# Enhancing a network coding security scheme to avoid packet dropping in wireless mesh networks

**HLHC Terblanche**

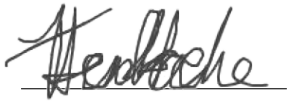
**20569807**

Dissertation submitted in partial fulfillment of the requirements for the degree *Magister Engineering* in Computer and Electronics at the Potchefstroom Campus of the North-West University

Supervisor: Me MJ Grobler

September 2013

I, Heila Levina Helena Catharina Terblanche, hereby declare that the dissertation entitled "Enhancing a network coding security scheme to avoid packet dropping in wireless mesh networks" is my own original work and has not already been submitted to any other university or institution for examination.



H.L.H.C. Terblanche

Student number: 20569807

Signed on the 22nd day of April 2013 at Potchefstroom.

# Acknowledgements

I dedicate this dissertation to all the people who supported and guided me through this time.

I want to thank the Lord Jesus for guiding me through this time and giving me the strength and all that I needed to complete this dissertation.

I want to thank my parents Leon and Heila Terblanche who supported me all these years. Thank you for all the emails, laughs and the cupcakes that kept me going.

I also want to thank my brother Johan who always gave me something to laugh about and my sister Carina for all those late night ice creams.

I want to thank my best friend Joánie Maass for all her friendship and for always motivating me.

To my loving husband Robbie Theron, thank you for always being there and for all your patience and understanding. You mean the world to me.

I want to thank my study leader Mrs. Leenta Grobler without whose guidance, support and reviewing this dissertation would not be possible.

I want to thank the Telenet research group for all the coffee, laughs and support throughout this time. I would also like to thank Mr. Henri Marais and Suné von Solms for all their help with this research.

Finally I want to thank the Telkom Centre of Excellence for funding this research.

# Abstract

With the increase of mobile and smart device usage, the interest in dynamically forming networks is rising. One such type of network is Wireless Mesh Networks (WMNs). WMNs are multi-hop networks, with a decentralised nature that can dynamically form into mesh topologies.

Network Coding (NC) is a method that is used to increase the efficiency of networks by encoding and decoding data on packet level by means of an XOR operation. NC works well with WMNs because it can exploit WMNs broadcast and opportunistic listening properties. When implementing NC on WMNs the issue of security has to be taken into consideration.

Dong *et al.* identified various security threats for intra-flow NC in WMNs. Intra-flow NC combines packets within individual flows, where the information is divided into different flows called generations, to optimize the decoding process.

They identified threats for each component of intra-flow NC for WMNs. These components include forwarding node selection, data packet forwarding and acknowledgement delivery. These threats respectively for each component are wormhole attacks and link quality falsification, packet pollution and packet dropping and acknowledgement-dropping, injection and delay.

We identified that most security schemes focus on packet pollution attacks in NC, but not on any other threats. Packet dropping is also a major threat in networks that is not addressed. Both packet pollution and packet dropping are threats identified for the data forwarding component of WMNs.

The Delayed Authentication with Random Transformations (DART) security scheme addresses packet pollution in intra-flow NC systems. The scheme is based on time asymmetry and checksums. The DART scheme only addresses packet pollution and not any of the other identified threats. The DART scheme was selected to be enhanced to also address packet dropping.

To enhance the DART scheme we added additional information to the DART scheme's checksum packets to detect malicious packet dropping nodes in the network. The information added to the checksum packet took the form of a HealthMatrix, which indicates how many packets a node has received and verified. The new scheme, called the Packet Dropping Detection (PDD) scheme collects the additional information from the checksum packets at the receiver node. The receiver sends the collected information to the source node which then uses the information to identify the malicious nodes in the network. These nodes are then removed from the network.

The results show that this new scheme causes a small decrease in throughput - about 2%. The identification of malicious nodes can be used as a diagnostic tool and faulty nodes can be repaired or removed form the network. The advantage to detect malicious packet dropping nodes far outweighs this decrease in throughput.

In this dissertation we investigate the effects of packet pollution and packet dropping on NC networks in WMNs. We also enhance an already existing scheme (DART) to add additional packet dropping detection security to it without a great loss in throughput.

