Using Topological Information in Opportunistic Network Coding

by

Magdalena Johanna (Leenta) Grobler

A thesis submitted to the Faculty of Engineering

in partial fulfilment of the requirement for the degree

MASTER OF ENGINEERING

in

COMPUTER ENGINEERING

at the

NORTH-WEST UNIVERSITY – POTCHEFSTROOM CAMPUS

Supervisor: Prof ASJ Helberg

November 2008

ABSTRACT

Recent advances in methods to increase network utilization have lead to the introduction of a relatively new method called Network Coding. Network Coding is a method that can reduce local congestion in a network by combining information sent over the network. It is commonly researched in the information theory field after it was first introduced by Ahlswede et al in 2000.

Network Coding was proven in 2003, by Koetter & Médard to be the only way to achieve the throughput capacity defined by the *Min cut Max flow theorem* of Shannon. It was applied deterministically in wired networks and randomly in wireless networks. Random Network Coding however requires a lot of overhead and may cause possible delays in the network.

We found that there is an open question as to determine where in a wireless network, Network Coding can be implemented. In this thesis we propose to find opportunities for the implementation of Network Coding, by searching for known deterministic Network Coding topologies in larger Networks. Because a known topology is used, we will then also know how Network Coding should be implemented. This method of finding opportunities for the implementation of Network Coding using topology can be combined with a routing algorithm to improve the utilization of a wireless network.

We implemented our method on three different topologies and searched 1000 random networks for the presence of these topologies. We found that these topologies occurred frequently enough to make our method a viable method of finding opportunities for the implementation of Network Coding.

Keywords:
Capacity, Information Theory, Network Coding, Network Topology, Wireless Mesh Networks

OPSOMMING

Onlangse vooruitgang in metodes wat die benutting van netwerke verhoog het gelei tot 'n relatief nuwe metode genaamd Netwerk Kodering. Netwerk Kodering is 'n metode wat die lokale kongestie in 'n netwerk kan verminder deur die inligting wat oor die netwerk gestuur word te kombineer. Dit word algemeen bestudeer deur navorsers in die veld van informasie teorie vandat dit die eerste keer in 2000 voorgestel is deur Ahlswede et al.

Netwerk Kodering is in 2003 deur Koetter & Médard bewys as die enigste manier waarop die deurset kapasiteit, soos voorspel deur die *Minimum sny Maksimum vloei* teorema van Shannon, bereik kan word. Dit word deterministies toegepas in vastelyn kommunikasie netwerke en willekeurig in draadlose kommunikasie netwerke. Willekeurige Netwerk Kodering het egter baie oorhoofse koste tot gevolg en kan ook vertragings in die netwerk veroorsaak.

Ons het gevind dat daar 'n oop vraag bestaan rakende die bepaling van die plek in 'n netwerk, waar Netwerk Kodering toegepas kan word. In hierdie tesis stel ons 'n metode voor om geleenthede vir die implementering van Netwerk Kodering te vind deur te soek vir bekende Netwerk Kodering topologië binne dié van groter netwerke. Siende dat 'n bekende topologie gebruik word, sal ons dan ook weet hoe om Netwerk Kodering op daardie plek te implementeer. Hierdie metode om geleenthede vir die implementering van Netwerk Kodering m.b.v. topologie te vind, kan ook gekombineer word met 'n roeteringsalgoritme om die benutting van 'n draadlose kommunikasie netwerk te verbeter.

Ons het ons metode op drie verskillende topologië toegepas en 1000 willekeurige netwerke deursoek vir die teenwoordigheid van hierdie topologië. Ons het gevind dat hierdie topologië dikwels genoeg voorkom om, ons metode om geleenthede vir die implementering van Netwerk Kodering te vind, lewensvatbaar te maak.

ACKNOWLEDGEMENTS

I dedicate this thesis to all those who made the pursuit of this research possible:

First I would like to thank God who guided me in wisdom and provided me with perseverance as well as serenity at times when things didn't turn out the way I planned.

Prof. Albert S.J. Helberg, who in these two years went well beyond his official role as supervisor and therefore will now be my lifelong mentor & friend: Thank you for the countless hours you spent on correcting and improving my work and forming me as a researcher.

My husband, André who supported me with love and encouragement: You mean more to me than you will ever know!!

My parents: Martin & Magda and in-laws: Andries & Maria: For the unconditional love and support you've bestowed upon me over the years

My sisters: Marné & Jana and sister in-law Miemie: Thank you for phoning and visiting during these two years of research... without you I would surely be crazy by now.

Prof. George van Schoor always making sure that I am all right and prof. Robert Holm for the hours you spent proofreading my work.

The TeleNet Research Group (Andreas. Melvin & Suné): for being my sounding board and helping me realize my dreams.

Telkom Centre of Excellence: for the financial support without which this research would not have been possible.

Humbly and sincerely yours,

Grobler

Leenta Grobler

21 November 2008

DECLARATION

I, ***Magdalena Johanna Grobler*** declare herewith that this thesis entitled "***Using Topological Information in Opportunistic Network Coding***", which I herewith submit to the North-West University as partial completion of the requirements set for the ***Master of Engineering*** degree, is my own work and has not already been submitted to any other university.

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

Signature of candidate: Grobler University number: ***12791911***

Signed at: ***Potchefstroom*** this ***21st*** day of November 2008.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AODV	-	Ad-hoc On-demand Distance Vector Routing
CSDA	-	Channel Select Delete Algorithm
DSP	-	Digital Signal Processing
DSR	-	Dynamic Source Routing
HRPLS	-	Hybrid Routing Protocol for Large Scale Mobile Ad Hoc Networks with Mobile Backbones
MANET	-	Mobile Ad-hoc Network
MFCA	-	Matrix-based Fast Calculation Algorithm
MPR	-	Multipoint distribution Relay
MSDA	-	Matrix Select Delete Algorithm
NLOS	-	No Line of Sight
OLSR	-	Optimized Link State Routing
PDA	-	Personal Digital Assistant
QoS	-	Quality of Service
SHARP		Sharp Hybrid Adaptive Routing Protocol
WMN	-	Wireless Mesh Network
WOSPF	-	Wireless Open Shortest Path First

CHAPTER 1 - INTRODUCTION

1.1 Motivation

The universal need for better control over resources in communication networks is a problem that is studied continuously. Maximum network capacity needs to be defined and then utilised to ensure that as much information as possible is delivered in the most beneficial manner. In practical packet switched networks, the network resources are managed by routing protocols. Traditional routing sends information packets through a network, along the routes determined by the routing protocol. The nodes on these routes only store and forward the packets they receive and any coding or decoding of the data in the packets is only done at the sender and receiver nodes.

A relatively new approach to routing and the utilisation of network resources is Network Coding, a field that, to date, led to a wide variety of theoretical results, especially in wired networks. The resource control problem in wireless networks also needs to be addressed and this thesis investigates the opportunities for the implementation of Network Coding in wireless networks. One documented implementation of Network Coding in wireless networks is Random Network Coding [1]. This implementation holds a number of disadvantages, as discussed in section 1.2.2.2.

In this thesis we investigate whether opportunities for the implementation of Network Coding can be found by using the topological information of the network.

1.2 Background

1.2.1 Wireless Networks

Wireless communication is associated with computer and telecommunication networks that communicate with each other using an alternative form of transmission to connect without the use of wires [2]. Looking at the vast amount of literature available on wireless communication, one can see that it is certainly a "hot topic".

Many types of wireless networks exist, but in this section, we will focus on Wireless Mesh Networks (WMNs) [3], and a special type of mesh network: Mobile Ad-hoc Networks (MANETs) [4].

1.2.1.1 Wireless Mesh Networks (WMNs)

A WMN is a wireless network of radio nodes that forms a mesh topology [3]. The mesh topology generally involves high connectivity – a property that can not easily be assigned to wired networks due to the high cost that will be involved [5]. These nodes can be either mobile or stationary and generally accommodates multiple hops. A WMN may also be connected to other networks (like the Internet or a wireless sensor network) making these networks difficult to manage. The nodes in WMNs can be either routers or clients and they are self configuring. Management of WMNs can be either centralised or decentralised [3].

The advantages and disadvantages of Wireless Mesh Networks can be summarized as follows [3, 5]:

Advantages:

- Multiple hops increase the area that the network can cover, and therefore no line of sight (NLOS) communication is also possible.
- The mesh topology increases the robustness of the network against link failures, because more than one path to a node exists.

Disadvantages:

- The multiple hops may cause delay in the delivery of information.
- Protocols associated with WMNs tend to be either complex or overly simple.

1.2.1.2 Mobile Ad-hoc Networks (MANETs)

A MANET [4] is a special type of wireless mesh network that typically consist of mobile devices such as mobile routers, laptop computers, personal digital assistants (PDAs) etc. These wireless devices are connected by wireless links to form a varying arbitrary network topology. Because these nodes are free to move randomly and organize themselves arbitrarily, the topology may change rapidly and unpredictably.

The management of ad-hoc networks is decentralized. This implies that each node present in the network acts as a forwarding node, forwarding messages to other nodes. The selection of forwarding nodes by the network itself, changes dynamically with the topology. A MANET may operate as a standalone network, or be connected to a larger network such as the Internet.

In other words a MANET is a network that is highly mobile, that receives frequent routing updates and that may contain nodes with high processing power.

The advantages and disadvantages of MANETs can be summarized as follows [4]:

Advantages:

- Adaptability to link failures because networks are often highly interconnected.
- Flexibility as a result of the wireless nature.
- Enables communication in environments with little or no infrastructure.

Disadvantages:

- Vulnerable to malicious attacks
- Congestion in the network
- Poor utilization of the network.

These disadvantages manifest as a result of a combination of factors: The use of an open medium, with a decentralized nature and a topology that changes dynamically, with poor

physical security. One of the biggest challenges in working with MANETs is to determine the network capacity.

1.2.2 Wireless Network Capacity

In order to manage the performance of a wireless network, it is important to be able to estimate the capacity of the network in some or other way. Jun *et al* [6] and Gupta *et al* [7] have shown that for WMNs, a throughput capacity of $0.0976\sqrt{n}$ (where n is the number of nodes) is achievable when $C = 6$ [8] (where C is the connectivity of the network), and analytical results have shown that the throughput capacity per node decreases as the node density increases [5].

Efforts to determine the capacity of MANETs include the Matrix Select-Delete Algorithm (MSDA) [9], Channel-sharing Select-Delete Algorithm (CSDA) [9] and Matrix-based Fast Calculation Algorithm (MFCA) [10] as described in section 2.3.1.2.

In our research we focus on developing a new technique to reduce congestion in wireless networks and thereby improving the capacity utilization of the network, by implementing Network Coding.

1.2.3 Wireless Routing Protocols

We foresee that Network Coding and a routing protocol can be combined, so let us take a look at some background on this. Routing protocols can be divided into three categories, proactive, reactive and hybrid routing protocols [11] which will now be briefly summarized.

1.2.3.1 Proactive routing protocols

When a proactive routing protocol is implemented, each node in the network stores information on the paths to every other node in the network [11, 12]. This information can then be used when a node wants to communicate with another node in the network. This category of routing algorithm has a disadvantage in terms of scalability, because the time it takes for all nodes to generate their information of the entire networks topology may be long and because the number of messages to maintain route information grow very fast. [12]. Examples of proactive routing protocols are: Optimized Link State Routing Protocol (OLSR) [13] and Wireless Open Shortest Path First (WOSPF) [14]. Both are described in section 2.2.1.1.1 and 2.2.1.1.2.

1.2.3.2 Reactive routing protocols

With the implementation of reactive routing protocols, the nodes store routes only to immediate neighbours, and then determine multi-hop routes as needed when communicating with another node [11, 15]. This is favourable because maintenance overhead is decreased. The disadvantage however is that this may take long because paths have to be calculated every time a node wants to send a message through the network [15]. Examples of reactive routing protocols are: Dynamic Source Routing (DSR) [16] and Ad-hoc On-demand Distance Vector routing (AODV) [17]. These two are described in section 2.2.1.1.3 and 2.2.1.1.4.

1.2.3.3 Hybrid routing protocols

Hybrid routing combines the strengths of the abovementioned routing protocols [11, 18]. It implements proactive routing within a given radius of nodes within a network and reactive routing outside that radius [18]. Examples of hybrid routing protocols are: Hybrid Routing Protocol for Large Scale Mobile Ad Hoc Networks with Mobile Backbones (HRPLS) [19] and Sharp Hybrid Adaptive Routing Protocol (SHARP) [20].

1.2.4 Network Coding

Network Coding is a field that was first introduced in 2000 [21] as a method to utilize the maximum capacity of a wired network and improve the flow of information in that network. It suggested coding at packet level in wired peer-to-peer networks. The idea sprouts from research done in [22] on satellite communications using a source coding system which consists of multiple sources, encoders, and decoders.

1.2.4.1 Network Coding Theory

We use the original Network Coding example network [21] (Butterfly network) to explain Network Coding and how it can be used to utilize the maximum throughput capacity of the network. We first refer to the Min Cut, Max Flow theorem [23]

Theorem 1:
"The maximum flow of information in a network is equal to the sum of the cut of the link capacities."

Using the theorem - we cut the network in figure 1-1, separating the sender node S and receiver nodes X and Y to cross as few of the links as possible to determine the minimum cut.

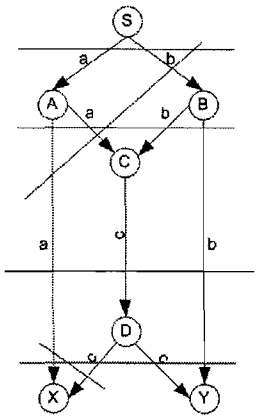


Figure 1-1 Butterfly Network explaining Min Cut, Max Flow Theorem

When each of the links have unit capacity we can see that the minimum cut, and therefore the maximum flow (or maximum throughput capacity) of this network is equal to 2. The only way that we can utilize this maximum throughput capacity, is by using Network Coding [24].

We will now discuss how two different types of Network Coding (Deterministic Network Coding and Random Network Coding) work.

1.2.4.1.1 Deterministic Network Coding

Continuing with the Butterfly Network [21], two nodes *A* and *B* need to transmit their messages to both Nodes *X* and *Y*. The links in figure 1-2 all have unit capacity and messages *a* and *b* are binary. Each of the nodes can deliver their own message to the node that it is directly connected to, but they have to route their messages through the network to reach the other node. When making use of traditional routing (Figure 1-2a), node *C* simply replicates the information it receives from the previous sender node. In this case the two messages *a* and *b* will reach node *C* simultaneously. Node *C* will send out message *a* first, and then message *b*. Thus, at the end of a single arbitrary time unit, only node *Y* will have received both messages, while node *X* still has only message *a*. This results in a throughput of 1.5 per unit of time [21].

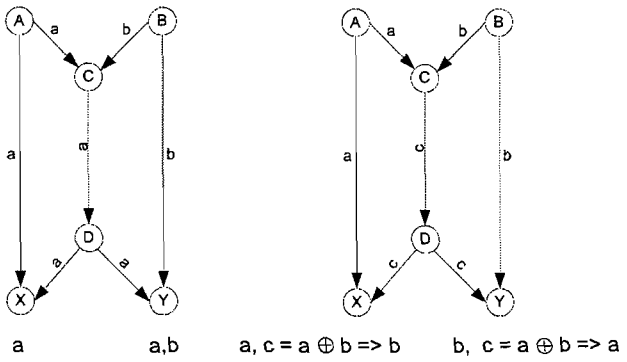


Figure 1-2 Butterfly Network a) Without Network Coding b) With Network Coding

When we make use of Network Coding (Figure 1.2b), we give node *C* the capability to transmit a linear combination (logical XOR) of the binary messages *a* and *b*. Message *c* has the same length as message *a* and *b*, and is transmitted via node *D* to nodes *X* and *Y*. We then give nodes *X* and *Y* the capability to decode message *c* by using the other message it already received and solving the two linearly independent equations. In this special case, it merely means adding the single message that the node has already received to the network coded message. This time, by the end of a single arbitrary time unit, both nodes *X* and *Y* have both messages. Two messages were delivered, making the throughput equal to 2 per unit of time [21].

This method however changes the way node *C* works, because it has to form linear combinations of the messages it receives before forwarding it. It also requires nodes *X* and *Y* to have knowledge of the network topology and how the messages reaching it are encoded in order to deduce the two original messages from the messages it received. Thus a complete view of the network must be available to the nodes, and some management information on where Network Coding took place.

The advantage of using this type of Network Coding, is that you know exactly how the information that reach nodes *X* and *Y* where combined. The disadvantage when we want to implement this opportunistically in a wireless network is that nodes in the network do not know how to change their functionality to implement Network Coding in this way.

One way in which Network Coding has been implemented in wireless networks, is Random Network Coding [1].

1.2.4.1.2 Random Network Coding

To see how Random Network Coding works [1], consider a portion of a MANET as depicted in figure 1.3. The five sender nodes transmit their messages to be delivered at receiver node *R*. The intermediate nodes are configured to form random linear combinations of the messages as they pass through the network. A coding vector that indicates how the messages have been combined is transmitted as part of the header of each message. Node *R* then receives a set of equations and waits until it has received as many linearly independent equations as the number of messages expected. These linearly independent equations for the example in Fig.1.3 can be shown as follows:

$$\begin{aligned} y_1 &= m_1 \\ y_2 &= m_1 \oplus m_2 \\ y_3 &= m_2 \oplus m_3 \oplus m_4 \\ y_4 &= m_4 \oplus m_5 \\ y_4 &= m_5 \end{aligned} \qquad (1)$$

Once node *R* has received enough linearly independent equations, it can solve *y* and all 5 messages will be delivered.

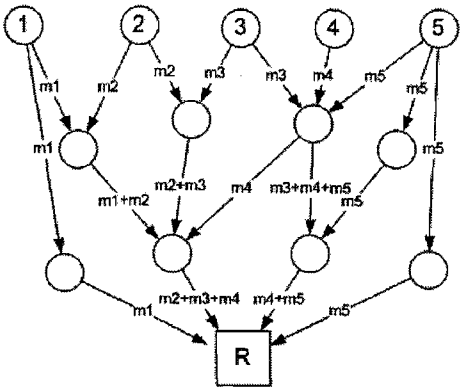


Figure 1-3 MANET with random Network Coding

The advantage [1] of this implementation of Network Coding is that centralized control of the network is not necessary and that the receiver does not need information on how the information it receives was combined. The disadvantage however is that because the receiver does not know how the information was combined, it does not know how long it should wait for sufficient information to reach it. To solve this, the information on how it was combined is included in a header, but it may still take very long for efficient information to reach the node.

We can see that in both these methods of Network Coding that the messages are delivered with fewer transmissions than with traditional routing. In the case of the deterministic approach to Network Coding, it has little overhead, which may be useful to improve network utilisation in a wireless network.

1.2.4.2 Benefits of Network Coding

The use of Network Coding in a network may provide the following benefits:

1.2.4.2.1 Throughput Capacity [25], [26], [27]:

Throughput capacity can be defined in terms of the amount of information sent through the network in a specific time. Improved throughput capacity in networks was the first major result of Network Coding. If we refer to the throughput capacity achieved with Network Coding in the deterministic example, we see that we have achieved the maximum throughput capacity as calculated using the min-cut max-flow theorem.

1.2.4.2.2 Robustness [24], [26], [27], [28]:

The robustness of the network refers to the ability of the network to remain functioning even though a link has failed completely. With Network Coding messages from a node are sent over more than one link and therefore the information from that node can still be delivered even though one of the links failed completely.

1.2.4.2.3 Adaptability [27], [29]:

Adaptability is an important benefit when looking at MANETs, as this refers to the ability of the network to cope with nodes constantly joining and leaving the network, resulting in a constantly changing topology. If Network Coding is applied opportunistically, the changing topology will cause different opportunities for the implementation of Network Coding.

1.2.4.2.4 Security [27]:

The security benefit is an intrinsic benefit, seeing that linear combinations of data are sent over the network, and not the actual data. This benefit, while useful, is however not sufficient to ensure security against malicious attacks on a network. If a malicious entity listens long enough and receives enough messages to decode the information, the information can still be eavesdropped. Plenty of research is however done on solving this [30, 31].

We can now see that the implementation of Network Coding can be beneficial in wireless networks.

1.3 Validation for study

We couldn't find any documented methods of determining where in a wireless network, Network Coding can be implemented. Nor was it found how Network Coding can be implemented at that specific location when an opportunity presents itself. It was stated in [32] that determining where Network Coding can be applied was still an open question.

1.4 Research Goal

The goal of this research is to provide a method of determining where in a wireless network, Network Coding can be implemented and comment on possible benefits of using this new method.

