
        return 0;
}

int NetworkCoding::initialize(ErrorHandler *errh)
{
    queue = (Packet **) CLICK_MALLOC(sizeof(Packet *) * (queue_size + 1));
    if (queue == 0)
        return errh->error("out of memory");
    return 0;
}

void NetworkCoding::push(int, Packet *p)
{
    if (node_function == 1 || node_function == 2)
    {
        if (queue_occup[0] == 0 && queue_occup[1] == 0)
        {
            //click_chatter("Queue empty - placing first packet in queue %d", p->length());
            queue[0] = p;
            queue_occup[0] = 1;
            queue[1] = queue[0];
            queue_occup[0] = 0;
            queue_occup[1] = 1;
        }
        else if (queue_occup[0] == 0 && queue_occup[1] == 1)
        {

```

```

queue[0] = p;
queue_occup[0] = 1;
queue_occup[1] = 1;
if (node_function == 2)
{
    if (isCodingPossible() == 1)
    {
        click_chatter("Packets Coded %u",1);
        output(0).push(codeQueue());
        queue_occup[0] = 0;
        queue_occup[1] = 0;
    }
    else if (isForwardPossible(queue[1]) == 1)
    {
        click_chatter("Packets NOT Coded but 1 packet forwarded %u",1);
        output(0).push(changeForwardAddrXover(queue[1]));
        queue[1] = queue[0];
        queue_occup[0] = 0;
        queue_occup[1] = 1;
    }
    else
    {
        click_chatter("Unknown packet received %u",1);
        output(0).push(queue[1]);
        queue[1] = queue[0];
        queue_occup[0] = 0;
        queue_occup[1] = 1;
    }
}
else if (node_function == 1)
{
    if (isDecodingPossible() == 1)
    {
        click_chatter("Packets Decoded %u",1);
        output(0).push(decodeQueue());
        output(0).push(queue[1]);
        queue_occup[0] = 0;
        queue_occup[1] = 0;
    }
    else
    {
        click_chatter("Packets cannot be decoded, forwarding oldest packet in buffer %u",1);
        output(0).push(queue[1]);
        queue[1] = queue[0];
        queue_occup[0] = 0;
        queue_occup[1] = 1;
    }
}
}
}
else if (node_function == 3)
{
    if (isForwardPossible(p) == 1)
    {
        click_chatter("Packets only forwarded %u",1);
        output(0).push(changeForwardAddr(p));
    }
    else
    {
        click_chatter("Unknown packet received %u",1);
        output(0).push(p);
    }
}
else
{
    output(0).push(p);
}
}

void NetworkCoding::cleanup(CleanupStage)
{
    for (int i = 0; i < 2; i++)
        queue[i]->kill();
    CLICK_LFREE(queue, sizeof(Packet *) * (queue_size + 1));
    queue = 0;
}

CLICK_ENDDECLS
EXPORT_ELEMENT(NetworkCoding)

```

## HostEtherFilter2Addr.cc

```

#include <click/config.h>
#include "hostetherfilter2addr.hh"
#include <click/confparse.hh>
#include <click/error.hh>
#include <click/glue.hh>

```

```

#include <click/etheraddress.hh>
#include <clicknet/ether.h>

CLICK_DECLS
HostEtherFilter2Addr::HostEtherFilter2Addr() {}
HostEtherFilter2Addr::~HostEtherFilter2Addr() {}

int HostEtherFilter2Addr::configure(Vector<String> &conf, ErrorHandler *errh)
{
    if (cp_va_kparse(conf, this, errh,
        "SOURCE_DEST", cpkM, cpInteger, &_func,
        "ETHER1", cpkM, cpEthernetAddress, &_addr1,
        "ETHER2", cpkM, cpEthernetAddress, &_addr2,
        cpEnd) < 0)
        return -1;
    return 0;
}

Packet * HostEtherFilter2Addr::drop(Packet *p)
{
    if (noutputs() == 2)
        output(1).push(p);
    else
        p->kill();
    return 0;
}

Packet * HostEtherFilter2Addr::simple_action(Packet *p)
{
    const click_ether *e = (const click_ether *) (p->data());
    if (_func == 1)
    {
        if ((memcmp(e->ether_shost, _addr1, 6) == 0) || (memcmp(e->ether_shost, _addr2, 6) == 0))
        {
            return p;
        }
        else
        {
            return drop(p);
        }
    }
    else if (_func == 2)
    {
        if ((memcmp(e->ether_dhost, _addr1, 6) == 0) || (memcmp(e->ether_dhost, _addr2, 6) == 0))
        {
            return p;
        }
        else
        {
            return drop(p);
        }
    }
    else
    {
        return drop(p);
    }
}

CLICK_ENDDECLS EXPORT_ELEMENT(HostEtherFilter2Addr)

```

# Appendix F

## Data CD

The data CD contains:

- All the referenced and non-referenced literature used,
- The conference contributions from the dissertation in pdf format,
- The OPNET custom model code and project files,
- A complete collection of the Click modular router code and Linux bash scripts used.