***Keywords:*** *Network Coding, Packet Dropping, Packet Pollution, Security, Wireless Mesh Networks*

# Opsomming

Die toename in die gebruik van slim toestelle het veroorsaak dat belangstelling toeneem in netwerke wat dinamies vorm. 'n Voorbeeld van so tipe netwerk is draadlose roosternetwerke (WMNs). 'n WMN is 'n draadlose multi-hop netwerk met 'n gedesentraliseerde aard wat dinamies in rooster topologië kan vorm.

Netwerk kodering is 'n metode wat gebruik word om die doeltreffendheid van 'n netwerk te verhoog. Dit geskied deur die enkodering en dekodering van data op pakkie vlak d.m.v. 'n XOR bewerking. Netwerk kodering werk baie goed saam WMNs a.g.v. WMNs se uitsending en opportunistiese luister eienskappe. Wanneer netwerk kodering op WMNs toegepas word, moet die kwessie van sekuriteit ook in ag geneem word.

Dong *et al.* het verskeie bedreigings identifiseer vir intra-vloei Netwerk kodering wat toegepas is op WMNs. Intra-vloei Netwerk kodering kodeer net pakkies van dieselfde vloei saam. Die bedreigings is geïdentifiseer vir elke komponent van intra-vloei Netwerk kodering. Die komponente sluit in die seleksie van aanstuur nodusse, die aanstuur van data pakkies en die aflewering van erkenningspakkies. Die bedreigings wat geïdentifiseer is, is onderskeidelik "wormhole" aanvalle en skalel kwaliteit vervalsing, pakkie besoedeling en pakkie weggooi aanvalle en die weggooi, vertraging en inpsuiting van erkenningspakkies.

Ons het bevind dat meeste sekuriteits skemas op pakkie besoedeling fokus en nie op enige van die ander bedreigings vir Netwerk kodering nie. Die weggooi van pakkies is ook ook 'n groot bedreiging vir netwerke, wat nie aangespreek word nie.

Die DART (Delayed Authentication with Random Transformations) sekuriteits skema spreek pakkie besoedeling aan in intra-vloei netwerk kodering. Die skema is gebaseer op tyd asimmetrie en "checksum" pakkies. Die DART skema spreek net pakkie besoedeling aan en nie die weggooi van van pakkies nie. Ons het veronderstel dat 'n beter sekuriteits skema ontwikkel kon word deur 'n reeds bestaande skema (DART)

wat pakkie besoedeling aanspreek te verbeter deur pakkie weggooi nodusse op te spoor.

Ons het gevind dat deur ekstra inligting by die ”checksum” pakkies van die DART skema by te voeg, ons noddusse wat pakkies weggooi, kon opsoor. Die inligting wat by die ”checksum” pakkies bygevoeg is, is in die vorm van ’n gesondheidsmatriks. Die gesondheidsmatriks dui aan hoeveel pakkies ’n node ontvang en geverifier het. Die ontvanger node kollekteer die ekstra inligting van die ”checksum” pakkies. Die ontvanger node stuur dan al die inligting terug na die sender node wat dit gebruik om die kwaadwillige nodusse in die netwerk op te spoor en te verwyder.

Die resultate wys dat die nuwe voorgestelde skema ’n vermindering van 2% in deurset van die netwerk veroorsaak. Die voordeel om nodusse wat pakkies weggooi te kan opspoor maak op vir die feit dat daar ’n vermindering in die deurset is.

In hierdie proefskrif ondersoek ons die effek van die pakkie besoedeling en die weggooi van pakkies op netwerk gekodeerde netwerke in WMNs. Ons verbeter ook ’n reeds bestaande skema (DART) deur pakkie weggooi nodusse op te spoor en te verwyder sonder ’n groot verlies in die deurset van die netwerk.