1.5 Issues addressed

The following issues were addressed:

- Identifying known deterministic Network Coding topologies to search for in wireless networks.
- Finding all the routes of a certain length within a network.
- Determining where and how often in the MANET known topologies appear.
- Determining whether the presence of the known topology provides an opportunity to implement Network Coding deterministically.
- Measurement of the improvement that the implementations of opportunistic deterministic Network Coding offer.
- Comments on the practicality of implementing Network Coding in this manner.

1.6 Research Methodology and Overview of Thesis

Our approach to address the abovementioned issues follows what is commonly referred to as the "scientific method" or "method of research" as can be found in [33] and [34]. We interpreted this as presented in figure 1-4:

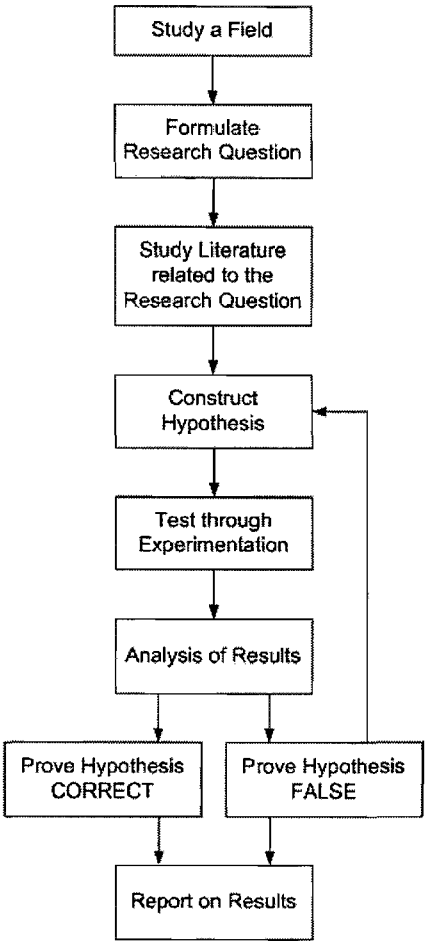


Figure 1-4 The Scientific Method

Before we discuss how this method was followed in this research, we clarify common differences when performing research in technology and engineering, with that of sciences.

Firstly, the research question when performing technological research, tends to focus on a improvement of methods rather than the explanation of a phenomenon, as is often the case with scientific research.

Secondly the term *Hypothesis* as commonly used in scientific research is defined by [35] as:

i. *"A proposition, or set of propositions, set forth as an explanation for the occurrence of some specified group of phenomena, either asserted merely as a provisional conjecture to guide investigation (working hypothesis) or accepted as highly probable in the light of established facts."*
ii. *"A proposition assumed as a premise in an argument."*
iii. *"The antecedent of a conditional proposition."*

When we however use this term in technological research, we use this term as it is defined by [36]:

"An idea has not yet been proved to be true or correct."

Thirdly the experiments performed in technological research refer to implementations of the hypothesis to improve a method/methodology, in order to prove that the idea (often a new method) is viable, in contrast to science where the experiments aim to better describe a phenomenon. This new method or methodology is then tested for various scenarios in order to comment on the practicality of the implementation of the new method. The results from these tests are then used to decide whether the hypothesis was proven true or false.

We followed this methodology for our research as follows:
We studied literature from the fields of communication networks, routing and Network Coding, and realised that there were no documented methods to determine where in a Network, Network Coding can be applied. Our research question after this realisation was:

Can Topological information be used to determine opportunities for the implementation of Network Coding?

We then studied various methods (described in chapter 2) which would enable us to use the topological information of a network. Our hypothesis after the study of the literature and methods was:

Opportunities for the implementation of Network Coding can be found by looking for "known Network Coding topologies" within larger networks.

We implemented this idea in the method as described in chapter 3. We then adapted the method for the scenarios documented in chapters 4, 5 and 6. We tested the method for the three scenarios in 1000 randomly generated networks and analysed it to comment on the viability of the idea as can be seen in chapter 7. Our hypothesis was proven as described in chapter 8 where the thesis closes with a summary of the research and its implications.

In the appendix we included the conference articles that flowed from this research. We also included the source code used for the implementation of our method.

In this chapter, we motivated the research and gave some background on the field. Issues that were addressed listed together with how they were dealt with using the scientific method. Finally we provided the reader with an overview of the thesis that follows.

CHAPTER 2 - LITERATURE STUDY

2.1 Introduction

In this chapter we discuss the methods that motivated this research and made the implementation of our new idea possible. The chapter starts off by looking at network topology and ways to construct a representation of the topology. We discuss an algorithm called Exact Multiplication to find the existing routes between two communicating nodes within the network. The chapter goes further by discussing the advantages of creating more possible routes in an existing and proposes network, and suggests Network Coding as a viable technique for achieving this. We finally mathematically describe Network Coding. The chapter closes with comments on how the information presented in this chapter can be applied and used to realise our new idea.

2.2 Network Topology

To fully understand a given network, we must first know its topology. We want to know how many nodes the network includes, what the connectivity (the number of nodes connected to the node in question) of the network is and to which other nodes each of the nodes are connected. One way to accumulate this information is to visually inspect the network, but

because we will be using a wireless network, this may not be a very productive method. Another way we can find this information is by using the information in the routing tables of each of the communicating nodes.

2.2.1 Routing tables

A routing table can be described as a database storing the routes between communicating nodes within a network. From this database the topology of a network may be built. Contents of routing tables may differ but will include at least three different fields [37]:

- The network ID of the receiver node
- The implementation cost of the path between the sender and receiver nodes
- The next hop on the way to the receiver node.

A routing table is constructed by the routing protocol. Not all routing protocols will make it possible to for all nodes in that network to know the topology of the complete network [37],

2.2.1.1 Routing protocols

The two routing most common routing protocols associated with multi-hop wireless networks are the Dynamic Source Routing (DSR) and Ad-hoc On demand Distance Vector (AODV) protocols [16, 17], Both these protocols are classified as reactive.

Because we are interested in wireless ad-hoc networks, we will also discuss two proactive link state protocols, the Wireless Open Shortest path first (WOSPF) and Optimised Link State Routing (OLSR) protocol.

2.2.1.1.1 DSR

DSR [16] enables the network to be largely self-configuring. Each time a sender wants to send a message to a receiver node, route request packets are broadcast through the whole network. The receiver node then answers with an acknowledge packet, containing the route from the sender node to the receiver node. The message is then sent to the receiver node with information of complete route in its header. This may reduce throughput as a result of too much overhead.

2.2.1.1.2 AODV

When the AODV protocol [17] is used, a route discovery request is broadcast locally when a message is to be sent from a sender node to a receiver node. Each node in the communication path then only record the information of the destination of the message that must be sent and the next hop on the route to reach that destination. This is repeated until the destination is found. AODV can be called a lightweight protocol as it requires little overhead.

2.2.1.1.3 WOSPF

Another protocol that is currently under investigation is a link-state protocol called Wireless Open Shortest Path First (WOSPF) [14] protocol otherwise known as OSPFv2 [38]. This is a proactive protocol and similar to OSPF, the WOSPF protocol lets every node construct a map of how the network is connected and it allows for multi-hop communication. One node is nominated to flood the network with packets to enable other nodes to construct their own maps of the network. Therefore each node has knowledge of the topology of the complete network [39]. This information can be found in the routing table and can be investigated to determine the topology of the network.

2.2.1.1.4 OLSR

If we use ad-hoc type wireless networks, one elected node that has to flood the network with packets (as in WOSPF) may not be very productive and OLSR [13] may be a better option. With the OLSR protocol, each node discovers its two-hop neighbours and a number of "managing" nodes are then elected as multipoint distribution relays (MPRs). The MPR nodes then gather information on the topology and forward the topology information to the other managing nodes. OLSR does not promise the same reliability as WOSPF due to the lack of a reliable algorithm for ad-hoc wireless networks, but it forwards topology data often enough to ensure database synchronisation across all managing nodes. The managing nodes thus contain a complete topology of the network and may act as the decision maker in assigning Network Coding tasks to certain nodes.

2.2.2 Topology Representation

Whichever protocol we use to determine the network's topology, we would want to represent this data in a way that we can use the topology information to perform calculations on it and draw conclusions from the calculations. There are two main ways in which the topology information can be represented, i.e. graphical and using matrices.

2.2.2.1 Graphical

The first is by displaying it graphically. We can draw a diagram of connected nodes and then by inspecting the diagram make some observations. By counting the number of links connected to the nodes we can determine the connectivity and by careful inspection we may even be able to see whether patterns exist within the network. This effort may be tedious and there is information that can be missed if one is not attentive enough. Calculations on a graph can also not be done directly and another method should therefore rather be used.

2.2.2.2 Adjacency matrices

The creation of an adjacency matrix is a constructive way of representing the data in a routing table. Calculations can easily be done and pattern-searches will be much simpler. When we use the same notation as used in [24], an adjacency matrix of a network is constructed as follows:

We describe the network as a directed graph $G(V, A, \mathrm{R})$ where V represents the set of nodes in the network, A is the adjacency matrix describing the topology of the nodes in V and $\mathrm{R}[r(e)]_{e \in A}$ describes the link capacities. The adjacency matrix $A = (a_{ij})$ is an $(n \times n)$ matrix (n is the number of nodes in the network), with $a_{ij} \in \{1, 0\}$. Both the rows and columns represent the nodes in the network and the n'th row and the n'th column represent the same node $(a_{1n} = a_{n1} = a_n)$. A connection of one node to another is indicated by an 1 in the matrix and to indicate that a node is not connected (or within transmission range) to another node a 0 is placed in the matrix. In a network and therefore in matrix A each node is at least connected to itself – therefore the main diagonal of the matrix will contain only $1's$. If we combine these characteristics we can say:

$$A_{n \times n} = (a_{ij}) = \begin{pmatrix} 1 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 1 \end{pmatrix} \text{where } a_{ij} \in \{1, 0\} \qquad (2)$$

If we were to subtract the identity matrix I_n from $A_{n \times n}$ we would be left with $B_{n \times n}$ the true representation of how the nodes are connected to each other:

$$A_{n\times n} - I_n = \begin{pmatrix} 1 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 1 \end{pmatrix} - \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 0 \end{pmatrix} \quad (3)$$
$$= B_{n\times n} \in \{1,0\}$$

Calculations can then be performed on $B_{n\times n}$, like determining the connectivity of the network (the minimum of the sum of the rows containing $1's$ is equal to the connectivity of the network).

2.2.2.2.1 Example

We can illustrate how to this works by using the network in figure 2-1.

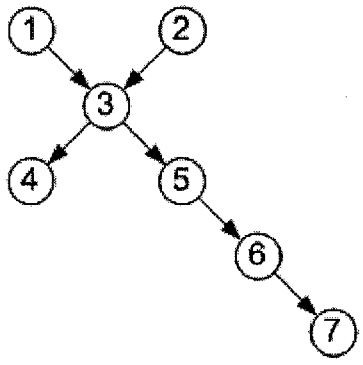


Figure 2-1 Example Network

We describe the network as a directed graph, as in section 2.2.2.2 so that:

$$A_{7\times7} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \quad (4)$$

from which we can calculate using (3):

$$B_{7\times7} = A_{7\times7} - I_7$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (5)$$

From (5) we can then see that the connectivity *(C)* of this network is 1.

2.3 Routes within Networks

Once we have an accurate representation of the topology of the network, we can start to use the information in it. One thing that we can derive from the adjacency matrix is the routes within the network. Once we know what routes in a network exist, we can understand how routing will take place and how many hops between sender and receiver will be used for a message to be delivered. One way in which the routes in a network can be determined using the adjacency matrix, is by exact multiplication.

2.3.1 Exact Multiplication

Exact multiplication is a method that can be used to calculate the *n-hop* paths (the paths of communication connecting nodes that are *n-hops* away from each other) between any two communicating nodes in a network. It was introduced by [9, 10] who incorporated the result of Exact multiplication into algorithms calculating the capacity of ad-hoc networks, which is what first drew our attention to these papers.

2.3.1.1 Exact Multiplication Method

In [9] and [10] the authors use the adjacency matrix as described in equation 2 as the *n-hop* adjacency matrix without regarding the directedness of the network. By using exact multiplication the *i+1-hop* adjacency matrices can then be calculated by defining:

$$A_{n\times n}^{(i+1)} = A_{n\times n}^{(i)} \times A_{n\times n} \tag{6}$$

From [9, 10] we then recall:

$$A_{n\times n}^{(1)} = A_{n\times n} \tag{7}$$

and the new matrix

$$A_{n\times n}^{(i+1)}(u,v) = \begin{cases} i+1 & \text{if } A_{n\times n}^{(i)}(u,s)=0, A_{n\times n}^{(i)}(u,v)=i, A_{n\times n}^{(i)}(s,v)=i \\ A_{n\times n}^{(i)}(u,v) & \text{if } A_{n\times n}^{(i)}(u,v)>0 \end{cases} \tag{8}$$

where u is the row index and v the column index. Using this notation, $A_{n\times n}^{1}(u,v) = a_{ij}$ as used in equation 2. This will give the shortest hop count between any two nodes. All *k-hop* adjacency matrices can be calculated from the *1-hop* adjacency matrix in equation 4 and therefore all potential *k-hop* paths [9, 10]. This can then be used for further calculations for instance capacity calculation as suggested by [9, 10].

2.3.1.1.1 Example

Continuing with the example in 2.2.2.2.1, we can now say from (7) that:

$$A_{7\times7}^{(1)} = A_{7\times7} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (9)$$

Further, using exact multiplication we can calculate $A_{7\times7}^{(2)}$ using (8):

$$A_{7\times7}^{(2)} = \begin{bmatrix} 1 & 2 & 1 & 2 & 2 & 0 & 0 \\ 2 & 1 & 1 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 2 & 0 \\ 2 & 2 & 1 & 1 & 2 & 0 & 0 \\ 2 & 2 & 1 & 2 & 1 & 1 & 2 \\ 0 & 0 & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 & 1 \end{bmatrix} \quad (10)$$

and also $A_{7\times7}^{(3)}$:

$$A_{7\times7}^{(3)} = \begin{bmatrix} 1 & 2 & 1 & 2 & 2 & 3 & 0 \\ 2 & 1 & 1 & 2 & 2 & 3 & 0 \\ 1 & 1 & 1 & 1 & 1 & 2 & 3 \\ 2 & 2 & 1 & 1 & 2 & 3 & 0 \\ 2 & 2 & 1 & 2 & 1 & 1 & 2 \\ 3 & 3 & 2 & 3 & 1 & 1 & 1 \\ 0 & 0 & 3 & 0 & 2 & 1 & 1 \end{bmatrix} \quad (11)$$

and finally $A_{7\times7}^{(4)}$:

$$A_{7\times7}^{(4)} = \begin{bmatrix} 1 & 2 & 1 & 2 & 2 & 3 & 4 \\ 2 & 1 & 1 & 2 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 & 1 & 2 & 3 \\ 2 & 2 & 1 & 1 & 2 & 3 & 4 \\ 2 & 2 & 1 & 2 & 1 & 1 & 2 \\ 3 & 3 & 2 & 3 & 1 & 1 & 1 \\ 4 & 4 & 3 & 4 & 2 & 1 & 1 \end{bmatrix} \quad (12)$$

Since no *0's* are left after (12) we know that any node in the network can reach any other node in the network within 4 hops. Exact multiplication can therefore also be used to determine the longest path in a network, by means of:

$$\max\left[A_{n\times n}^{(i+1)}(u,v)\right], \text{ if } A_{n\times n}^{(i+1)}(u,v) \neq 0 \ \forall \ \{u,v\} \quad (13)$$

2.3.1.2 Using Adjacency Matrices for Capacity Calculation

The capacity of a wireless ad-hoc network can be defined as:

Definition 2[9]:
"Capacity is the number of k-hop sessions that share the same channel [9, 10], where channel is a means of communication or access."

There are three algorithms that use adjacency matrices to calculate the capacity of wireless ad-hoc networks. These three algorithms will now be described.

2.3.1.2.1 Matrix Select-Delete Algorithm (MSDA)

MSDA [9] uses the original generated connection matrix (equation 3) as the *1-hop* adjacency matrix. It then defines a delay of k and uses exact multiplication (equations 5 and 6) to generate $A_{n\times n}^{k}$. It is then required to calculate all the *k-hop* paths, the literature [9] and [10] did not describe how this should be done; therefore we suggest doing it as follows:

1. Start by looking in the upper-triangle for a value equal to k. We store the column numbers of the start and end nodes (as the first and last values) in a vector *Path* of size $k+1$ as the source and sink nodes.

2. We then move to the row number corresponding to the column where the sink was found and search for the value *k-1*. Ignoring the columns of the source and sink as well as the values on the diagonal of the $A^{k}_{n\times n}$ matrix. This value is stored as the second value in the vector *Path*.
3. We repeat the search in the corresponding row of this node's column for the value *k-2*, to fill the second to last element in *Path*.
4. We repeat this process until vector *Path* is filled and then perform a check that the last found node is indeed connected to the first node in Path. If it is not connected, the path is deleted.
5. All the generated Path vectors are then stored in *PathSet*.

Continuing with MSDA, for every node in *PathSet*, the probability that the node is present in *PathSet* is then calculated, and the sum of the probabilities on a path are then used to calculate the probability of occurrence of each path.

The path with the lowest probability is then selected and all the paths containing any of the nodes that are present in the selected path are then deleted. By doing this once for every network topology, we find all the independent *k-hop* paths in $A_{n\times n}$. From this, we can calculate the capacity of the wireless ad-hoc network by using [9]:

$$N_s = \min\left[\left\lfloor \frac{BW_{node}}{BW_{packet}} \right\rfloor, \frac{D_E(n^2 - n - n_0)}{\sum_{i\in A_{(n\times n)}} i - n}\right] \qquad (14)$$

Where BW_{node} is the bandwidth available to a node, BW_{packet} is the bandwidth needed per packet and $\lfloor BW_{node} / BW_{packet} \rfloor$ is the number of *1-hop* sessions that the channel can support. D_E is the end-to end delay and n is the number of nodes in the matrix $A_{n\times n}$, while n_0 is the number of zeros in the matrix.

2.3.1.2.1.1 Example

Referring to the example in 2.3.1.1 we go further to construct the *3-hop* paths by using $A^{(3)}_{7\times 7}$ calculated in (11) and the five steps mentioned in section 5.2.1.1. We then find:

$$PathSet = [(1 \quad 3 \quad 5 \quad 6), (2 \quad 3 \quad 5 \quad 6), (3 \quad 5 \quad 6 \quad 7)] \tag{15}$$

Because we regard the directedness of the network, the route (4 3 5 6) are not included in *PathSet*. Through inspection of (15) we can see that nodes 1, 2 and 7 occur once, while nodes 3, 5 and 6 each occur three times. We use this to calculate the probabilities of each path:

$$\begin{aligned}
(1 \quad 3 \quad 5 \quad 6) &= \frac{1+3+3+3}{12} = 83\% \\
(2 \quad 3 \quad 5 \quad 6) &= \frac{1+3+3+3}{12} = 83\% \\
(3 \quad 5 \quad 6 \quad 7) &= \frac{3+3+3+1}{12} = 83\%
\end{aligned} \tag{16}$$

From (16) we can then see that all the paths have equal probability. We therefore use the first of the three paths as our selected path and all the paths containing any of the nodes that are present in the selected path are then deleted leaving us with only one path:

$$PathSelect = (1 \quad 3 \quad 5 \quad 6) \tag{17}$$

We can calculate the capacity of the wireless ad-hoc network using (14) by substituting the following values: $\lfloor BW_{node} / BW_{packet} \rfloor = \lfloor 1/1 \rfloor$, $D_E = 3$, $n = 7$, $n_0 = 30$ such that:

$$\begin{aligned}
N_s &= \min\left[\left\lfloor \frac{1}{1} \right\rfloor, \frac{3(7^2 - 7 - 30)}{13 - 7}\right] \\
&= \min\left[1, \frac{36}{6}\right] \\
&= 1
\end{aligned} \tag{18}$$

The MSDA assumes [9] that each channel is used by only one session, meaning the number of sessions equal the number of paths. This is because a session belongs to only one path, making every path equal to a same hop session.

2.3.1.2.2 Channel-sharing Select-Delete Algorithm (CSDA)

CSDA [9] follows the same general pattern as MSDA, the difference being that it allows for channel sharing. With CSDA, the capacity can be calculated by using [9]:

$$N_s = \left\lfloor \text{count} \times \min \left[\left\lfloor \frac{BW_{node}}{BW_{packet}} \right\rfloor, \frac{D_E(n^2 - n - n_0)}{\sum_{i \in A_{(n \times n)}} i - n} \right] \right\rfloor \quad (19)$$

Where $\lfloor BW_{node} / BW_{packet} \rfloor$ is the number of 1-*hop* sessions that the channel can support, D_E is the end-to end delay and n is the number of nodes in the matrix $A_{n \times n}$.

2.4 Adding more links to networks

Now that we have a way of finding the routes in a network, we need a way that we can increase the number of *1-hop* links without installing more network hardware. By increasing the number of links in a network we can improve the quality of service of the network in terms of less congestion in the network and therefore better network utilization, and lower delay. We propose to achieve this, by implementing Network Coding in our network.

2.4.1 Network Coding

When implementing Network Coding, we create the equivalent of extra virtual links in the network when the information sent through the network is combined.

2.4.1.1 A Mathematical notation for Network Coding

We introduced the concept of Network Coding in section 1.2.2. We now present the mathematical notation for Network Coding as presented by [24]

We use graph G from section 2.2.2.2, a source matrix S , and a demand matrix D . We assume that all the links have unit capacity. S is a $K \times |V|$ binary matrix that describes which of K possible source nodes (if indeed any) enter the network at nodes $v \in V$. Likewise D is a $K \times |V|$ binary matrix that describes which of K possible sources (if indeed any) are required at nodes $v \in V$. If block codes are to be used with Network Coding, it can more accurately be

put that $\mathrm{B}=(B_1,...,B_K)$ is a collection of K input processes. We then let $\mathrm{B}_1,\mathrm{B}_2,...$ describe samples of source vector B with $\mathrm{B}_t=(B_{1,t},B_{2,t},...,B_{K,t})$ being the vector source samples at integer time t while $B_k^n=(B_{k,1},...,B_{k,n})$ represents the first n samples of the k'th random process $B_{k,1},B_{k,2},...$ for any $k\in \mathrm{K}=\{1,...,K\}$. The source matrix $\mathrm{S}=[s(k,v)]_{(k,v)\in \mathrm{K}\times V}$ has entries:

$$s(k,v)=\begin{cases}1 \text{ if source } B_k \text{ is available at node } v\\ 0 \text{ otherwise}\end{cases} \quad \textbf{(20)}$$

The demand matrix $\mathrm{D}=[d(k,v)]_{(k,v)\in \mathrm{K}\times V}$ has entries:

$$d(k,v)=\begin{cases}1 \text{ if node } v \text{ requires source } B_k\\ 0 \text{ otherwise}\end{cases} \quad \textbf{(21)}$$

It can then be assumed that when $s(k,v)=1$ then $d(k,v)=0$ (and vice versa), and $\sum_{v\in V}d(k,v)>0$ for all $k\in \mathrm{K}$. This implies that any node can be a transmitter and a receiver, given that it is not for the same source.

But we can define a network code (f^n,g^n) for graph G with n nodes to enable it to be a set of encoders and decoders. Then every encoder f_a^n in $f^n=\{f_a^n\}_{a\in A}$ is a deterministic mapping of the output on edge $a\in A$ with A as in equation 2, a function of the accessible inputs. Also every decoder $\mathrm{g}^n=\{g_{k,v}^n\}_{(k,v)\in \mathrm{K}\times V}$ defines K node decoders $(g_{1,v}^n,...,g_{k,v}^n)$ for each node $v\in V$ and then every decoder $g_{k,v}^n$ is a deterministic mapping that reconstructs the source B_k^n at node v. This can then be further described and defined as in [24].

2.4.1.1.1 Example

We can illustrate how this will work, by using the Butterfly topology illustrated in figure 2-2.

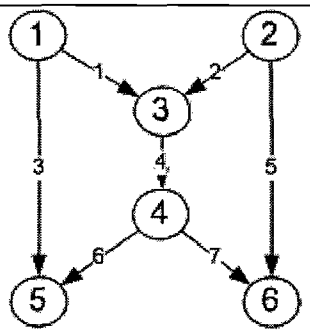


Figure 2-2 Butterfly Example

From this figure, we can see that there are 7 links *(a)* connecting the 6 nodes *(v)* of this network. We can then say that S is a $2\times|7|$ binary matrix, when nodes *1* and *2* act as the two source nodes:

$$S=\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 0\\ 0 & 1 & 1 & 0 & 0 & 1\end{bmatrix} \quad (22)$$

And D is a $2\times|7|$ binary matrix, when nodes *5* and *6* act as the two receiver nodes:

$$D=\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 1 & 0\end{bmatrix} \quad (23)$$

And an encoder f_3^7 should be created as a deterministic mapping of the output on link $4\in A$, to be a function of the inputs from links 1 and $2\in A$. Also two decoders $g^7=\{g_5^7, g_6^7\}$ should be created to reconstruct the information from the sources at nodes 5 and $6\in V$.

In chapter 1 we have described the benefits that the implementation of Network Coding presents and in this chapter we have defined the theory behind it. We also introduced in chapter 1 the concept of Random Network Coding as an alternative implementation of Network Coding in wireless networks but mentioned that it requires more overhead than the deterministic approach. Therefore we can deduce that it would be fruitful if we can find a way

to implement the techniques documented in this chapter to identify opportunities within wireless networks for the implementation of deterministic Network Coding.

2.5 Conclusion

Looking at the literature presented in this chapter, we have collected methods to take information from a routing table and present it in and adjacency matrix. We can then calculate from this adjacency matrix, the connectivity of the network and calculate by means of MSDA, the capacity of the network. Finally we saw how deterministic Network Coding works in theory. Our goal is to seek for opportunities within the network to implement Network Coding.

In this chapter we discussed the methods that made the implementation of our new idea possible.

CHAPTER 3 – FINDING OPPORTUNITIES FOR NETWORK CODING USING NETWORK TOPOLOGY

3.1 Introduction

We found that there is currently no method that indicates where in a wireless network or how exactly at that specific location deterministic Network Coding could be implemented. This led to the question: Can known topologies for deterministic Network Coding be used to apply Network Coding at favourable locations in a wireless network?

In this chapter we propose to examine the adjacency matrix of a wireless network, to see if we can find sub-matrices that are known Network Coding topologies. If we can find these sub-matrices, we know where in the network the opportunity to implement Network Coding exists. Because we use **known** topologies we then also know how to implement Network Coding, i.e. which nodes should change in their functionality, and which should have information on the network topology. Our aim in this is to improve the local throughput of part of the network, by using deterministic Network Coding topologies in a selection of the information flow.

We foresee the following benefits when using this concept:

- An improvement in the utilization of the network's capacity if the network can deliver more messages with the same number of links per unit time.
- Lower occupation of the total network because the combined messages will be sent over dedicated links.
- Improved quality of service (QoS) because of a lower delay and therefore less congestion in the network. QoS can be defined [40] as the ability to prioritise applications present in the network in order to assure a defined level of performance, when congestion is present in the network.

We propose the following method to illustrate this concept:

i. Select a Network Coding topology of which the gain and capacity is known.
ii. Derive the connection matrix of the larger network from a suitable distance vector routing algorithm,
iii. Search the larger network matrix for the known topology structure.
iv. Implement Network Coding at the appropriate nodes.
v. Re-iterate steps (iii) and (iv) after a routing update.

3.2 Alternatives

In our endeavour to apply this method, we implemented three search methods. These methods are described in the following sections.

3.2.1 Iterative looping

The first method was to translate the known Network Coding topology to an adjacency matrix as can be seen in equation 24, and search for that specific smaller matrix within the larger network adjacency matrix.

$$\text{Butterfly} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (24)$$

The simplest method to do this was iterative looping [41]. The fundamental concept of iterative looping is a nested looping structure. This effectively moves a window across the network adjacency matrix and continually compares the data in the current window to that of the desired data, in this case the Butterfly matrix.

This method was implemented in MATLAB to search networks of between 10 and 100 nodes for only this specific Butterfly adjacency matrix. We therefore did not locate any Butterflies that were numbered of orientated differently.

Although this method was able to accurately locate the Butterfly adjacency matrix within the larger adjacency matrix it scales poorly, and is computationally intensive. For this reason we propose using a different method.

3.2.2 Cross-correlation

The second method is based on the two dimensional cross correlation of the network's *1-hop* adjacency matrix and the Butterfly adjacency matrix. The cross correlation of two datasets can be seen as the similarity of those two datasets, and is commonly [42] used to search for patterns in a random dataset by forming the cross-correlation to known patterns. This technique finds application in digital signal processing (DSP) [43] as well as image processing [44].

This method was implemented in MATLAB to search networks of between 10 and 100 nodes for only this all permutations of the Butterfly adjacency matrix and the concept of cross-correlation was extended to two dimensions with the network adjacency matrix being used as the random dataset and the Butterfly adjacency matrix being used as the known pattern. The result of the cross-correlation operation *xcor(adjacency, butterfly)* , our own function based on the *xcorr* function found in the signal processing toolbox of MATAB, results in a new matrix M with values indicating the level of cross-correlation. In order to extract the correct values, the auto-correlation of the butterfly *xcor* matrix was formed. By scanning M for this value the location of the Butterfly matrix can be determined.

Advantages of using the cross-correlation method is scalability, as the method remains computationally efficient when using larger matrices. A further advantage is the ability to detect where partial Butterfly structures occur within the network connection matrix. These

partial structures then have the potential to become butterflies if the topology of the network changes slightly.

This method was compared to the previous method of Iterative looping for both accuracy and the computational efficiency. For small matrices of less than 40x40 both methods performed equally well in terms of computation time. For larger matrices of 100x100 the cross-correlation method's computation time is significantly less than that of the iterative looping. As the matrix dimensions increase the computational advantages of the cross-correlation method increases.

The cross-correlation implementation at first yielded a known topology in one of every hundred explored networks. We then realized that we had to test for all permutations of the known topology, as the orientation of the topology and the numbering of the nodes may differ for each network. We then searched for all the possible permutations of the known topology. These results were presented in [45], see Appendix A.

We however decided to implement another method because the first two methods took very long to complete a search of a network, and when we pre-program all the permutations of the Butterfly adjacency matrix as known patterns to search for the cross correlation method is quite inefficient.

3.2.3 Implementing Exact Multiplication for Searching

We explored a third method, where the aim was to extend the work done in [9] with exact multiplication, as was presented in section 2.3.1.2. In other words, use the exact multiplication method to get all the paths of a certain length in a network. This information would then be used to search for shared nodes and paths within the network in order to create subsets of nodes that can be compared to each other to identify the known topology.

In the implementation of this idea, it was realized that exact multiplication, as implemented in MSDA only yields the shortest paths between two nodes and not all paths are needed for comparison of the subsets.

We created our own scheme for constructing subsets of all the paths of a certain length through concatenation, making it in effect a brute-force search. This scheme was then

incorporated with the proposed method to find opportunities for Network Coding using network topology and implemented to produce the results as described in chapter 7. This implementation is discussed in the next section.

3.3 Path search using Concatenation

With our scheme to find all the paths of a certain length within a network through concatenation, we start by using the true representation of the network's topology – the $B_{n\times n}$ matrix of equation 2 in section 2.2.2.2 so that:

$$B_{n\times n} = (b_{ij}) \in \{1, 0\} \tag{25}$$

We then search in every row of the $B_{n\times n}$ matrix, for all the *1's* present. Each time a *1* is found, a *1-hop* route exists, and is stored in a $r\times 2$ matrix $O_{r\times 2}$, where r is the number of *1-hop* routes that exist, so that:

$$O_{r\times 2} = [o_{r1} \quad o_{r2}] \tag{26}$$

where

$$\begin{matrix} o_{r1} = i \,\forall\, b_{ij} = 1 \\ o_{r2} = j \,\forall\, b_{ij} = 1 \end{matrix} \text{ when } 1 \le r \le (n^2 - n) \text{ and } i < j \tag{27}$$

Once we have this matrix of *1-hop* paths, we can start concatenating them to form longer paths which we can then use to search for a known Network Coding topology.

In some of the literature the Butterfly is called the canonical example [46] of Network Coding, emphasising that the Butterfly typology is simple yet significant, without loss of generality. We therefore have decided on using this known Network Coding Topology. If we refer to Fig. 2-2 in the previous chapter, we can see that there exists a *1-hop* and *3-hop* path between two sets of nodes. These two *3-hop* paths share one common link, therefore to find the Butterfly topology, the *1-hop* and *3-hop* paths of the $B_{n\times n}$ adjacency matrix need to be constructed in order to be used in the searching algorithm.

The *1-hop* paths were constructed and stored in matrix $O_{r\times 2}$. In order to construct the *3-hop* paths, we must first construct the *2-hop* paths. To do this we create through concatenation of the *1-hop* paths, a $s\times 3$ matrix $T_{s\times 3}$ where s is the number of *2-hop* paths that exist, so that:

$$T_{s\times 3}=[t_{s1} \quad t_{s2} \quad t_{s3}] \tag{28}$$

where:

$$[t_{s1} \quad t_{s2} \quad t_{s3}]=[o_{r1} \quad o_{r2} \quad o_{h2}] \text{ if } (o_{r2}=o_{h1}) \text{ and } r<h\le|O_{r\times 2}| \tag{29}$$

We can then construct the *3-hop* paths by creating a $u\times 4$ matrix $D_{u\times 4}$, where u is the number of *3-hop* paths that exist, so that:

$$D_{u\times 4}=[d_{u1} \quad d_{u2} \quad d_{u3} \quad d_{u4}] \tag{30}$$

where:

$$[d_{u1} \quad d_{u2} \quad d_{u3} \quad d_{u4}]=\begin{cases}[o_{q1} \quad o_{q2} \quad t_{s2} \quad t_{s3}] \text{ if } (o_{q2}=t_{s1}) \text{ and } s<q\le O_{r\times 2}\\ [t_{s1} \quad t_{s2} \quad t_{s3} \quad o_{q2}] \text{ if } (t_{s3}=o_{q1}) \text{ and } s<q\le T_{s\times 3}\end{cases} \tag{31}$$

a concatenation of an *1-hop* path with a *2-hop* path or a concatenation of a *2-hop* path with an *1-hop* path .

Our algorithm then uses (26) and (31) to find the Butterfly topology. We create a new $c\times 4$ matrix $H_{c\times 4}$ where c is the number of *3-hop* paths that have first and last entries corresponding to the *1-hop* paths:

$$H_{c\times 4}=[d_{u1} \quad d_{u2} \quad d_{u3} \quad d_{u4}] \text{ if } (o_{l1}=d_{u1}),(o_{l2}=d_{u4}) \text{ and } u<l\le|O_{r\times 2}| \tag{32}$$

We then search within the matrix of (32), (which is the potential Butterfly's "wings") to see whether there are rows within it so that:

$$(d_{u2} \quad d_{u3}) = (d_{p2} \quad d_{p3}) \text{ and } u \le p \le |H_{c\times 4}| \tag{33}$$

This will confirm a shared link between the wings, and therefore verify the presence of a Butterfly topology. Two examples of this method can be found in chapter 4.

We also applied this method to two variations on the Butterfly topology. In the first variation we search for a shared node instead of a shared link and use the *1-hop* paths (26) and *2-hop* paths (28) to find the topology, therefore in this case $H_{c\times 3}$ will be a $c\times 3$ matrix where c is the number of *2-hop* paths that have first and last entries corresponding to the *1-hop* paths such that

$$H_{c\times 3} = [d_{u1} \quad d_{u2} \quad d_{u3}] \text{ if } (o_{l1} = d_{u1}), (o_{l2} = d_{u3}) \text{ and } u \le l \le |O_{r\times 2}| \tag{34}$$

We then search with in the matrix in (34), (which is the potential Butterfly's "wings") to see whether there are rows within it so that:

$$(d_{u2} = d_{p2}) \text{ and } u \le p \le |H_{c\times 3}|. \tag{35}$$

This will confirm a shared node between the wings, and therefore verify the presence of the topology we searched for. Two examples of this method can be found in chapter 5.

In the second variation, we search for a shared path instead of a shared link and use the *1-hop* paths (26) and *4-hop* paths (36) to find the topology, and we also construct the *4-hop* paths through concatenation. This may be of a *1-hop* path with a *3-hop* path or a concatenation of a *3-hop* path with a *1-hop* path to create the matrix:

$$W_{y\times 5} = [w_{y1} \quad w_{y2} \quad w_{y3} \quad w_{y4} \quad w_{y5}] \tag{36}$$

where:

$$\begin{bmatrix} w_{y1} & w_{y2} & w_{y3} & w_{y4} & w_{y5} \end{bmatrix} = \begin{cases} \begin{bmatrix} o_{q1} & o_{q2} & d_{u2} & d_{u3} & d_{u4} \end{bmatrix} \text{ if } (o_{q2} = d_{u1}) \text{ and } u \le q \le O_{r\times 2} \\ \begin{bmatrix} d_{u2} & d_{u2} & d_{u3} & d_{u4} & o_{q1} \end{bmatrix} \text{ if } (d_{u4} = o_{q1}) \text{ and } u \le q \le D_{u\times 3} \end{cases} \quad \textbf{(37)}$$

Therefore in this case $H_{c\times 5}$ will be a $c\times 5$ matrix where c is the number of *4-hop* paths that have first and last entries corresponding to the *1-hop* paths such that:

$$H_{c\times 5} = \begin{bmatrix} w_{y1} & w_{y2} & w_{y3} & w_{y4} & w_{y5} \end{bmatrix} \text{ if } (o_{l1} = w_{y1}), (o_{l2} = w_{y5}) \text{ and } y \le l \le |O_{r\times 2}| \quad \textbf{(38)}$$

We then search within the matrix of (38), (which is the potential Butterfly's "wings") to see whether there are rows within it so that:

$$(w_{y2} = w_{p2}), (w_{y3} = w_{y3}), (w_{y4} = w_{p4}) \text{ and } y \le p \le |H_{c\times 5}| \quad \textbf{(39)}$$

This will confirm a shared path between the wings, and therefore verify the presence of the topology we searched for. Two examples of this method can be found in chapter 6.

This new method of finding opportunities of Network Coding using topology completed the search of a network much faster than the iterative looping and cross-correlation methods, with less computational intensity. The results of the implementation of this method can be found in chapter 7.

In this chapter, we introduced Finding Opportunities for Network Coding using topology together with some methods to provide the opportunity to implement our new method.

CHAPTER 4 - SHARED LINK IMPLEMENTATION

In order to be able to comment on the practicality of implementing our new method, we first implemented our method on a topology that will test whether a shared link exists. Two examples of how the Butterfly topology can be found in a network will be presented in this chapter.

4.1 Background

The topology we decided on to search for first was the Basic Butterfly topology, which can be seen in figure 4-1.

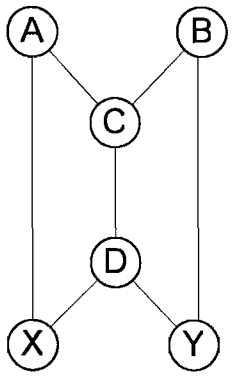


Figure 4-1- Basic Butterfly Topology

From this figure, we can see that there exists a 1-hop path [A X] as well as a 3-hop path [A C D X] between nodes A and X. The same can be seen between nodes B and Y. We can also see that these two 3-hop paths have one common link (or hop) – [C D]. We can therefore use this path information to search for the topology in a larger network.

4.1.1 Implementation Example 4-1

Initial Step (Steps i – iii of our method)

To illustrate how we find the Butterfly topology in a larger network, we use the 8 node network depicted in figure 4-2a, from which we derived an 8x8 connection matrix figure 4-2b. In this network we decided that the network may not be cyclic, and therefore only communication in one direction (down) will be allowed.