# Contents

**List of Figures** **xii**

**List of Tables** **xiv**

**List of Acronyms** **xvi**

**List of Symbols** **xvii**

**1 Introduction** **1**

1.1 Background . . . . . 1

1.2 Motivation . . . . . 4

1.3 Purpose of Research . . . . . 4

1.4 Issues to be addressed . . . . . 5

1.5 Research Methodology . . . . . 5

1.6 Document Structure . . . . . 7

**2 Literature Study** **10**

2.1 Wireless Mesh Networks . . . . . 10

2.1.1 Characteristics . . . . . 10

2.1.2 Application Fields . . . . . 11

2.1.3 Advantages of Wireless Mesh Networks . . . . . 14

2.1.4 Disadvantages of Wireless Mesh Networks . . . 14
2.2 Network Coding . . . 15
2.2.1 How does Network Coding work? . . . 15
2.2.2 Types of Network Coding . . . 15
2.2.3 Application Fields . . . 19
2.2.4 Advantages of Network Coding . . . 19
2.2.5 Disadvantages of Network Coding . . . 20
2.3 Security in networks . . . 20
2.4 Security in Network Coding . . . 21
2.4.1 Intra-flow Network Coding . . . 22
2.5 DART security scheme . . . 26
2.5.1 Description of DART . . . 26
2.6 Simulation Design and Performance Metrics . . . 27
2.6.1 Simulation Design . . . 27
2.6.2 Throughput . . . 28
2.6.3 Latency . . . 28
2.6.4 Malicious node Detection . . . 28

**3 Experimental Design** **30**

3.1 Network Coding . . . 30
3.2 DART Scheme . . . 31
3.2.1 Checksum packet generation . . . 32
3.3 PDD Scheme . . . 35
3.3.1 Basic DART network flow . . . 36
3.3.2 The PDD scheme explained: . . . 37
3.3.3 How each node handles the checksum packet and HealthMatrix 40

3.4 Experimental Set-up . . . . . 46
3.4.1 Simulation Design . . . . . 46
3.5 Experiment Assumptions . . . . . 49
3.6 Simulation Description . . . . . 50
3.6.1 Node Descriptions . . . . . 50
3.6.2 Simulation Set-up . . . . . 51

**4 Verification and Validation** **57**
4.1 Experimental Model Validation . . . . . 57
4.1.1 Normal Source Receiver Network Scenario . . . . . 58
4.1.2 Source Receiver with DART Security Scenario . . . . . 59
4.1.3 Source Receiver with PDD Security Scenario . . . . . 60
4.2 Computerized Model Verification . . . . . 61
4.3 Operational Validation . . . . . 66

**5 Simulation Results** **68**
5.1 Simulation Parameters . . . . . 68
5.2 Throughput of the networks without malicious nodes present . . . . . 70
5.3 Throughput of the Normal network with a malicious node present . . . 71
5.4 Throughput of the DART network with a malicious node present . . . . 73
5.5 Throughput of the PDD network with a malicious node present . . . . . 74
5.6 Comparing the throughput for the Normal, DART and PDD schemes . . 75
5.7 Latency of the networks . . . . . 78
5.8 Detection of packet dropping nodes . . . . . 82

**6 Conclusion** **86**
6.1 Summary of Research . . . . . 86

6.2 Conclusions . . . . . 88
6.3 Future work . . . . . 88

**Bibliography** **89**

**Appendices**

**A Conference Contributions** **93**

# List of Figures


1.1 The Research Methodology . . . . . 8

2.1 Backbone Wireless Mesh Network . . . . . 12
2.2 Client Wireless Mesh Network . . . . . 12
2.3 Hybrid Wireless Mesh Network . . . . . 13
2.4 Butterfly Network without Network Coding . . . . . 16
2.5 Butterfly Network with Network Coding . . . . . 16
2.6 Network Coding Example . . . . . 17
2.7 Network Coding Decode Example . . . . . 18
2.8 Inter-flow Network Coding . . . . . 22
2.9 Intra-flow Network Coding . . . . . 22

3.1 Example of the DART security scheme . . . . . 34
3.2 Example of how the HealthMatrix increments in the network . . . . . 38
3.3 Example1 - The PDD network with its HealthMatrices . . . . . 42
3.4 Example2 - The network with its HealthMatrices at time T . . . . . 45
3.5 Example2 - The network with its HealthMatrices at time 2T . . . . . 45
3.6 Example2 - The network with its HealthMatrices at time 3T . . . . . 46
3.7 The representation of the WMN used in this model. . . . . 47
3.8 The Flow of the Simulation . . . . . 52