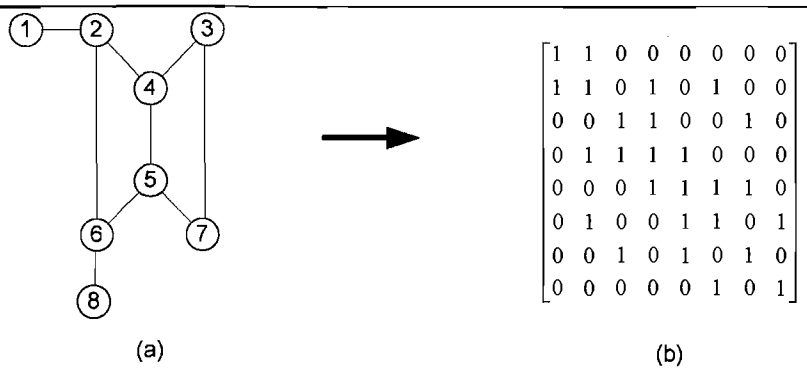


Figure 4-2 a) Network for Implementation Example 1 b) Connection matrix of Network

Finding required topology (Step iv)

In order to search the 8×8 connection matrix for the known topology, we store vectors containing all of the *1-hop* and *3-hop* paths in the network. The original adjacency matrix (figure 4-2b) is used to get all the 1-hop paths, which is stored in $O_{r\times 2}$.

$$O_{9\times 2} = [(1\ \ 2)(2\ \ 4)(2\ \ 6)(3\ \ 4)(3\ \ 7)(4\ \ 5)(5\ \ 6)(5\ \ 7)(6\ \ 8)] \quad \textbf{(40)}$$

From the paths in (40) we create *2-hop* paths, by concatenating paths that share a first or last node and storing it in $T_{s\times 3}$. In other words (1 2) and (2 4) becomes (1 2 4) etc.

$$T_{8\times 3} = [(1\ \ 2\ \ 4)(1\ \ 2\ \ 6)(2\ \ 4\ \ 5)(2\ \ 6\ \ 8)(3\ \ 4\ \ 5)(4\ \ 5\ \ 6)(4\ \ 5\ \ 7)(5\ \ 6\ \ 8)] \quad \textbf{(41)}$$

By combining the paths in (40) and (41) in a similar manner, we create *3-hop* paths and store it in $D_{u\times4}$. In other words, (1 2) and ([2 4 5) or (4 5) and (1 2 4) becomes (1 2 4 5) etc.

$$D_{b\times4}=\left[(1\ 2\ 4\ 5)(1\ 2\ 6\ 8)(2\ 4\ 5\ 6)(2\ 4\ 5\ 7)(3\ 4\ 5\ 6)(3\ 4\ 5\ 7)(4\ 5\ 6\ 8)\right] \quad (42)$$

We then compare the paths in (40) with those in (42) and we then store all the *3-hop* paths that share start and end nodes with *1-hop* paths in $H_{c\times4}$.

$$H_{2\times4}=\left[(2\ \ 4\ \ 5\ \ 6)(3\ \ 4\ \ 5\ \ 7)\right] \quad (43)$$

Within these paths, we then search for a common second and third element and store the second and third elements in *ComNPath* (the second element is the node where encoding should take place), the two first elements are stored in *ButTop* and the two last elements (where information should be decoded) are stored in *ButBot*.

$$ComNPath=\left[(4\ \ 5)\right] \quad (44)$$

$$ButTop=\left[(2\ \ 3)\right] \quad (45)$$

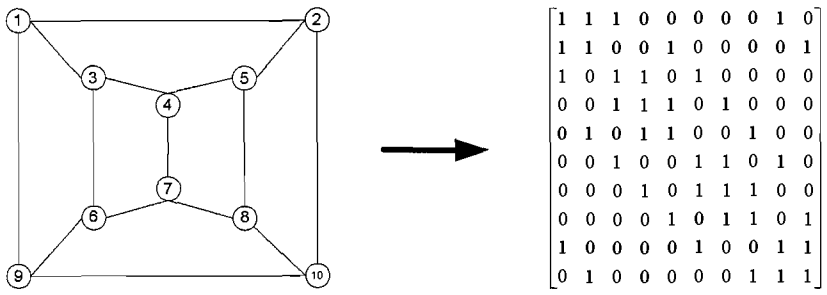
$$ButBot=\left[(6\ \ 7)\right] \quad (46)$$

These three vectors are used to know which nodes should have what encoding/decoding functionality and also for cases where more than one Butterfly per network is found. In such cases, only Butterflies that are disjoint (don't share any of the nodes in the three vectors) can be used simultaneously. An example of this case can be seen in implementation example 4-2.

4.1.2 Implementation Example 4-2

Initial Steps (i – iii)

It happens in networks with $C = 3$ (meaning each node is connected to three other nodes) and higher, that networks may contain more than one Butterfly. An example of this is the 10 node network in figure 4-3a. For this network $C = 3$.



$$\begin{bmatrix} 1&1&1&0&0&0&0&0&1&0 \\ 1&1&0&0&1&0&0&0&0&1 \\ 1&0&1&1&0&1&0&0&0&0 \\ 0&0&1&1&1&0&1&0&0&0 \\ 0&1&0&1&1&0&0&1&0&0 \\ 0&0&1&0&0&1&1&0&1&0 \\ 0&0&0&1&0&1&1&1&0&0 \\ 0&0&0&0&1&0&1&1&0&1 \\ 1&0&0&0&0&1&0&0&1&1 \\ 0&1&0&0&0&0&0&1&1&1 \end{bmatrix}$$

Figure 4-3 a) Network for Implementation Example 2 b) Connection matrix of Network

Finding the required topology (Step iv)

In order to search the 10×10 connection matrix for the known topology, we store vectors containing all of the *1-hop* and *3-hop* paths in the network. The original connection matrix (Figure 4-3b) is used to get all the 1-hop paths, stored in $O_{r\times 2}$.

$$O_{15\times 2} = [(1\ \ 2)(1\ \ 3)(1\ \ 9)(2\ \ 5)(2\ \ 10)(3\ \ 4)(3\ \ 6)(4\ \ 5)(4\ \ 7)(5\ \ 8)(6\ \ 7)(6\ \ 9)(7\ \ 8)(8\ \ 10)(9\ \ 10)] \quad (47)$$

In this case we decided to allow cyclic paths, meaning that the "reverse" of the above *1-hop* paths are also possible, and will also be considered in the creation of the *2-hop* and *3-hop* paths. Through concatenation (as was described in detail for the first implementation example), the *2-hop* and *3-hop* paths can then be found and stored in $T_{s\times 3}$ and $D_{u\times 4}$ respectively.

We again use the paths in $O_{r\times 2}$ and $D_{u\times 4}$ to determine whether there are any 1-hop paths and 3-hop paths that share the same start and end nodes, and this is stored in *ComDPath*.

$$ComDPath = [(1\ \ 3\ \ 6\ \ 9)(2\ \ 5\ \ 8\ \ 10)(3\ \ 4\ \ 7\ \ 6)(4\ \ 3\ \ 6\ \ 7)(4\ \ 5\ \ 8\ \ 7)(5\ \ 4\ \ 7\ \ 8)] \quad (48)$$

This vector is then searched to determine whether any two of the 3-hop paths in *ComDPath* contains a shared second hop:

(1 3 6 9) and (4 3 6 7) share (3 6)
(4 5 8 7) and (2 5 8 10) share (5 8)
(3 4 7 6) and (5 4 7 8) share (4 7)

We can therefore conclude that there are 3 Butterflies present in the network in figure 4-3. The three Butterflies offer three opportunities for the implementation of Network Coding. However, if we look at the information stored in *ComNPath*, *ButTop* and *ButBot* we see, that the three cannot co-exist, as they all share some nodes. We can then select one of the three for implementation of Network Coding.

If we choose the centre butterfly in nodes (3 4 7 6) and (5 4 7 8) we know that node 4 should have encoding capability, thus the node should logically XOR the information received. Nodes 6 and 8 should have decoding capability, that is, they should derive the two original messages from the information received. The search is then reiterated after possible topology changes. (Steps v and vi).

This concludes the discussion of the search for the Butterfly topologies; the implementation of our method on two further topologies will be discussed in the following two chapters.

In this chapter, two examples of how the Butterfly topology can be found in a network were presented. This topology relied upon the existence of a shared link between two sets of nodes that complied with the other comparison criteria.

CHAPTER 5 - SHARED NODE IMPLEMENTATION

In chapter 4 the implementation relied upon the existence of a shared link. In this chapter we simplify the Butterfly topology to test for a shared node in order to compare the presence of this topology in networks to the presence of the Butterfly topology. We will refer to this topology as a Bowtie.

5.1 Background

The Bowtie topology we are referring to can be seen in Figure 5-1.

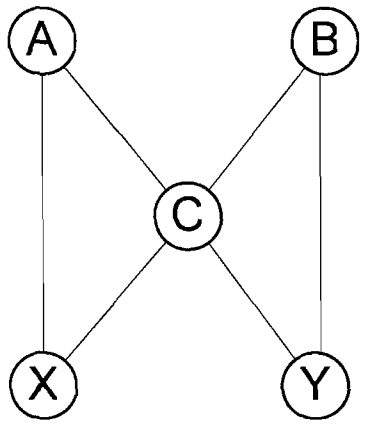


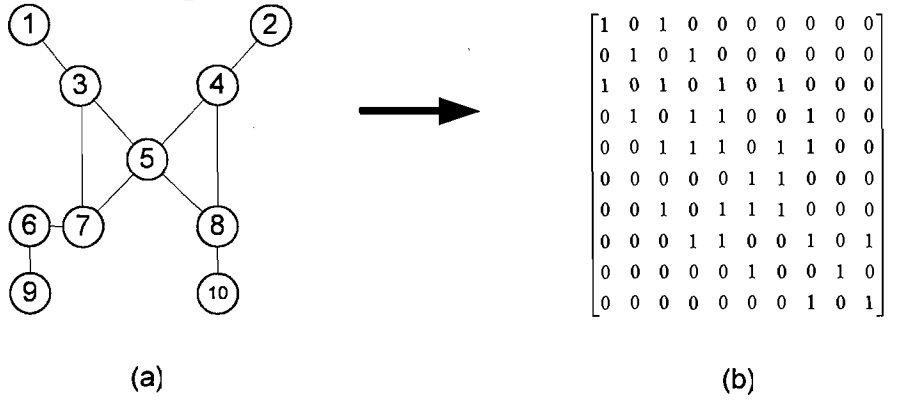
Figure 5-1 - Bowtie Topology

From this figure, we can see that there exists a *1-hop* path [A X] as well as a *2-hop* path [A C X] between nodes A and X. The same can be seen between nodes B and Y. We can also see that these two *2-hop* paths have one common node – [C]. We can therefore use this path information to search for the topology in a larger network, using the same methodology as presented in chapter 4 with a few simple modifications.

5.1.1 Implementation Example 5-1

Initial Steps (i – iii)

To illustrate how we find the Bowtie topology in a larger network, we use the 10 node network depicted in figure 5-2a from which we derived a 10x10 connection matrix (figure 5-2b).



$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 5-2 a) Network for Implementation Example 1 b) Connection matrix of Network

Finding required topology (Step iv)

In order to search the 10×10 connection matrix for the known topology, we store vectors containing all of the *1-hop* and *2-hop* paths in the network. The original connection matrix (figure 5-2b) is used to get all the *1-hop* paths, which is stored in $O_{r\times 2}$.

$$O_{11\times 2} = [(1\ \ 3)(2\ \ 4)(3\ \ 5)(3\ \ 7)(4\ \ 5)(4\ \ 8)(5\ \ 7)(5\ \ 8)(6\ \ 7)(6\ \ 9)(8\ \ 10)] \quad \textbf{(49)}$$

From the paths in $O_{r\times 2}$ we create *2-hop* paths, by concatenating paths that share a first or last node and storing it in $T_{s\times 3}$. In other words (1 3) and (3 5) becomes (1 3 5) etc.

$$T_{13\times 3} = [(1\ 3\ 5)(1\ 3\ 7)(2\ 4\ 5)(2\ 4\ 8)(3\ 5\ 7)(3\ 5\ 8)(3\ 7\ 6)(4\ 5\ 7)(4\ 5\ 8)(4\ 8\ 10)(5\ 7\ 6)(5\ 8\ 10)(7\ 6\ 9)] \quad \textbf{(50)}$$

We then compare the paths stored in $O_{r\times 2}$ with those stored in $T_{s\times 3}$ and we then store all the *2-hop* paths that share start and end nodes with *1-hop* paths in *ComDPath*.

$$ComDPath = \begin{bmatrix} (3 & 5 & 7)(4 & 5 & 8) \end{bmatrix} \quad (51)$$

Within these paths, we then search for a common **second** element and store that element in *ComNPath* (this is the node where encoding should take place), the two first elements are stored in *ButTop* and the two last elements (where information should be decoded) are stored in *ButBot*.

$$ComNPath = \begin{bmatrix} (5) \end{bmatrix} \quad (52)$$

$$ButTop = \begin{bmatrix} (3 & 4) \end{bmatrix} \quad (53)$$

$$ButBot = \begin{bmatrix} (7 & 8) \end{bmatrix} \quad (54)$$

These three vectors are used to know which nodes should have what encoding/decoding functionality and also for cases where more than one Bowtie per network is found. In such cases, only Bowties that are disjoint (don't share any of the nodes in the three vectors) can be used simultaneously. An example of this case can be seen in implementation example 5-2.

5.1.2 Implementation Example 5-2

Initial Steps (i – iii)

It happens in networks with a connectivity $C = 4$ (meaning each node is connected to four other nodes) and higher, that networks may contain more than one Bowtie. An example of this is the 10 node network in figure 5-3a. For this network $C = 4$.

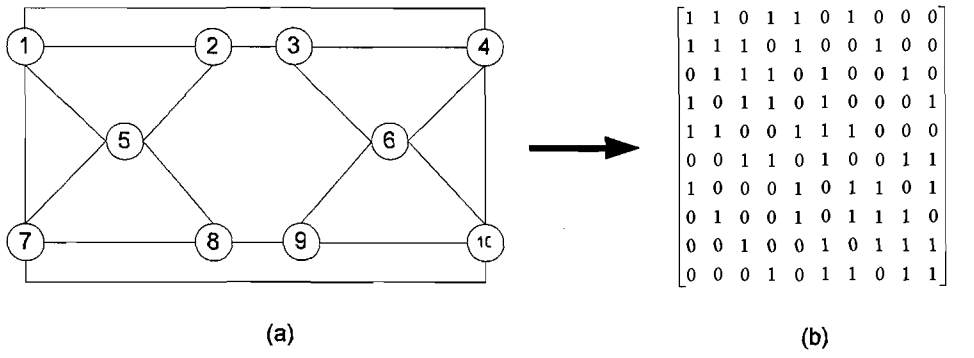


Figure 5-3 a) Network for Implementation Example 2 b) Connection matrix of Network

Finding the required topology (Step iv)

In order to search the 10×10 connection matrix for the known topology, we store vectors containing all of the 1-hop and 2-hop paths in the network. The original connection matrix (Figure 5-3b) is used to get all the 1-hop paths, stored in $O_{r\times 2}$.

$$O_{20\times 2}=[(1\ \ 2)(1\ \ 4)(1\ \ 5)(1\ \ 7)(2\ \ 3)(2\ \ 5)(2\ \ 8)(3\ \ 4)(3\ \ 6)(3\ \ 9)(4\ \ 6)(4\ \ 10)(5\ \ 7)(5\ \ 8)(6\ \ 9)(6\ \ 10)(7\ \ 8)(7\ \ 10)(8\ \ 9)(9\ \ 10)] \quad (55)$$

In this case we again decided like in example 4-2, to allow the network to be cyclic, implicating that the reverse or the above *1-hop* paths is also possible, and will also be considered in the creation of the *2-hop* paths. Through concatenation (as was described in detail for the first implementation example), the *2-hop* paths can then be found and stored in $T_{s\times 3}$.

We again use the paths in $O_{r\times 2}$ and $T_{s\times 3}$ to determine whether there are any *1-hop* paths and *2-hop* paths that share the same start and end nodes, and this is stored in *ComDPath*.

$$ComDPath=[(1\ \ 5\ \ 7)(2\ \ 5\ \ 8)(3\ \ 6\ \ 9)(4\ \ 6\ \ 10)] \quad (56)$$

This vector is then searched to determine whether any two of the *2-hop* paths in *ComDPaths* contains a shared node: (1 5 7) and (2 5 8) share (5) while (3 6 9) and (4 6 10) share (6). We can therefore conclude that there are two Bowties present in the network in figure 5-3. The two Bowties offer two opportunities for the implementation of Network Coding. This time, if we look at the information stored in *ComNPath*, *ButTop* and *ButBot* we see, that the two can co-exist, as they share no nodes. We can then select both for implementation of Network Coding.

We then know that nodes 5 and 6 should have encoding capability, thus the node should logically XOR the information received. Nodes 7, 8, 9 and 10 should have decoding capability, that is, they should derive the two original messages from the information received.

The search is then reiterated after possible topology changes. (Steps v and vi).

This concludes the discussion of the search for the Bowtie topology; one other topology will be discussed in the following chapter.

In this chapter, two examples of how the Bowtie topology can be found in a network were presented. This topology relied upon the existence of a shared node between two sets of nodes that complied with the other comparison criteria.

CHAPTER 6 - SHARED PATH IMPLEMENTATION

We decided to also generalize the topology discussed in chapter 4 to test for a shared path, instead of a link (as in chapter 4) or node (as in chapter 5). We will refer to this topology as an Extended Butterfly

6.1 Background

The Extended Butterfly topology can be seen in Figure 6-1.

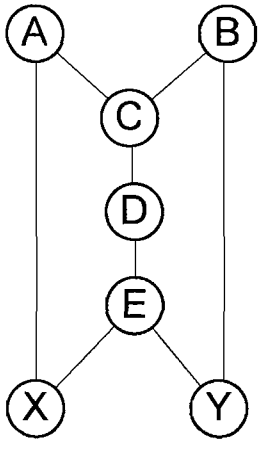


Figure 6-1 - Basic Extended Butterfly Topology

From this figure, we can see that there exists a 1-hop path [A X] as well as a 4-hop path [A C D E X] between nodes A and X. The same can be seen between nodes B and Y. We can also see that these two 4-hop paths share one common path – [C D E]. We can therefore use this path information to search for the topology in a larger network.

6.1.1 Implementation Example 6-1

Initial (Steps i – iii)

To illustrate how we find the Extended Butterfly topology in a larger network, we use the 9 node network depicted in figure 6-2a, from which we derived a 9×9 connection matrix (figure 6-2b).

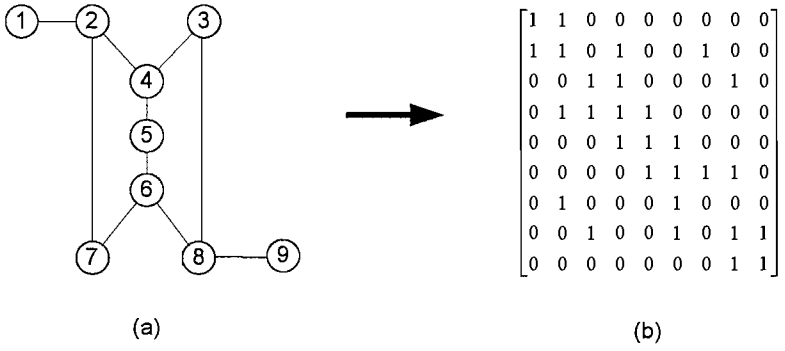


Figure 6-2 a) Network for Implementation Example 1 b) Connection matrix of Network

Finding required topology (Step iv)

In order to search the 9×9 connection matrix for the known topology, we store vectors containing all of the *1-hop* and *4-hop* paths in the network. The original connection matrix (figure 6-2b) is used to get all the 1-hop paths, which is stored in $O_{r\times 2}$.

$$O_{11\times 2}=[(1\ \ 2)(2\ \ 4)(2\ \ 7)(3\ \ 4)(3\ \ 8)(4\ \ 5)(5\ \ 6)(5\ \ 7)(6\ \ 7)(6\ \ 8)(8\ \ 9)] \quad (57)$$

From the paths in $O_{r\times 2}$ we create *2-hop* paths, by concatenating paths that share a first or last node and storing it in $T_{s\times 3}$. In other words (1 2) and (2 4) becomes (1 2 4) etc.

$$T_{9\times 3}=[(1\ \ 2\ \ 4)(1\ \ 2\ \ 7)(2\ \ 4\ \ 5)(3\ \ 4\ \ 5)(3\ \ 8\ \ 9)(4\ \ 5\ \ 6)(5\ \ 6\ \ 7)(5\ \ 6\ \ 8)(6\ \ 8\ \ 9)] \quad (58)$$

By concatenating the paths in $O_{r\times 2}$ and $T_{s\times 3}$ in a similar manner, we create 3-hop paths and store it in $D_{u\times 4}$. In other words, (1 2) and (2 4 5) or (4 5) and (1 2 4) becomes (1 2 4 5) etc.

$$D_{6\times 4}=\left[(1\ \ 2\ \ 4\ \ 5)(2\ \ 4\ \ 5\ \ 6)(3\ \ 4\ \ 5\ \ 6)(3\ \ 4\ \ 5\ \ 7)(4\ \ 5\ \ 6\ \ 8)(5\ \ 6\ \ 8\ \ 9)\right] \quad (59)$$

Further, we create 4-hop paths by concatenating the paths in $O_{r\times 2}$ and $D_{u\times 4}$ and store it in $W_{y\times 5}$.

$$W_{6\times 5}=\left[(1\ \ 2\ \ 4\ \ 5\ \ 6)(2\ \ 4\ \ 5\ \ 6\ \ 7)(2\ \ 4\ \ 5\ \ 6\ \ 8)(3\ \ 4\ \ 5\ \ 6\ \ 7)(3\ \ 4\ \ 5\ \ 6\ \ 8)(4\ \ 5\ \ 6\ \ 8\ \ 9)\right] \quad (60)$$

We then compare the paths stored in $O_{r\times 2}$ with those stored in $W_{y\times 5}$ and we then store all the 4-hop paths that share start and end nodes with 1-hop paths in *ComDPath*.

$$ComDPath=\left[(2\ \ 4\ \ 5\ \ 6\ \ 7)(3\ \ 4\ \ 5\ \ 6\ \ 8)\right] \quad (61)$$

Within these paths, we then search for a common second, third and fourth element and store these elements in *ComNPath* (the second element is the node where encoding should take place), the two first elements are stored in *ButTop* and the two last elements (where information should be decoded) are stored in *ButBot*.

$$ComNPath=\left[(4\ \ 5\ \ 6)\right] \quad (62)$$

$$ButTop=\left[(2\ \ 3)\right] \quad (63)$$

$$ButBot=\left[(7\ \ 8)\right] \quad (64)$$

These three vectors are used to know which nodes should have what encoding/decoding functionality and also for cases where more than one Extended Butterfly per network is found. In such cases, only Extended Butterflies that are disjoint (don't share any of the nodes in the

three vectors) can be used simultaneously. An example of this case can be seen in implementation example 6-2.

6.1.2 Implementation Example 6-2

Initial (Steps i – iii)

It happens in networks with a connectivity $C = 3$ (meaning each node is connected to three other nodes) and higher, that networks may contain more than one Extended Butterfly sub-network. An example of this is the 14 node network in figure 6-3a. For this network $C = 3$.

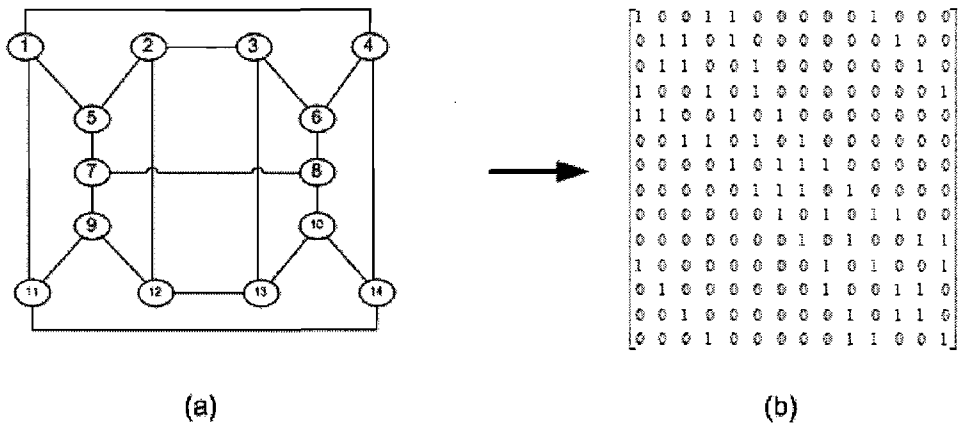


Figure 6-3 – a) Network for Implementation Example 2 b) Connection matrix of Network

Finding the required topology (Step iv)

In order to search the 14×14 connection matrix for the known topology, we store vectors containing all of the *1-hop* and *4-hop* paths in the network. The original connection matrix (figure 6-3b) is used to get all the *1-hop* paths, stored in $O_{r\times 2}$.

$$O_{20\times 2}=[(1\ 4)(1\ 5)(1\ 11)(2\ 3)(2\ 5)(2\ 12)(3\ 6)(3\ 13)(4\ 6)(4\ 14)(5\ 7)(6\ 8)(7\ 9)(8\ 10)(9\ 11)(9\ 12)(10\ 13)(10\ 14)(11\ 14)(12\ 13)] \quad (65)$$

As in examples 4-2 and 6-2 we decided to allow cyclic paths, meaning that the "reverse" of the above *1-hop* paths is also possible, and will also be considered in the creation of the *2-hop*, *3-hop* and *4-hop* paths. Through concatenation (as was described in detail for the first implementation example), the *2-hop*, *3-hop* and *4-hop* paths can then be found and stored in $T_{s\times 3}$, $D_{u\times 4}$ and $W_{y\times 5}$ respectively.

We again use the paths in $O_{r\times 2}$ and $W_{y\times 5}$ to determine whether there are any *1-hop* paths and *4-hop* paths that share the same start and end nodes, and this is stored in *ComDPath*.

$$ComDPath = [(1\ \ 5\ \ 7\ \ 9\ \ 11)(2\ \ 5\ \ 7\ \ 9\ \ 12)(3\ \ 6\ \ 8\ \ 10\ \ 13)(4\ \ 6\ \ 8\ \ 10\ \ 14)] \quad (66)$$

This vector is then searched to determine whether any two of the *3-hop* paths in *ComDPaths* contains a shared second hop:
(1 5 7 9 11) and (2 5 7 9 12) share (5 7 9) and (3 6 8 10 13) and (4 6 8 10 14) share (6 8 10)

We can therefore conclude that there are 2 Extended Butterflies present in the network in figure 6-3. The two Extended Butterflies offer two opportunities for the implementation of Network Coding. When we look at the information stored in *ComNPath, ButTop* and *ButBot* we see, that the two can co-exist, as they do not share nodes. We can then select both for implementation of Network Coding.

Hence we know that nodes 5 and 6 should have encoding capability, thus these nodes should logically XOR the information received. Nodes 11, 12, 13 and 14 should have decoding capability, that is, they should derive the two original messages from the information received. The search is then reiterated after possible topology changes. (Steps v and vi).

This concludes the discussion of the search for the Extended Butterfly topology. We also want to comment on the practicality of the implementation of our method of finding opportunities for Network Coding using topology. To do this, we searched a number of networks for the topologies described in chapters 4, 5 and 6, to see how often these topologies occur in random networks. The results obtained from repeated searches for the three discussed topologies can be found in chapter 7.

In this chapter, two examples of how the Extended Butterfly topology can be found in a network were presented. This topology relied upon the existence of a shared path between two sets of nodes that complied with the other comparison criteria.

CHAPTER 7 - Results

We implemented our new method of finding opportunities for Network Coding using network topology, which was introduced in chapter 3, to look for the topologies that were described in the previous three chapters. We wanted to determine how often these topologies occur within random networks. This then enabled us to comment on the practicality of the implementation of our method.

7.1 Shared Link Implementation

7.1.1 Experimental Setup

We implemented our method as described in chapter 3 in MATLAB on networks with between 10 and 20 nodes and searched for the Butterfly topology as described in chapter 4. We generated 1000 random connection matrices for each size network. Our random matrix generator generated 1000 unique adjacency matrices, using the following steps:

1. The user specifying the number of nodes in the network
2. The user specifying a specific connectivity, C
3. The generator then created a $n \times n$ *zero-matrix*
4. *1's* were then placed on the main diagonal of the matrix to represent the nodes being connected to themselves

5. For each of the *n* rows, the number of *1's* was then counted, and while the number of *1's* was smaller than the specified connectivity:
    a. Pseudo random numbers were generated using the Messenne Twister Algorithm [47] as implemented in MATLAB.
    b. These numbers were then divided by another random number (calculated using the computer's clock) to be a number between *0* and *1* to become a *0* or *1* after it was rounded.
6. For every *0* or *1* placed in the top triangle of the matrix (as we start at the top of the matrix and the matrix is symmetrical) the number was also mirrored across the main diagonal to the corresponding position in the bottom triangle of the matrix.
7. If a row contained more *1's* than the specified connectivity, the matrix was discarded and steps 1 – 6 were repeated.

We also checked that a particular connection matrix is not used more than once (in case the generator favoured a particular matrix and kept on testing it). We did not check whether the generated matrix was a permutation of a matrix that was already tested, due to the computational intensity of implementing that too.

The generated matrices were specified with C of between 3 and 7. The connectivity of 3 is the minimum connectivity that will allow the possibility of a butterfly network to exist, but does not guarantee the existence of a butterfly network in large networks.

The results of this implementation are described in the following sections.

7.1.2 Networks Containing Butterflies

Table 1 below summarizes the results in terms of the average number of networks out of the 1000 that contained the Butterfly topology. We discuss the meaning of these results after figure 7-1.

Table 1 - Number of Networks Containing Butterflies

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	271	918	996	1000	1000
12 nodes	211	846	991	1000	1000
14 nodes	192	805	988	1000	1000
16 nodes	173	763	985	1000	1000
18 nodes	138	714	974	998	1000
20 nodes	129	674	967	1000	1000

These results are depicted in figure 7-1.

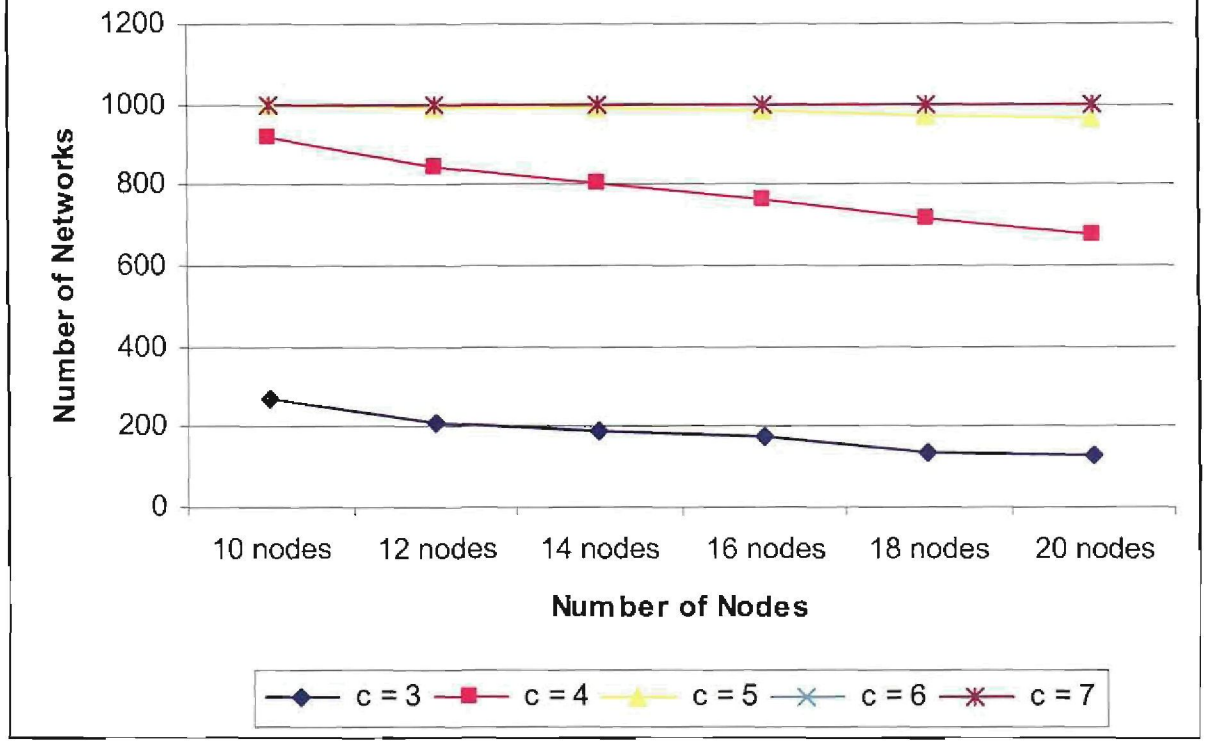


Figure 7-1 – Number of Networks Containing Butterflies

Looking at figure 7-1, we can see that when $C = 5$ or higher, most networks contained Butterflies. For $C = 5$ or lower, the number of networks containing Butterflies decrease significantly.

7.1.3 Number of Butterflies per Network

Table 2 below summarizes the results in terms of the average number of Butterflies that were found in each of the networks.

Table 2 - Number of Butterflies per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	3	7	23	66	164
12 nodes	3	6	20	52	126
14 nodes	3	5	17	47	103
16 nodes	2	5	15	41	90
18 nodes	3	4	13	35	82
20 nodes	2	4	11	32	72

These results are depicted in figure 7-2.

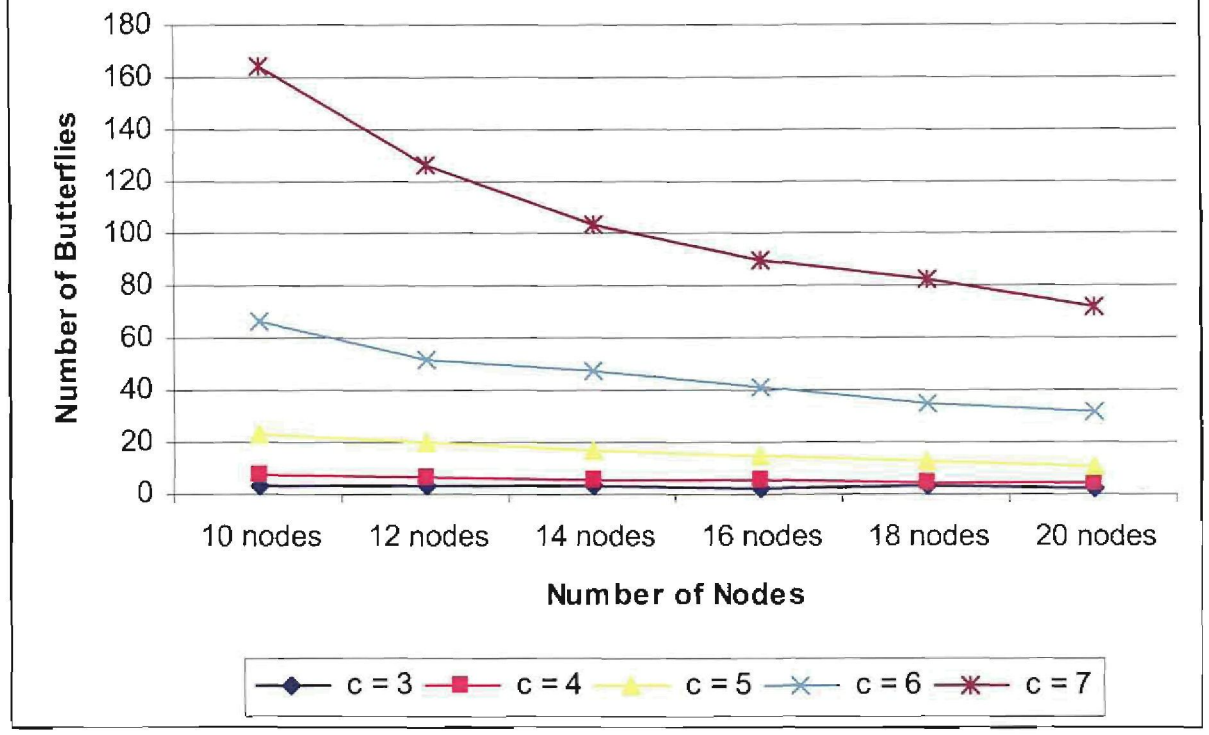


Figure 7-2 – Number of Butterflies per Network

Looking at the graph in figure 7-2, we can see at soon as $C = 6$ or higher the opportunities (number of Butterflies) are much more than with a lower connectivity, but with a lower connectivity the number of opportunities per network stayed relatively constant when the network grew.

7.1.4 Number of Disjoint Butterflies per Network

Table 3 below summarizes the results in terms of the average number of Butterflies in each of the networks found to be disjoint.

Table 3 - Number of Disjoint Butterflies per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	3	6	16	33	48
12 nodes	3	5	15	31	49
14 nodes	3	5	13	30	47
16 nodes	2	5	13	28	47
18 nodes	3	4	11	25	46
20 nodes	2	4	10	23	43

These results are depicted in figure 7-3.

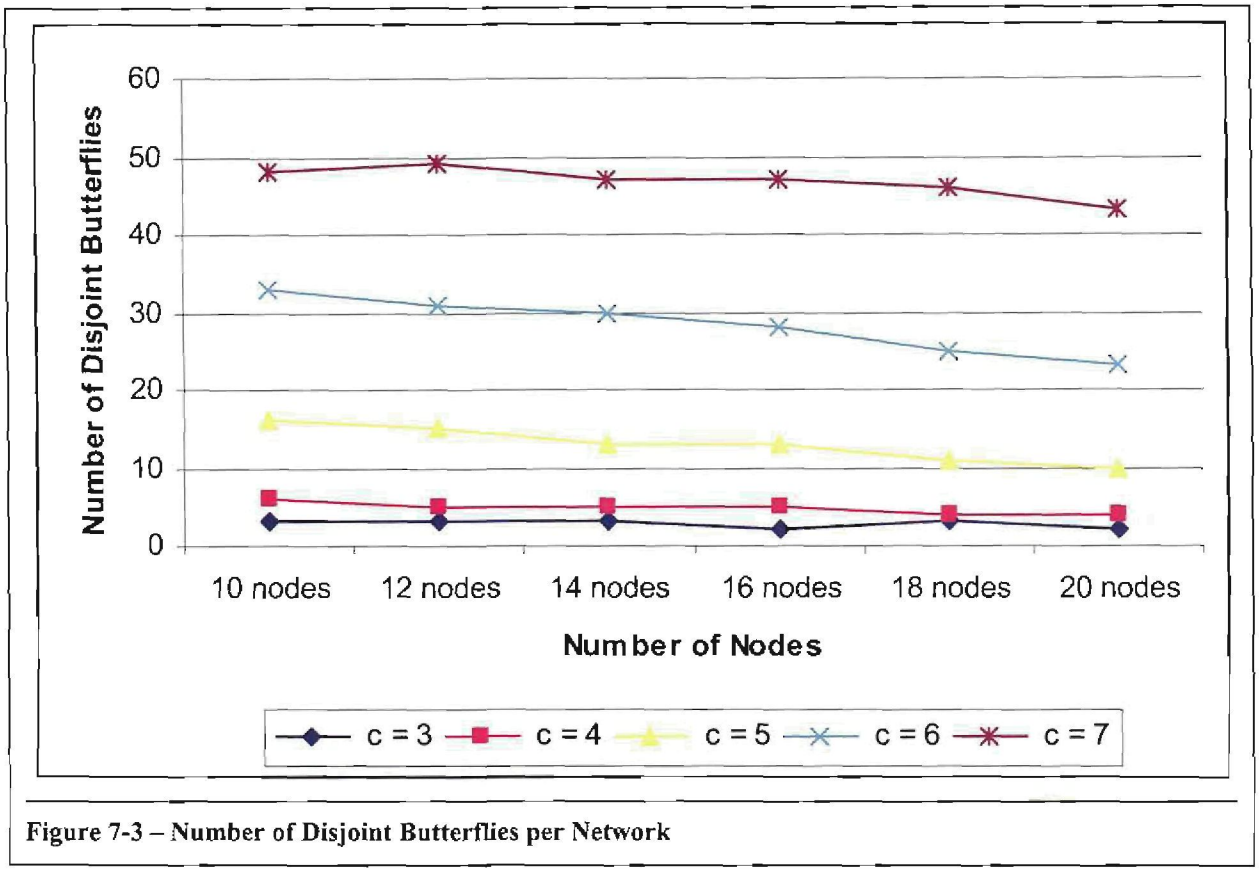


Figure 7-3 – Number of Disjoint Butterflies per Network

In figure 7-3 we can see that as the connectivity increases, the number of disjoint Butterflies (useable opportunities) per network also increases.

7.1.5 Percentage Extra Links

If we use the number of disjoint Butterflies, and add it to the number of *1-hop* paths that were in the network to begin with, we are able to calculate the percentage more *1-hop* paths there will be due to the implementation of our method. This improvement is summarized in table 4.

Table 4 – Percentage Extra Links (1)

	C = 3	*C* = 4	*C* = 5	*C* = 6	*C* = 7
10 nodes	6	12	32	66	76
12 nodes	5	9	27	56	69
14 nodes	5	8	22	50	58
16 nodes	3	8	20	43	52
18 nodes	4	6	15	35	45
20 nodes	3	5	13	30	36

From table 4 we can derive that in all the tested networks with $C = 3$ *to* 7, some improvement can be seen. The biggest improvement in comparison to the lower connectivity, could be seen when $C = 6$. The percentage extra links decreases as the network size increases.

7.1.6 Summary of Results

In all of the networks where butterflies were found when C = 3, despite not yielding very many butterflies, the butterflies were all disjoint. We saw that when C = 6 or higher, all the networks contained butterflies. Further we found that for low connectivity (C = 3 to 4), the number of butterflies stays relatively constant for all size networks, but with higher connectivity (C = 6 to 7), the number of butterflies decrease significantly with each bigger network. This can be explained by the higher possibility of islands forming in larger networks. A big increase in the percentage extra links if Network Coding were implemented could be seen when $C = 6$ (or higher), this percentage decreases with increasing network size.