4.1 The representation of the network for Example 1. . . . . 58

4.2 The output by MATLAB after decoding . . . . . 62

4.3 The $H_s$ and $CHK_s$ matrices generated by MATLAB for generation G . . . 63

4.4 The result of *RSide* and *LSide* for the code vector c1 and coded packet e1 64

4.5 The result of *RSide* and *LSide* for the code vector $\vec{c1}$ and the invalid coded packet $\vec{e1}$ . . . . . 65

4.6 The output of the simulation where the packet was polluted and where the generation was decoded . . . . . 66

5.1 The Cumulative distribution function (CDF) graph for the throughput of the Normal, DART and PDD schemes . . . . . 70

5.2 The CDF graph for the throughput of the Normal scheme with a malicious node present . . . . . 71

5.3 The CDF graph for the throughput of the DART scheme with a malicious node present . . . . . 73

5.4 The CDF graph for the throughput of the PDD scheme with a malicious node present . . . . . 74

5.5 The CDF graph for the throughput of the Normal, DART and PDD schemes with a malicious pollution node present . . . . . 76

5.6 The CDF graph for the throughput of the Normal, DART and PDD schemes with a malicious packet dropping node present . . . . . 77

5.7 The CDF graph for the latency of the Normal, DART and PDD schemes 79

5.8 The CDF graph for the latency of the Normal, DART and PDD schemes with a malicious packet polluting node present . . . . . 80

5.9 The CDF graph for the latency of the Normal, DART and PDD schemes with a malicious packet dropping node present . . . . . 81

5.10 Example: Explanation of why a false positive is detected. . . . . 83

# List of Tables


3.1 Checksum Packet Descriptions . . . . . 39

3.2 Checksum Packet sent by source . . . . . 40

3.3 Checksum Packet sent by source . . . . . 40

3.4 Checksum Packet sent by source . . . . . 41

4.1 The outputs of the simulated Normal network . . . . . 58

4.2 The outputs of the simulated DART network . . . . . 59

4.3 The outputs of the simulated PDD network . . . . . 60

4.4 Inputs for NC Encoding function test . . . . . 61

4.5 Inputs for function test . . . . . 62

4.6 Inputs for the NC Decoding function test . . . . . 62

4.7 The outputs of the nodes after the source sent 32 packets . . . . . 65

4.8 The outputs of the nodes after the receiver decoded a generation . . . . 67

5.1 Detection Rate of the PDD scheme . . . . . 75

5.2 Throughput of the networks . . . . . 76

5.3 Latency of the networks . . . . . 82

# List of Algorithms

1 Node Placement and network set-up . . . . . 53
2 Node Roles Assignment Algorithm . . . . . 54

# List of Acronyms

**WMN** Wireless Mesh Network

**NC** Network Coding

**RLNC** Random Linear Network Coding

**PDD** Packet Dropping Detection

**DART** Delayed Authentication with Random Transformations

**CDF** Cumulative distribution function

**OSI** Open Systems Interconnection

# List of Symbols

$\mathbf{F_q}$ Finite field

$\mathbf{G}$ Generation

$\tilde{\mathbf{c}}$ Coding vector

$\tilde{\mathbf{e}}$ Coded packet

$\mathbf{s}$ Random seed

$\mathbf{t}$ Timestamp

$\mathbf{d_{max}}$ Maximum connection distance

$\mathbf{numNodes}$ Number of nodes in the network

$\mathbf{V}$ Vertices matrix

$\mathbf{VNC}$ Node coordinates matrix

# Chapter 1

# Introduction

---

*This chapter gives an introduction to this dissertation. Section 1.1 provides background to the research topic. Section 1.2 and 1.3, respectively provides the motivation and problem statement for the research. Section 1.4 describes the issues to be addressed while section 1.5 provides the methodology followed to conduct the research for this dissertation. Finally, section 1.6 gives a brief overview of the document.*

---

## 1.1 Background

Wireless Mesh Networks (WMNs) are wireless multi-hop networks, arranged in a mesh topology, which consist of wireless clients, routers and gateways. They support ad-hoc networking that has self-forming, self-healing and self-organisation properties.