7.2 Shared Node Implementation

7.2.1 Experimental Setup

To search for a Bowtie we chose the parameters as follows: $C \geq 4$ (because 4 other nodes have to be connected to the central node to form a Bowie) and we used the shared *1-hop* and *2-hop* paths to look for this topology. Again 1000 random connection matrices with between 10 and 20 nodes were explored. The results of this implementation will now be discussed.

7.2.2 Networks Containing Bowties

Table 5 below summarizes the results in terms of the average number of networks out of the 1000 that contained the Bowtie topology.

Table 5 - Number of Networks Containing Bowties

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	0	905	998	999	999
12 nodes	0	835	998	999	999
14 nodes	0	775	992	997	993
16 nodes	0	667	973	999	952
18 nodes	0	634	961	999	879
20 nodes	0	548	940	998	739

These results are depicted in figure 7-4.

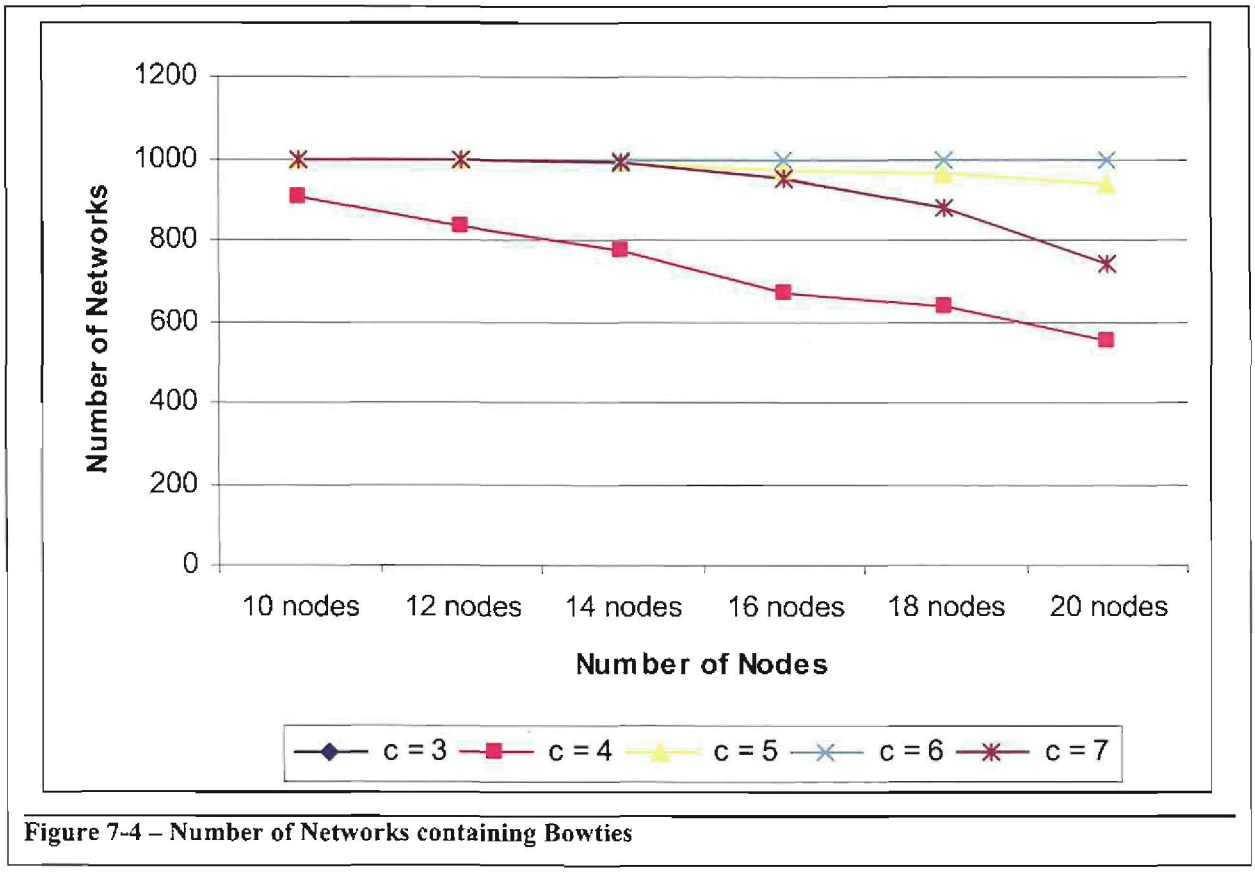


Figure 7-4 – Number of Networks containing Bowties

From figure 7-4 we can see that most networks contained Bowties and the number of networks containing Bowties seem to decrease as the network size grows, except for $C = 6$ which stays constant when more nodes join the network. As was expected networks with a $C = 3$ did not contain this topology

7.2.3 Number of Bowties per Network

Table 6 below summarize the results in terms of the average number of Bowties that was found in each of the networks.

Table 6 - Number of Bowties per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	0	3	10	24	48
12 nodes	0	3	8	20	24
14 nodes	0	3	7	16	14
16 nodes	0	2	6	15	8
18 nodes	0	2	6	12	6
20 nodes	0	2	5	12	5

These results are depicted in figure 7-5.

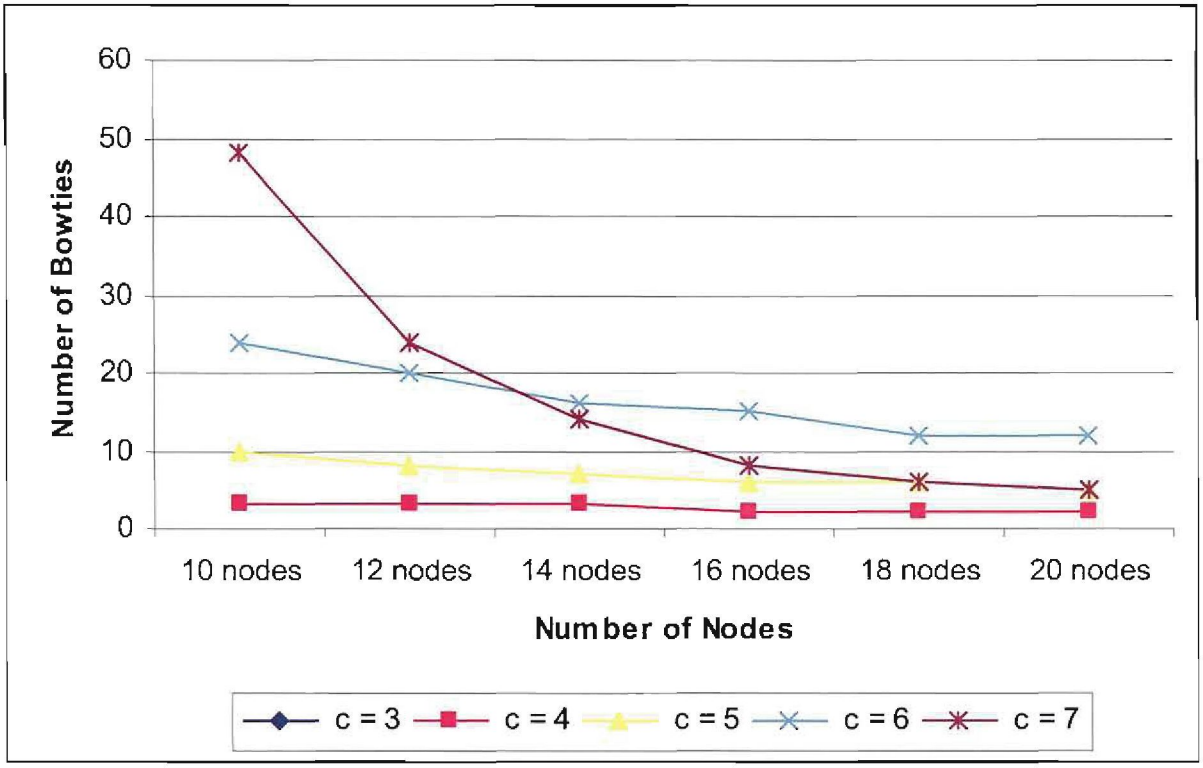


Figure 7-5 – Number of Bowties per Network

In figure 7-5 we can see when $C = 4$ *to* 5, the number of Bowties stays relatively constant when the network grows in comparison with the case when $C = 6$ *to* 7. We also observe that in networks of 14 nodes and more, $C = 6$ yielded the most Bowties per network.

7.2.4 Number of Disjoint Bowties per Network

Table 7 below summarize the results in terms of the average number of Bowties in each of the networks found to be disjoint.

Table 7 - Number of Disjoint Bowties per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	0	7	32	41	73
12 nodes	0	6	28	36	65
14 nodes	0	6	25	34	59
16 nodes	0	6	24	31	57
18 nodes	0	5	21	28	52
20 nodes	0	5	20	26	50

These results are depicted in figure 7-6.

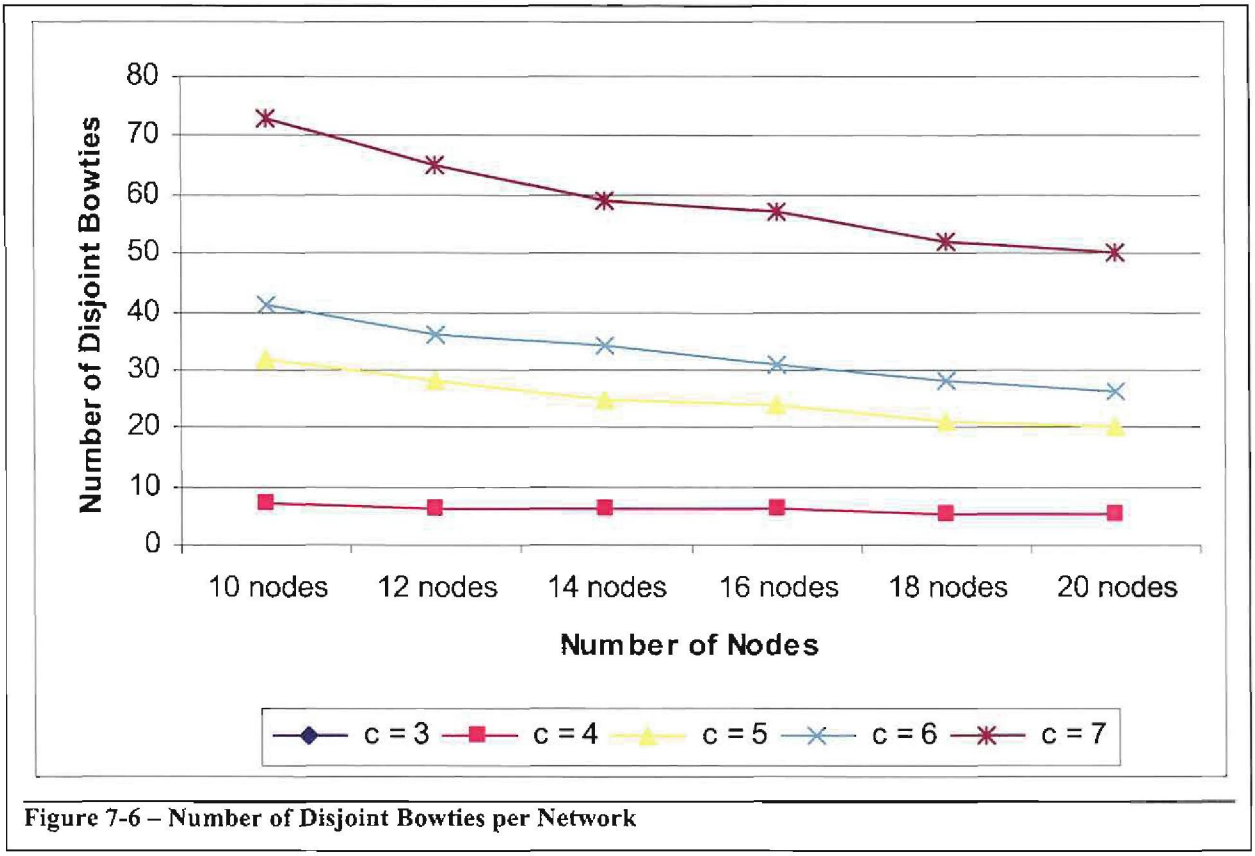


Figure 7-6 – Number of Disjoint Bowties per Network

In figure 7-6 we can see that as the capacity increases, the number of disjoint Bowties (useable opportunities) per network also increases, while the number of disjoint Bowties decreases with network size.

7.2.5 Percentage Extra Links

If we use the number of disjoint Bowties, and add it to the number of *1-hop* paths that were in the network to begin with, we are able to calculate the percentage more *1-hop* paths there will be due to the implementation of our method. This improvement is summarized in table 8.

Table 8 – Percentage Extra Links (2)

	$C = 3$	$C = 4$	$C = 5$	$C = 6$	$C = 7$
10 nodes	0	14	54	70	76
12 nodes	0	11	51	65	69
14 nodes	0	10	42	63	68
16 nodes	0	9	37	58	63
18 nodes	0	7	30	39	47
20 nodes	0	6	26	34	44

As we have already mentioned, when looking for this topology, no extra links would be created in the network for networks with $C = 3$ and lower. When $C = 5$ and higher, an improvement of almost half the original number of links will be possible. However, like with the previous topology, this improvement in extra number of links decreases as network size increases.

7.2.6 Summary of Results

Although most networks contained at least one of these Bowties, the number of Bowties per network was not very high. A $C = 6$ gave the best results in terms of presence of Bowties, for all other C's the number of networks containing Bowties decreased as the networks got bigger. This may be due to the complex nature of this topology (each node has to be connected to exactly four other nodes, two of which also have to be connected to each other on each side). Again we found that for low connectivity ($C = 4$ *to* 5), the number of Bowties stays relatively constant for all size networks, but with higher connectivity ($C = 6$ *to* 7), the number of Bowties decrease exponentially with each bigger network. We again connect this phenomenon to the higher possibility of islands forming in larger networks. An improvement in the percentage of extra links available can be seen in al networks where $C \geq 4$, which decreases as the network size increases.

7.3 Shared Path Implementation

7.3.1 Experimental Setup

Finally, we implemented our idea on the Extended Butterfly topology. For this implementation, we specified $C \geq 3$ and we used the shared *1-hop* and *4-hop* paths to look for this topology. We tested the same size and same number of random networks as for the previous two topologies. The results of this implementation will now be discussed.

7.3.2 Networks Containing Extended Butterflies

Table 9 below summarizes the results in terms of the average number of networks out of the 1000 that contained the Extended Butterfly topology.

Table 9 - Number of Networks Containing Extended Butterflies

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	410	974	1000	1000	1000
12 nodes	396	972	998	1000	1000
14 nodes	378	963	996	1000	1000
16 nodes	382	951	998	1000	1000
18 nodes	315	942	994	1000	1000
20 nodes	343	968	1000	1000	1000

These results are depicted in figure 7-7.

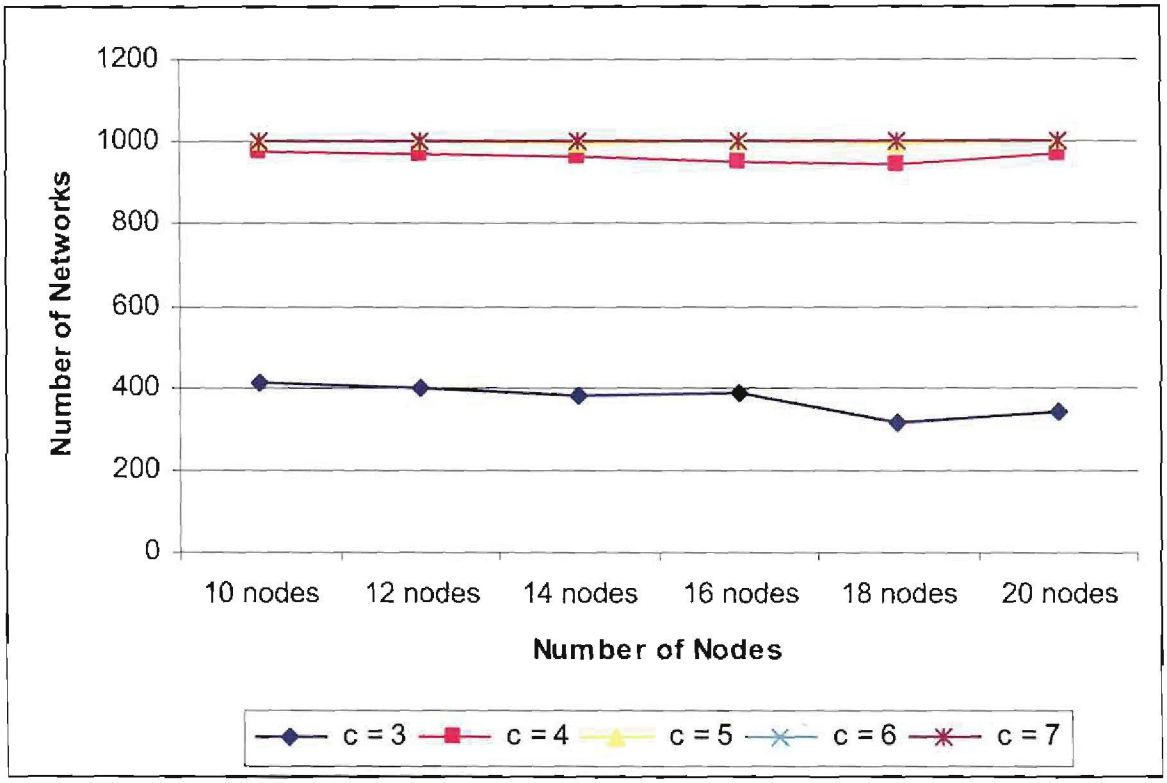


Figure 7-7 – Number of Networks containing Extended Butterflies

From figure 7-7 we can see that in almost all the networks where $C = 4$ or higher, Extended butterflies could be found, and this stayed constant regardless of the network size.

7.3.3 Number of Extended Butterflies per Network

Table 10 below summarize the results in terms of the average number of Extended Butterflies that was found in each of the networks.

Table 10 - Number of Extended Butterflies per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	2	10	32	84	186
12 nodes	2	9	28	72	154
14 nodes	2	8	25	66	136
16 nodes	2	7	23	58	123
18 nodes	2	7	21	52	106
20 nodes	2	7	19	48	101

These results are depicted in figure 7-8.

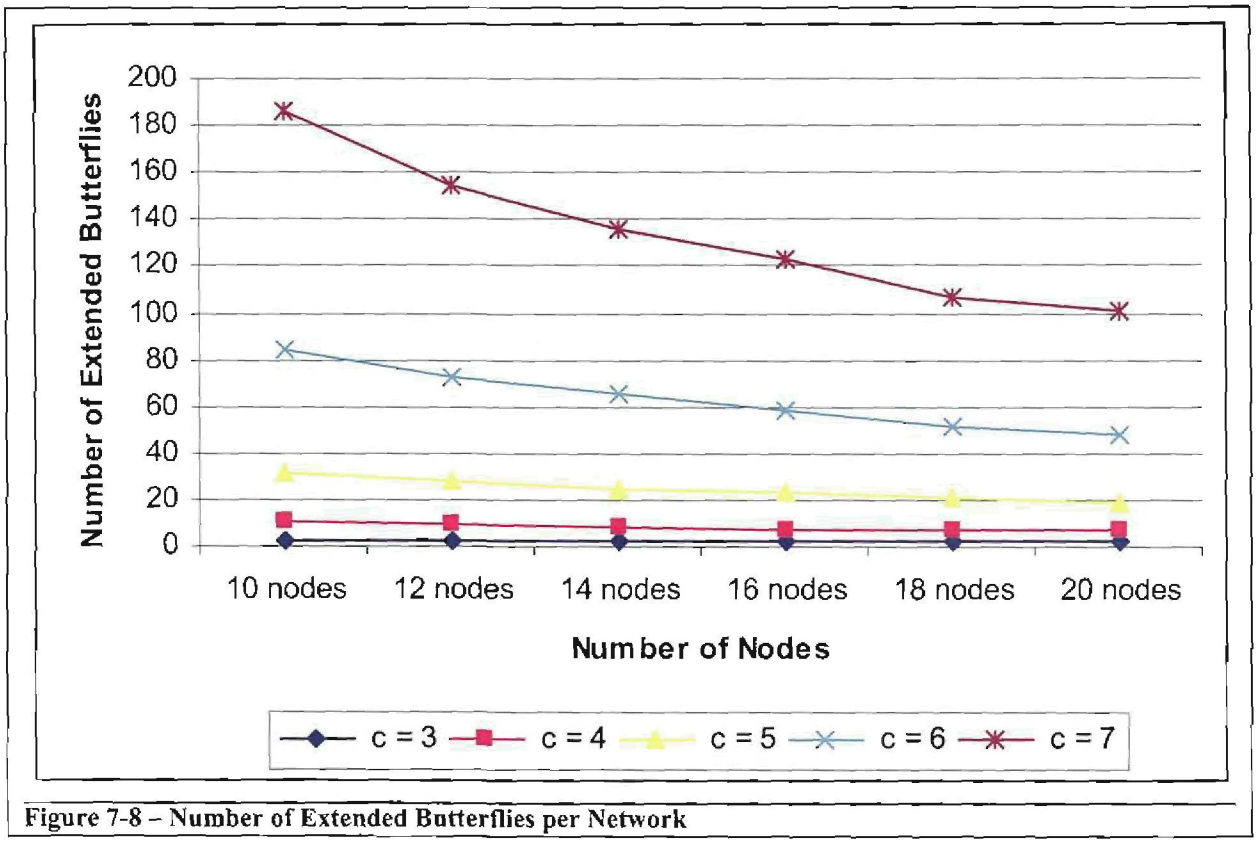


Figure 7-8 – Number of Extended Butterflies per Network

In figure 7-8 it can be seen, that $C = 6$ and higher yielded significantly more Extended Butterflies than in networks with lower connectivity.