There are three different architectures: backbone or infrastructure mesh networks, client mesh networks and hybrid mesh networks [1]. In this case, we focus on client WMNs.

The advantages of WMNs are [1]:

- Increased network robustness;
- Reliable service coverage;
- Maintainability;
- Low installation costs.

Despite these advantages, the decentralised nature and the openness of the medium, creates a security risk because no authentication is needed and the network is therefore vulnerable to eavesdropping.

Network Coding (NC) is a technique that can improve the efficiency of a network. It is well suited to implementation in WMNs because it exploits the wireless broadcast and opportunistic listening properties of WMNs. This is accomplished by forming linear combinations of the received packets and forwarding the combined packets, thereby minimising transmissions. A coding vector is attached to the new packet specifying which packets were combined to generate the coded packet. These combined packets can easily be decoded at the receiver node. This technique was first proposed in 2000 by Ahlswede *et al.* in [2]. Random Linear Network Coding (RLNC) was first introduced by Ho *et al.* [3], where the elements in the coding vector are randomly chosen from a finite field, and the packets are combined accordingly. In RLNC, the intermediate nodes can decide which packets to combine, using the random coding vectors, before forwarding them. With RLNC, there is no need for a centralised control mechanism and coded packets do not have to arrive in sequence at the receiver.

The advantages of NC include robustness, maximising the throughput, increasing the efficiency and minimising the delay of the network [4,5].

RLNC can easily be implemented in WMNs as shown by Ho *et al.* in [6]. Although using NC with WMNs has many benefits, it also introduces vulnerabilities to the network that has to be addressed.

To ensure that the decoding process is optimised, the information that has to be send is divided into chunks of $n$ packets, called generations. There are two general approaches for NC in WMNs - *intra-flow NC* and *inter-flow NC*. Intra-flow NC combines packets within individual generations and inter-flow NC combines packets across different generations.

In [7], Dong *et al.* analysed the threats for both general approaches to NC. They divided each approach into different components. The components for intra-flow NC were *forwarding node selection and rate assignment*, *data packet forwarding* and *acknowledgement delivery*. The threats identified for the *data packet forwarding* component are packet pollution and packet dropping. We focus on the *data packet forwarding* component of intra-flow NC. For this study, the other identified threats were not taken into account. Thus, we focus on packet pollution and packet dropping. Of the two, packet pollution has the highest impact on the throughput of a network using NC.

Packet pollution occurs when a malicious node injects corrupt packets into the network. Packet pollution attacks can cause a significant decrease in the throughput of the network. Because packets are combined, the pollution can spread quickly through the network. The Delayed Authentication with Random Transformations (DART) scheme proposed by Dong in [8] addresses packet pollution and does not incur as much overhead as other schemes. Packets can be dropped by a malicious node or the packets can be lost due to channel errors. Packet dropping attacks where the malicious node drops all the packets received are known as black hole attacks. With grey hole attacks, packets are dropped at random intervals. Packet dropping attacks are not as severe as packet pollution but still has a negative impact on the throughput of the network.

The DART security scheme was proposed in 2009 by Dong *et al.* in [8]. This scheme is based on time asymmetry and checksums and addresses the packet pollution threat. When implementing the DART security scheme the source divides the data into generations. Coded packets are generated from the active generation and send into the network. The source also generates checksum packets, at constant time intervals, for the active generation and broadcasts it to all the forwarder nodes.

The coded packets that arrive at the forwarder and receiver nodes are stored in an unverified queue. These packets are then verified upon reception of a checksum packet, and only packets that arrived at the node before the checksum was created are verified. Verified packets are forwarded. If a malicious node injects a polluted packet into the network, it will not propagate further than one hop, because of the verification of packets at each node. The DART security scheme only addresses packet pollution and not packet dropping.

## 1.2 Motivation

There are numerous schemes that were proposed for security purposes like [6,9,10] but by adding the additional security overhead the advantage gained by NC is cancelled out. Most of these schemes focus on packet pollution and do not take any of the other identified security vulnerabilities into account. Thus by taking an existing scheme, that does not add as much overhead as other schemes, and expanding it by adding more security features, a scheme that addresses more security threats can be developed. In this instance, the DART security scheme can be expanded by adding features that can identify packet dropping nodes and exclude them from the network.

## 1.3 Purpose of Research

Existing security schemes for NC only address singular threats. The goal of this research is to determine whether an existing security scheme that already addresses packet pollution, can be expanded to address an additional security threat namely packet dropping.

## 1.4 Issues to be addressed

The main objective of the research, is to determine whether the existing DART security scheme can be expanded to address more NC security threats. To achieve this the following issues will be addressed:

- The implementation of a WMN in Matlab®.
- To investigate the effects of packet pollution and packet dropping on networks.
- To implement the DART security scheme proposed in [8].
- To expand the scheme to include packet dropping detection.

## 1.5 Research Methodology

The scientific method was followed in order to complete this research. This method is described by Gauch in [11]. Our interpretation of this method is described in Fig. 1.1.

**Study a Field**
A literature survey was conducted in the chosen research field. This was done in section 1.1.

**Define a Research Problem**
A research problem was identified in section 1.3 while the issues that were to be addressed was described in section 1.4.

**Study Literature**
An in-depth literature study was performed in Chapter 2. The research topics included background on WMNs, NC, security in NC, the DART security scheme and performance metrics.

**Construct Hypothesis**

A hypothesis is defined by the Oxford dictionary as

> *'a supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation.'*

We hypothesised that the DART security scheme could be expanded by using a simple packet acknowledgement scheme to detect packet dropping.

**Test through Simulation**

The experiment was designed in Chapter 3. In this experiment, deductive logic was used. Deductive logic starts at the general principles of the model and derives the data while inductive logic starts with the data and derives the general principles of the model from it.

**Validation and Verification**

Computerised Model Verification is defined by Sargent [12] as

> *'assuring that the computer programming and implementation of the conceptual model is correct.'*

The primary techniques used to verify the computerised model is structured walk-throughs and traces. For this research, each of the core functions used in the simulation were tested separately to ensure correctness. Traces were used to ensure the correct flow of the data in the simulation.

Operational Validity is defined by Sargent [12] as

> *'determining that the model's output behaviour has sufficient accuracy for the model's intended purpose over the domain of the model's intended applicability.'*

We interpret this as seeing whether the research question is answered. In this case,

> Does the method add additional security to the DART security scheme?

Additionally the performance metrics for the different networks (with and without security) were compared to see whether they fall within reasonable bounds.

The results and simulation were verified and validated in Chapter 4.

**Analysis of Results**
After validation and verification, the simulation was implemented and results obtained. These results were presented and analysed in Chapter 5.

**Conclusion**
After the results had been verified and validated, conclusions were drawn in Chapter 6.

## 1.6 Document Structure

The remainder of this document is structured as follows:

In Chapter 2, an in depth literature study is done on the relevant research topics. These topics include WMN, NC and security in NC along with their advantages and disadvantages. The chapter also discusses the DART security scheme along with the performance metrics used in the simulation. In Chapter 3, the experimental design is described including the model, assumptions and the simulation. In Chapter 4, the validation and verification methods are discussed. The simulation and operational models are also validated and verified. Chapter 5 discusses the simulation results. The document concludes with Chapter 6 in which conclusions are drawn about the results along with recommendations for future work.