7.3.4 Number of Disjoint Extended Butterflies per Network

Table 11 below summarize the results in terms of the average number of Butterflies in each of the networks that were found to be disjoint.

Table 11 - Number of Disjoint Extended Butterflies per Network

	c = 3	c = 4	c = 5	c = 6	c = 7
10 nodes	2	7	21	48	97
12 nodes	2	7	18	41	80
14 nodes	2	7	17	37	71
16 nodes	2	6	16	34	66
18 nodes	2	6	15	31	58
20 nodes	2	6	14	29	56

These results are depicted in figure 7-9.

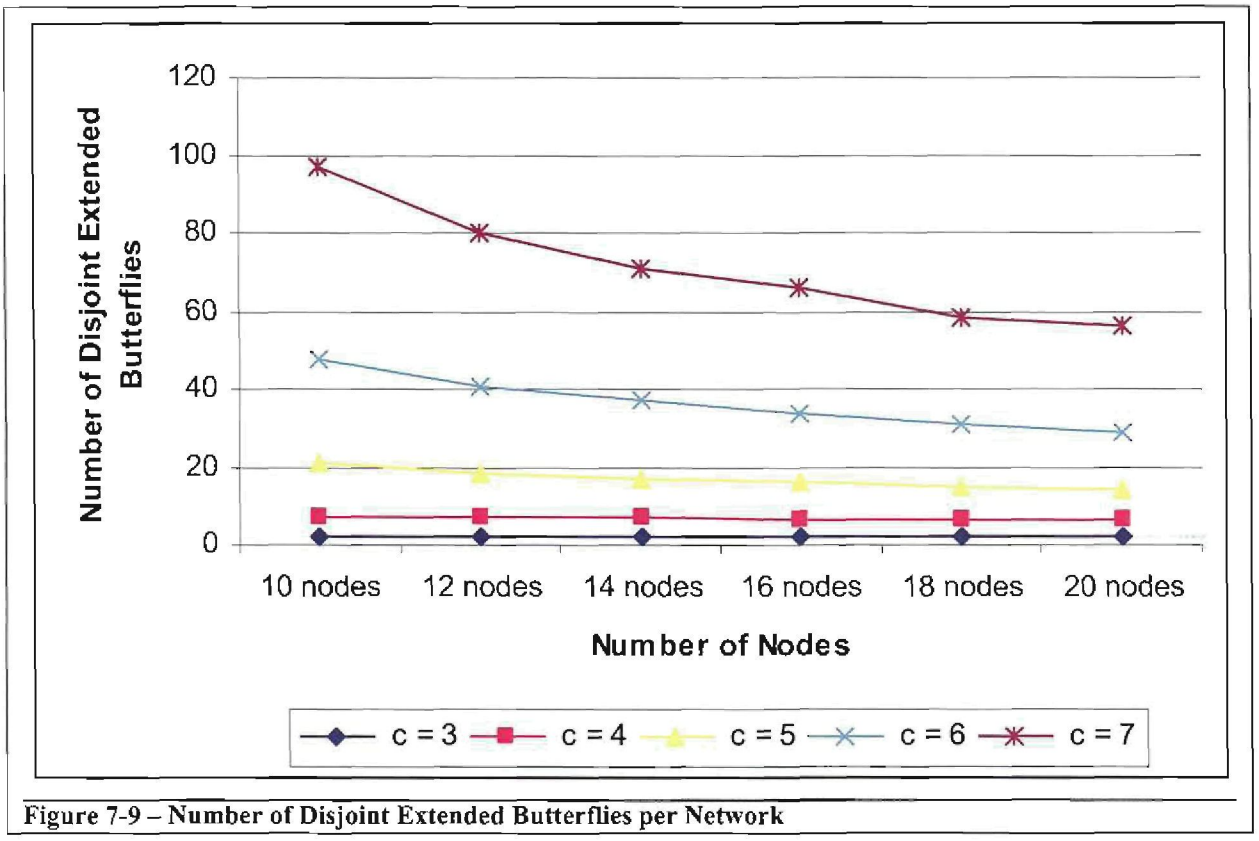


Figure 7-9 – Number of Disjoint Extended Butterflies per Network

In figure 7-9 we can see the same pattern as in figure 7-8. Again $C = 6$ and higher yielded many more disjoint Extended Butterflies than in networks with lower connectivity.

7.3.5 Percentage Extra Links

If we use the number of disjoint Extended Butterflies, and add it to the number of *1-hop* paths that were in the network to begin with, we are able to calculate the percentage more *1-hop* paths there will be due to the implementation of our method. This improvement is summarized in table 12.

Table 12 – Percentage Extra Links (3)

	$C=3$	$C=4$	$C=5$	$C=6$	$C=7$
10 nodes	4	14	42	48	76
12 nodes	4	13	33	41	71
14 nodes	3	12	28	37	62
16 nodes	3	9	25	34	57
18 nodes	3	8	21	31	51
20 nodes	3	8	18	29	45

All the tested cases showed evidence of more available links if Network Coding was implemented. In networks where $C= 7$ the biggest improvement could be seen.

7.3.6 Summary of Results

This topology shows the same tendency as was the case for the normal Butterfly. We found that more networks contained Extended Butterflies than normal Butterflies. When $C = 3$, twice as many networks contained Extended Butterflies, but there were not more Extended Butterflies than Butterflies per Network. This may be because there are not as much opportunities for *4-hop* paths in a sparsely connected network. All networks with $C = 5$ and higher contained Extended Butterflies. If Network Coding were implemented at the locations identified by our method with this topology, an improvement would be seen in all cases. The biggest improvement will be in networks with a $C= 7$.

7.4 Capacity

Apart from determining how often opportunities for the implementation of Network Coding can be found using our method, we also wanted to see what the effect of the implementation of Network Coding at these identified locations would be on the capacity of a network. Capacity of WMNs is a complex problem due to the various factors that influence it [3, 5]. Studies have shown that a throughput capacity of $0.0976\sqrt{n}$ (where n is the number of nodes) is achievable when C = 6 [7], but analytical results have shown that the throughput capacity per node decreases as the node density increases [5]. If we calculate the capacity for MANETs using MSDA, we see that we do not increase the quantitative capacity of the network by adding the extra virtual links.

We have however made a qualitative improvement if we look at the percentage extra links as described in section 7.1.5, 7.2.5 and 7.3.5. If we implement Network Coding and effectively

add more virtual links to the network utilize the existing capacity (bound by the number of nodes) better. The extra links will lead to less congestion which will lead to less delay when the information can be combined when sent over the network which will consequently improve QoS in the network.

7.5 Conclusion

Looking at the results presented in sections 7.1 – 7.4 of this chapter we can conclude the following:

1. Our method to locate opportunities for implementation of deterministic Network Coding proved fruitful, as most of the tested networks with $C = 5$ and higher contained the topologies that we searched for.

2. In general we observed that all three topologies showed significant benefit when $C = 6$, which seems to support [7] which suggested a connectivity of 6 for optimal throughput capacity of a mesh network.

3. The implementation of Network Coding in a WMN can qualitatively improve the throughput capacity of the network by better utilization of the available capacity. We show in tables 4, 8 and 12, the improvement in terms of the number of extra virtual links that are created in a network.

4. The percentage of extra virtual links that would be created when Network Coding is implemented at the locations where disjoint Butterflies were found also proves to be significant enough to encourage the implementation of our method.

5. All three topologies tend to respond the same to connectivity and network size.

From this we can then say that it would be worthwhile to implement our method, to identify opportunities for the implementation of Network Coding as the opportunities identified with our method will be significant enough.

In this chapter, we presented the results we obtained after implementing our new method of chapter 3 in 1000 networks, to look for the three topologies presented in chapters 4, 5 and 6.

CHAPTER 8 - Conclusion

In this chapter summarize the work that was done and also make mention of further observations we made that not necessarily formed part of this particular study. Finally we discuss the further work that can be done in a continuation of this work.

8.1 Summary of Work Done

This research aimed to make a positive contribution to improving the resource control problem in communication networks. More specifically, we wanted to provide a method of determining where in a wireless network, Network Coding can be implemented.

We did this by first studying the fields of wireless networks, routing and Network Coding. After becoming familiar with these fields, we collected methods to enable us to search the network for opportunities to implement Network Coding. These methods included finding the topological information of a network in routing tables and then presenting it in an adjacency matrix. We also noted that apart from using the adjacency matrix to find communication routes, it can be used to calculate the capacity of the network (using MSDA).

We identified the Butterfly topology as the known Network Coding topology to search for in the wireless network. Our own way of constructing the communication routes from the adjacency matrix was presented because exact multiplication did not provide us with all the routes between two communication nodes, but only the shortest route. We also presented our new method of looking for the Butterfly topology. The Butterfly topology was also simplified (to form a Bowtie) and generalized in terms of the Network Coding path (to form an Extended Butterfly). Two examples of our new method were discussed for each of the three topologies. These six examples showed all the ways we identified the known topologies to appear within a network.

In order to comment on the practicality of implementing Network Coding in this manner, we decided to implement it in a number of networks to see how often the topologies occur naturally. This was done, for each topology in 1000 adjacency matrices generated randomly using the Mersenne Twister algorithm in MATLAB.

8.2 Most Significant Results

From the results generated by the implementation of our method in the 1000 random adjacency matrices, we have seen that our method to locate opportunities for implementation of deterministic Network Coding have been successful. Most of the tested networks with $C = 5$ and higher contained the topologies that we searched for.

In general we observed that all three topologies showed significant benefit when $C = 6$, which seems to support [7] who suggested a connectivity of 6 for optimal throughput capacity of a mesh network. We also saw that the Extended Butterfly topology occurred most frequently in the random networks, except for the case where $C = 4$, that contained more Bowties.

The implementation of Network Coding in wireless networks can qualitatively improve the throughput capacity of the network by better utilization of the available capacity. The percentage of extra links as presented in chapter 7 showed that networks could benefit from the implementation of Network Coding and that percentage proves to be significant enough to encourage the implementation of our method.

From this we can deduce that with the implementation of our new method to locate opportunities for Network Coding to create more available links will be useful. For each

disjoint Butterfly we have one extra link available. This benefit is significant enough to encourage the implementation of our method because when we exploit these opportunities, the network would be less congested, meaning lower probability of lost packets due to long delays. This will then lead to a better utilized network.

8.3 Evaluation of our Method

The implementation of our method comes at a cost involving that "clever" nodes have to be created to be able to encode and decode messages. High connectivity is needed and there will be a route establishment delay. There also would have to be some form of centralized network control. This cost will be minimised if our method is implemented in a WMN because of its properties, especially if this WMN consisted of nodes with significant computational ability.

8.4 Further observations

In our research, we also noted that the capacity of wireless networks is a widely debated and researched issue to which a definite answer doesn't really exist yet. We also observed that the power expenditure issue in MANETs is possibly the biggest factor to keep in mind when developing technology for it.

8.5 Further work

As far as further work is concerned, it is suggested that our method be integrated in a routing protocol. This will enable us to test our method on a physical testing facility such as the one at the MERAKA institute. Other known Network Coding topologies may also be considered to see whether a more favourable topology than the Butterfly exists.

References

[1] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," In *Proceedings of the IEEE International Symposium on Information Theory* June 2003, page 442, Yokohama, Japan, 2003.

[2] Goldsmith, *Wireless Communication*, California: Stanford University, 2005.

[3] I.F. Akyildiz et al, "Wireless Mesh Networks: A survey", *Computer Networks*, vol. 47, pp. 445 – 487, 2005.

[4] Mitrokotsa, N. Komninos, C. Douligeris, (2007), "Intrusion Detection with Neural Networks and Watermarking Techniques for MANET", In *Proceedings of IEEE International Conference on Pervasive Services 2007* (ICPS'07), July 15-20, 2007, pp. 118 - 127 Istanbul, Turkey, 2007.

[5] I. F. Akyildiz and X. Wang, "A Survey on Wireless Mesh Networks", *IEEE Communications Magazine*, vol. 43, no. 9, pp.23-30, 2005.

[6] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Networks" *IEEE Wireless Communication. Magazine,* vol. 10 pp. 8-14.

[7] P .Gupta and P.R., Kumar, "The capacity of wireless networks", IEEE Transactions on Information Theory, Volume 46, Issue 2, pp.388 – 404, 2000.

[8] L. Kleinrock, J. Silvester, "Optimum transmission radii for packet radio networks or Why six is a magic number", in: *Proceedings of the IEEE National Telecommunications Conference*, Birmingham, Alabama, pp. 4.3.1–4.3.5, 1978.

[9] J. Zhang and W. K. G. Seah, "Topology–based capacity analysis for ad hoc networks with end-to-end delay constraints," In *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication,* 31 May-2 June 2004, vol.2, pp. 541- 544, 2004.

[10] Ning Li, Yan Guo, Shaoren Zheng, Chang Tian, Jun Zheng, "A Matrix-Based Fast Calculation Algorithm for Estimating Network Capacity of MANETs," La Jolla,

CA, pp. 407-412, *Systems Communications* (ICW'05, ICHSN'05, ICMCS'05, SENET'05), 2005

[11] E. Royer and C.K. Toh., "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", *IEEE Personal Communication*, Volume 6, Issue 2, pp. 46-55, 1999.

[12] RC Shah, JM Rabaey "Energy aware routing for low energy ad hoc sensor networks", *Proc.of IEEE Wireless Communications and Networking Conference,* Orlando FL, USA, 2002.

[13] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum et L. Viennot, "Optimized Link State Routing Protocol", *IEEE INMIC*, Pakistan 2001.

[14] A. Al-Ali, "Evaluation of the Wireless OSPF Routing Protocol," Graduation Report, Imam Muhammad Bin Saud Islamic University, Saudi Arabia, 2008.

[15] A Mukhija, "Reactive Routing Protocol for Mobile Ad-hoc Networks", Masters Thesis, IIT Delhi, 2001.

[16] D.B. Johnson, D.A. Maltz, and J. Broch. "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks." *Ad Hoc Networking*, pp. 139-172, Addison-Wesley, 2001

[17] Charles E. Perkins and Elizabeth M. Royer. "Ad hoc On-Demand Distance Vector Routing." Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[18] L Wang, S Olariu, "A Two-Zone Hybrid Routing Protocol for Mobile Ad Hoc Networks" , *IEEE Transactions On Parallel and Distributed Systems*, , 2004

[19] M.D.N. Ahmed, A. Pandey, N. Kumar, P. Gupta "A Hybrid Routing Protocol for Large Scale Mobile Ad Hoc Networks with Mobile Backbone", *Proc. of 12th Int. Conf. Advanced Computing and Communication (ADCOM)*, 2004

[20] V. Ramasubramanian, Z. J. Haas and E.G. Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks", *Proc of MobiHoc '03*, 2003.

[21] Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Transactions on Information Theory*, vol. 46, July 2000, pp. 1204-1216, 2000.

[22] R. W. Yeung "Distributed Source Coding for Satellite Communications", *IEEE Transactions on Information Theory*, May 1999, vol. 43, pp. 1111-1120, 1999.

[23] P. Elias, A. Feinstein, and C. E. Shannon, "Note on maximum flow through a network," *IRE Trans. Information Theory*, vol. 2, pp. 117–119, 1956.

[24] R. Koetter and M. Médard, "Beyond routing: an algebraic approach to Network Coding", In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol.1, pp. 122- 130, 2002.

[25] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright and K. Ramchandran "The Benefits of Network Coding for Peer-to-Peer Storage Systems", *NetCod Workshop*, January 2007.

[26] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding", IEEE Transactions on Information Theory", February 2003, vol. 49, pp. 371- 381, 2003.

[27] C. Fragouli, J. Widmer, J. Le Boudec "Network Coding: An instant primer", *ACM SIGCOMM Computer Communication Review*, January 2006, vol.36, pp. 63 - 68, 2006.

[28] C. Fragouli, J. Widmer, J. Le Boudec "On the Benefits of Network Coding for Wireless Applications", *Netcod*, 2006.

[29] I. A. Qazi, P.Gandhi, "Performance Evaluation of Wireless Network Coding under Practical Settings", University of Pittsburgh, 2007.

[30] C Gkantsidis and P Rodriguez, "Cooperative security for Network Coding file distribution" *IEEE Infocom*, 2006

[31] N Cai and RW Yeung, "Secure Network Coding" *Proc. of IEEE International Symposium on Information Theory*, 2002.

[32] A.R. Lehman, "Network Coding", Ph. D dissertation, Massachusetts Institute Of Technology, United States of America, 2005.

[33] I. Lakatos, J. Worrall, G. Currie, *The Methodology of Scientific Research Programmes*, Cambridge University Press, 2003.

[34] H.G. Gauch, *Scientific Method in Practice*, Cambridge University Press, 2003.

[35] Random House, *Webster's Unabridged Dictionary*, Deluxe Edition, Random House Press, 2005.

[36] S. Hawker, M. Waite, *Oxford Paperback Dictionary and Thesaurus*, Second Edition, Oxford University Press, 2007

[37] R.P. Draves, C. King, S. Venkatachary and B.D. Zill, "Constructing optimal IP routing tables" *Proc. of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 88-97, 1999.

[38] J Moy, "RFC 2328: OSPF version 2", April 1998.

[39] T. Larsson and N. Hedman, "Routing Protocols in Wireless Ad-hoc Networks – A Simulation Study", Masters thesis, Lulea University of Technology, Stockholm, Sweden, 1998.

[40] John Evans, Deploying *IP and MPLS QoS for Multiservice Networks: Theory and Practice* , Clarence Filsfils (Morgan Kaufmann), 2007.

[41] R. Kelley, Iterative Methods for Optimization. SIAM, Philadelphia, 1999.

[42] D. Casasent and D. Psaltis "New optical transforms for pattern recognition", *Proceedings of the IEEE*, Volume 65, Issue 1, pp. 77- 84, 1977.

[43] C Knapp and G Carter, "The generalized correlation method for estimation of time delay", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume 24, Issue 4, pp. 320- 327, 1976.

[44] JP Lewis, "Fast normalized cross-correlation", *Vision Interface*, 1995.

[45] M.J. Grobler, A.S.J. Helberg, H.Marais and C.C. Naudé, "An Application of deterministic Network Coding in MANETs", *Proc. of the Southern African Networks and Applications Conference (SATNAC)*, Eastern Cape, South Africa, September 2008.

[46] K Jain, "Security based on network topology against the wiretapping attack", *IEEE Wireless Communications*, Volume 11, Issue 1, pp. 68- 71, 2004.

[47] M. Matsumoto, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Volume 8 , Issue 1, pp. 3 – 30, 1998

Appendix A - Conference Contributions from this Thesis

An Application of deterministic Network Coding in MANETs

Presented at:

Southern African Telecommunication Networks and Applications Conference (SATNAC)

September 2008

Wild Coast Sun, Eastern Cape, South Africa

Status:

Received an award for *Best Reviewed Paper.*

An Application of deterministic Network Coding in MANETs

Leenta M.J. Grobler, Albert S.J. Helberg, Henri Marais, Coenraad C. Naudé
School for Electrical, Electronic and Computer Engineering
North-West University - Potchefstroom Campus
Tel: (018) 299-1961 Tel Fax: (018) 299-1977
leenta.grobler@nwu.ac.za

Abstract—Recent advances in methods to increase network capacity has lead to the introduction of a new concept called Network Coding. Network Coding holds the promise of increasing capacity in MANETs as well. However, very few practical results and implementations are available. In this paper we propose the use of deterministic Network Coding in a MANET. We present and describe the tools used to implement and investigate this new method.

Index Terms — Capacity, Mobile Ad-hoc Networks, Network Coding, Network Topology.

I. INTRODUCTION

THE universal need for better control over resources in communication networks is a problem that is studied continuously. Maximum network capacity needs to be defined and then utilized to ensure that as much information as possible is delivered in the most economic manner. One way in which this can be addressed, is by using Network Coding. Research on Network Coding to date led to a wide variety of theoretical results especially in wired networks. In wired networks, Network Coding can be implemented deterministically.

We also want to address the capacity and resource control problem in Mobile Ad-hoc Networks (MANETs). MANETs are complex networks of which the topology changes constantly and unpredictably. One documented implementation of Network Coding in MANETs is Random Network Coding [1].

In this paper, we investigate the opportunities that the properties of MANETs provide for practical implementation of deterministic Network Coding.

This paper is structured in the following manner:

We first give an introduction to MANETs. We then look at the maximum throughput capacity with the Min cut Max flow theorem.

Network Coding is explained together with the advantages possible by combining the inherent properties of MANETs and the advantages of Network Coding. We introduce our new method to implement deterministic Network Coding in MANETs. Finally we present a modified capacity calculation algorithm to test the efficiency of our new method.

II. MOBILE AD-HOC NETWORKS

A Mobile Ad-hoc Network (MANET) is a type of wireless network that typically consist of mobile routers and in some cases also laptop computers. These wireless mobile nodes are connected by wireless links to form a varying arbitrary network topology.

Because these nodes are free to move randomly and organize themselves arbitrarily, the topology may change rapidly and unpredictably. The management of ad-hoc networks is decentralized. That implies that each node present in the network act as a forwarding node, forwarding messages to other nodes. The selection of forwarding nodes changes dynamically with the topology. A MANET may operate as a standalone network, or be connected to a larger network such as the internet.

In other words a MANET is a network that is highly mobile, consisting of nodes with high processing power that receive frequent routing updates.

The advantages and disadvantages of MANETs can be summarized as follows [2]:

Advantages:

i. Adaptability
ii. Flexibility
iii. Efficient communication in environments with little or no infrastructure.

Disadvantages:

i. Vulnerable to attacks
ii. Congestion in the network and poor utilization of the network.

These disadvantages manifest as a result of a combination of factors: The use of an open medium, with a decentralized nature and a topology that changes dynamically, with poor physical security.

One of the biggest challenges in working with MANETs is to determine the network capacity. When the capacity is known, using [3, 4], we can make use of Network Coding to utilize this capacity,

In our research we focus on developing a new technique to reduce congestion in a MANET while improving the utilization using Network Coding.

III. MIN CUT MAX FLOW THEOREM

As an introduction to the maximum capacity of a network, we refer to the Min Cut, Max Flow theorem

[5]. Alternative methods include Steiner tree packing, Linear Programming and a method called MSDA [3, 4].

Min-cut Max-flow theorem:

The maximum flow of information in a network is equal to the sum of the cut of the link capacities [5].

Using the theorem - we cut the network, separating the sender and receiver nodes (as can be seen in figure 1) to cross as little of the links as possible to determine the minimum cut. We can see that the minimum cut, and therefore the maximum flow (or maximum throughput capacity) of this network is equal to 2.

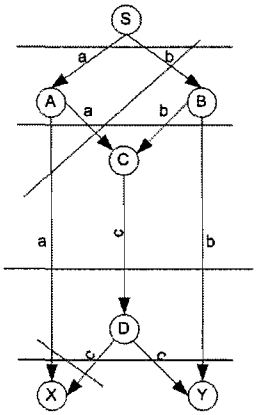


Fig. 1. Butterfly Network explaining Min Cut, Max Flow Theorem

The only way that we can utilize this maximum throughput capacity, is by using Network Coding [6].

IV. INTRODUCTION TO NETWORK CODING

A. What is Network Coding?

Network Coding is a field that was first introduced in 2000 [7] as a method to utilize the maximum capacity of a network and maximize the flow of information in that network. It suggested coding at packet level in wired P2P networks. The idea sprouts from research done in [8] on satellite communications using a source coding system which consists of multiple sources, encoders, and decoders.

Applications where Network Coding can be especially useful are MANETs, Power Line Communication as well as Wireless Sensor Networks.

B. How does Network Coding work?

We will now explain Deterministic Network Coding: We use the Butterfly network from [9], as depicted in figure 2, to explain the how Network Coding works. The links in the figure all have unit capacity and messages *a* and *b* are binary.

Two nodes, *A* and *B* need to transmit their messages to both Nodes *X* and *Y*. Each of the nodes can deliver their own message to the node that is connected to it, but have to route their messages through the network to reach the other node. When making use of traditional routing (Figure 2a), node *C* simply receives and replicate the information it receives from the previous sender node. In this case the two messages *a* and *b* will reach node *C* simultaneously. Node *C* will send out message *a* first, and then message *b*. Thus, at the end of a single arbitrary time unit, only node *Y* will have received both messages, while node *X* still has only message A. This results in a throughput of 1.5.

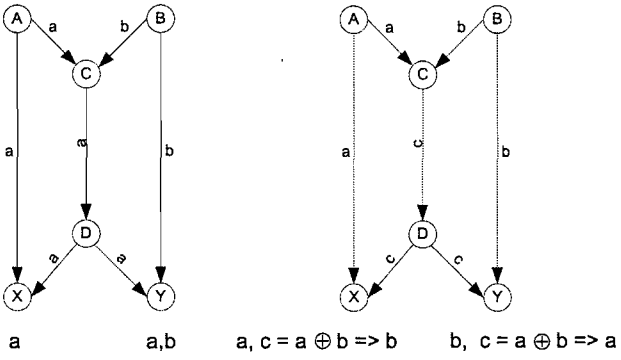


Fig. 2. Butterfly Network a) Without Network Coding b) With Network Coding

When we make use of Network Coding (Figure 2b), we give node *C* the capability to transmit a linear combination (logical XOR) of the binary messages *a* and *b*. Message *c* has the same length as message *a* and *b*, and is transmitted via node *D* to nodes *X* and *Y*. We then give nodes X and Y the capability to decode message *c* by using the other message it already received and solving the two linearly independent equations. In this special case, it merely means adding the single message that the node has already received to the network coded message. This time, by the end of a single arbitrary time unit, both nodes *X* and *Y* have both messages. Two messages were delivered, making the throughput 2. [7]

This method however changes the way node *C* works, because it has to form linear combinations of the messages it receives before forwarding it. It also requires nodes X and Y to have knowledge of the network topology and how the messages reaching it are encoded in order to deduce the two original messages from the messages it received.

An attempt to make Network Coding more practical for MANETs is Random Network Coding [1], but we want to see if we can use a known topology of the deterministic approach in a MANET. Because the deterministic approach to network coding requires little overhead, this approach may be useful to improve network utilization in a MANET.

C. What benefits do we get from Network Coding

The use of Network Coding in a network may provide the following benefits:

1) Throughput [7], [9], [11]:

The improved throughput in networks was the first major result of Network Coding.

If we refer to the throughput achieved with network coding in the deterministic example, we see that we have achieved the maximum throughput as calculated using

the min-cut max-flow theorem.

2) Robustness [9], [11], [12]:

The robustness of the network refers to the ability of the network to remain functioning even though a link has failed completely.

3) Adaptability [11], [13]:

Adaptability is an important benefit when looking at MANETs, as this refers to the ability of the network to cope with nodes constantly joining and leaving the network, resulting in a constantly changing topology.

4) Security [11]:

The security benefit is an inherent benefit, seeing that linear combinations of data are sent over the network, and not the actual data. This benefit while useful, is however not sufficient. If a malicious entity listens long enough and receive sufficient messages to decode the information, the information can still be eavesdropped.

Thus we see that Network Coding can address many of the problems associated with MANETs.

V. CONCEPTUAL NEW IDEA

We found that there is currently no method that indicates where in a MANET or *how* exactly at that specific location deterministic Network Coding could be implemented. This led to our research question: *Can a known topology used in deterministic Network Coding be used to apply Network Coding Practically in MANETs?*

Our hypothesis is:

The unique mobility present in MANETs creates multiple opportunities for opportunistic implementation of deterministic Network Coding in the routing of information.

We propose to examine the connection matrix of a MANET, to see if we can find sub-matrixes that are known Network Coding networks. If we can find these sub-matrixes, we know where in the MANET the opportunity to implement Network Coding exists. Because we use known topologies we then also know how to implement Network Coding, i.e. which nodes should change in their functionality, and which should have information on the network topology. Our aim in this is to improve the local throughput of part of the MANET, by using deterministic Network Coding topologies in a part of the information path.

We foresee the following benefits when using this concept:

1. An improvement in the total throughput (better utilization of the network's capacity)
2. Lower occupation of the total network.
3. Improved quality of service, because of a lower delay in the network.

We propose the following method to illustrate this concept:

i. Select a Network Coding topology of which the gain and capacity is known.
ii. Derive the connection matrix of this Network Coding topology.
iii. Determine the connection matrix of the larger MANET from a suitable distance vector routing algorithm;
iv. Search the larger MANET matrix for the known topology matrix.
v. Implement Network Coding at the appropriate nodes.
vi. Re-iterate steps (iv) and (v) after a routing update.

In order to evaluate whether or not it is worth it to implement deterministic Network Coding in the MANET, we compare the capacity of the MANET before and after the opportunistic implementation of Network Coding.

VI. IMPLEMENTATION OF METHOD

To demonstrate that implementation of this concept is practically possible, we will now use the Butterfly topology as our known topology, and search for it in a random MANET. We do this as follows:

A. Initial steps (Steps i – iii)

We translate the Butterfly network in Figure 2 to a 6x6 connection matrix:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

We then create a connection matrix of a random MANET of 7x7 with a connectivity of at least 3 connections per node, representing the state of the network for one stream of information. We use a 7x7 network, because it is currently the size of the largest physical testing facility in South-Africa (at the MERAKA institute), but we also test our technique in larger networks, to test scalability of the technique. This is sufficient for this implementation, because the size of the random MANET is one order bigger than the matrix that we search for. We scan the large matrix, using the method described in *step iv,* for all permutations of the smaller 6x6 matrix.

Once we have found the smaller matrix-location in the larger matrix, we know (referring to figure 2b) that node C should have encoding capability, thus the node should logically XOR the information received from nodes A and B. Nodes X and Y should have decoding capability, that is, they should derive the two original messages from the information received.

B. Finding the required topology (Step iv)

We generate a random $n \times n$ connection matrix with a connectivity of at least 3. We then want to locate the butterfly topology inherently present in the random matrix. In order to locate the butterfly matrix within the larger network connection matrix several methods can be

used. The simplest of these methods would be iterative looping. The fundamental concept of iterative looping is a nested looping structure. This effectively moves a window across the network connection matrix and continually compares the data in the current window to that of the desired data, in this case the butterfly matrix. Although this method is accurate it is not desired since it scales poorly, and is computationally intensive. For the reasons stated above we propose using a different method.

Our method is based on the two dimensional cross correlation of the network connection matrix and the butterfly matrix. The cross correlation of two datasets can be seen as the similarity of those two datasets, and is commonly used to search for patterns in a random dataset by forming the cross-correlation to known patterns. This technique finds application in digital signal processing (DSP) as well as image processing.

The concept of cross-correlation is extended to two dimensions with the network connection matrix being used as the random dataset and the butterfly matrix being used as the known pattern. The result of the cross-correlation operation *xcor(connection, butterfly)* results in a new matrix M with values indicating the level of cross-correlation. In order to extract the correct values, the auto-correlation of the butterfly matrix is formed. By scanning M for this value the location of the butterfly matrix can be determined.

Advantages of using the cross-correlation method is scalability, as the method remains computationally efficient when using larger matrices. A further advantage is the ability to detect where partial Butterfly structures occur within the network connection matrix. These partial structures then have the potential to become butterflies if the topology of the network changes slightly.

Both methods were implemented in Matlab in order to compare both the accuracy as well as the computational efficiency. For small matrices of less than 40x40 both methods perform equally well in terms of computation time. For larger matrices of 100x100 the cross-correlation method's computation time is significantly less than that of the iterative looping. As the matrix dimensions increase the computational advantages of the cross-correlation method increases.

We can then implement Network Coding at the appropriate nodes. The search is reiterated after possible topology changes. *(Steps v and vi).*

VII. CAPACITY CALCULATIONS

We define capacity of a MANET as: *Capacity is the number of 1-hop sessions that share the same channel. [3, 4]*

In order to test the improvement that our idea offers, we use a technique called MSDA [3, 4]:

MSDA uses the original generated connection matrix as the 1-hop adjacency matrix $A^1_{nxn} = A_{nxn}$. We then define a delay of k and use exact multiplication of A_{nxn} to calculate A^k_{nxn}.

It is then required to calculate all the *k-hop* paths. We do this with the following new algorithm:

1. We look in the upper-triangle of A^k_{nxn} for a value equal to k and store the column numbers of the start and end nodes (as the first and last values) in a vector *Path* of size *k+1* as the source and sink nodes.
2. We move to the row number corresponding to the column where the sink was found and search for the value *k-1*. Ignore the columns of the source and sink as well as the values on the diagonal of the A^k_{nxn} matrix. This value is stored as the second value in the vector *Path.*
3. We repeat the search in the corresponding row of this node's column for the value *k-2*, to fill the second to last element in *Path.*
4. We repeat this process until vector *Path* is filled and then perform a check that the last found node is indeed connected to the first node in *Path.* If it is not connected, the path is deleted.
5. All the generated *Path* vectors are then stored in *PathSet.*

Continuing with MSDA, for every node in *PathSet*, the probability that the node is present in *PathSet* is then calculated, and the sum of the probabilities on a path are then used to calculate the probability of each path.

The path with the lowest probability is then selected and all the paths containing any of the nodes that are present in the selected path are then deleted.

By doing this, we find all the independent *k-hop* paths in A_{nxn}.

This result is used in the MSDA algorithm, after which, we can calculate the capacity of the MANET as in [3]:

$$N_s = \min\left[\left\lfloor \frac{BW_{node}}{BW_{packet}} \right\rfloor, \frac{D_E(n^2 - n - n_0)}{\sum_{i \in A^k_{nxn}} i - n}\right] \quad (1)$$

Where $\lfloor BW_{node} / BW_{packet} \rfloor$ is the number of one-hop sessions that the channel can support, D_E is the end-to end delay, n is the number of nodes and n_0 is the number of 0's in the connection matrix A_{nxn}.

To calculate the benefit of using the Butterfly Network Coding network we then use the location of the Butterfly matrix in the larger matrix, and modify the MSDA algorithm in the following manner:

We ensure that the paths with node combination corresponding to nodes C and D in figure 2 are arranged at the top of *PathSet* to increase the probability that they are selected as a path. We also modify the algorithm not to delete *k-hop* paths with that specific node

combination..

With this modification included, we again calculate the capacity, using (1).

Finally we compare the calculated capacities of the MANET to see the improvement of implementing Network Coding in this way.

VIII. PRACTICAL RESULTS

We implemented our method as described in section IV in networks with 6, 7, 8, 9, 10 and 11 nodes respectively.

We generated 100 random connection matrices for each size network. Our random matrix generator generated 100 unique matrices, checking that a particular connection matrix is not used more than once, but not necessarily that it is not a permutation of a matrix that was already tested.

The generated matrices also had connectivity (C) of exactly 3, meaning that each node was connected to 3 other nodes. The connectivity of three is the minimum connectivity that will allow the possibility of a butterfly network to exist (refer to figure 2), but does not guarantee the existence of a butterfly network in large networks. Results will be better for networks with higher connectivity.

TABLE I
IMPLEMENTATION RESULTS

Nodes:	C:	Networks Searched:	Networks with Butterflies	Total Butterflies
6	3	100	100	1234
7	3	100	100	1608
8	3	100	100	2140
9	3	100	100	893
10	3	100	100	2038
11	3	100	100	1252

The results of this implementation can be seen in table 1.

In all of the networks butterflies were found (being small networks). In most instances far more than one butterfly network was found, although these are not necessarily disjoint or unique butterfly networks.

IX. DISCUSSION OF RESULTS

We find the following four improvements in the network:

i. More available paths:
For each butterfly network present, we have the possibility of one extra path available, sometimes more provided that two butterfly networks do not make use of the same link.

ii. Better utilization of the network
This can be seen as a qualitative improvement.

iii. We can define an upper limit for the network throughput.
The communication flow that passes through the paths making use of Network Coding has an upper limit of the throughput of the network code. In the best case, the paths used in the flow only meet in the Network Coding section and thereafter disjoin, thus the Min Cut of the flow is situated in the Network Coding section. In the worst case, the Min cut occurs outside the Network Coding section and has a lower value.

iv. This method is energy efficient, because fewer transmissions are needed to deliver the same amount of information.

This method proves to be successful, provided that we use a routing protocol that knows what the whole network looks like – such a protocol for MANETs is currently being developed.

The cost of using this method is the following:

i. Clever nodes:
ii. Route establishment delay
iii. A high connectivity is required

These drawbacks are compensated for when the MANET is implemented using Laptops in a close vicinity to each other. Laptops inherently have the high processing power to perform the computations for these "clever nodes" and are able to compensate for the route establishment delay.

X. CONCLUSION

The changing nature of MANETs offers many possibilities for the opportunistic implementation of Network Coding. It is impossible to guarantee a specific increase in capacity due to the constantly varying topology of the MANET. Theoretically this increase in topology can be as high as the maximum throughput capacity, but in practice this is seldom seen.

We have presented in this paper the necessary tools for the network planner to determine whether our Network Coding method can be useful or not for a specific implementation.

REFERENCES

[1] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," In *Proceedings of the IEEE International Symposium on Information Theory* June 2003, page 442, Yokohama, Japan, 2003.

[2] A. Mitrokotsa, N. Komninos, C. Douligeris, (2007), "Intrusion Detection with Neural Networks and Watermarking Techniques for MANET", In *Proceedings of IEEE International Conference on Pervasive Services 2007* (ICPS'07), July 15-20, 2007, pp. 118 - 127 Istanbul, Turkey, 2007.

[3] J. Zhang and W. K. G. Seah, "Topology-based capacity analysis for ad hoc networks with end-to-end delay constraints," In *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication,* 31 May-2 June 2004, vol.2, pp. 541- 544, 2004.

[4] Ning Li, Yan Guo, Shaoren Zheng, Chang Tian, Jun Zheng, "A Matrix-Based Fast Calculation Algorithm for Estimating Network Capacity of MANETs," La Jolla, CA, pp. 407-412, *Systems Communications* (ICW'05, ICHSN'05, ICMCS'05, SENET'05), 2005.

[5] P. Elias, A. Feinstein, and C. E. Shannon, "Note on maximum flow through a network," *IRE Trans. Information Theory*, vol. 2, pp. 117–119, 1956.
[6] S. A. Aly, V. Kapoor, J. Meng and A. Klappenecker, "Bounds on the Network Coding Capacity for Wireless Rondom Networks", *Information Theory and Applications Workshop*, Jan. 29 2007-Feb. 2 2007, pp. 231-236 ,2007.
[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Transactions on Information Theory*, vol. 46, July 2000, pp. 1204-1216, 2000.
[8] R. W. Yeung "Distributed Source Coding for Satellite Communications", *IEEE Transactions on Information Theory*, May 1999, vol. 43, pp. 1111-1120, 1999.
[9] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding", IEEE Transactions on Information Theory", February 2003, vol. 49, pp. 371- 381, 2003.
[10] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright and K. Ramchandran "The Benefits of Network Coding for Peer-to-Peer Storage Systems", *NetCod Workshop*, January 2007.
[11] C. Fragouli, J. Widmer, J. Le Boudec "Network Coding: An instant primer", *ACM SIGCOMM Computer Communication Review*, January 2006, vol.36, pp. 63 - 68 , 2006.
[12] R. Koetter and M. Médard, "Beyond routing: an algebraic approach to network coding", In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol.1 , pp. 122- 130, 2002.
[13] C. Fragouli, J. Widmer, J. Le Boudec "On the Benefits of Network Coding for Wireless Applications", *Netcod*, 2006.

Leenta M. J. Grobler completed her Bachelor of Engineering degree in 2006, in Computer and Electronic engineering at the Northwest University – Potchefstroom Campus.

She is currently a second year Masters Degree student in Computer Engineering for TELKOM COE at the Northwest University, Potchefstroom campus.

Ms. Grobler is registered with ECSA as a candidate engineer and a member of the SAIEE.

Appendix B - Source Code