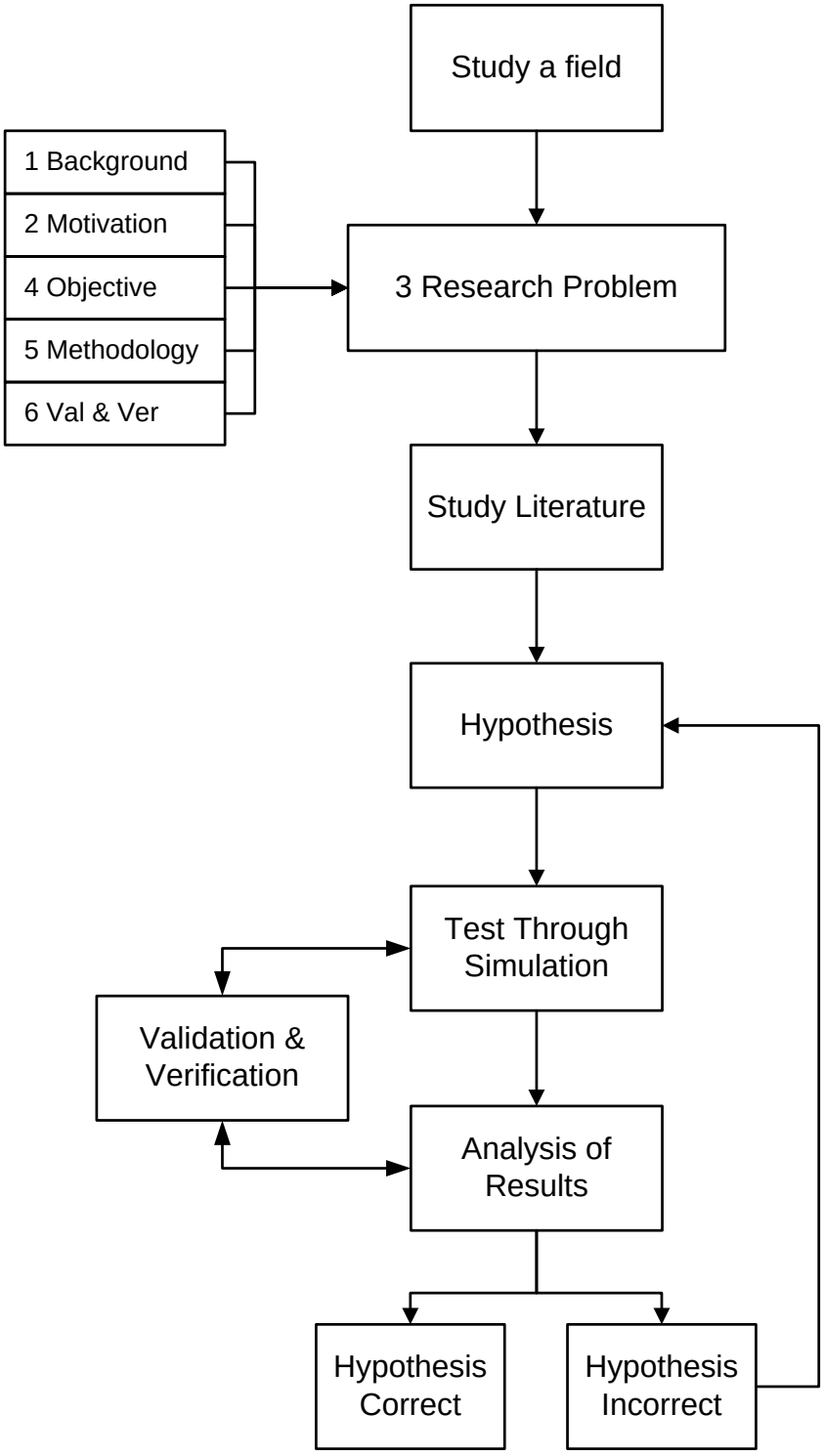



Figure 1.1: The Research Methodology

*In this chapter, an introduction to the dissertation was given. A brief background in section 1.1 along with a motivation for the research in section 1.2 was given. We described the purpose of the research along with the issues that were addressed, in sections 1.3 and 1.4. A description of the research method called the scientific method followed and finally the document structure was laid out in section 1.6.*

# Chapter 2

# Literature Study

---

*In this chapter background is given on WMNs, NC, security in networks, security in NC, the DART security scheme and the performance metrics used.*

---

## 2.1 Wireless Mesh Networks

### 2.1.1 Characteristics

WMNs are multi-hop wireless networks that are a special case of ad-hoc networks. They support ad-hoc networking with self-healing, self-organising, self-forming properties.

They consist of wireless clients, routers and gateways that form into a mesh topology. They can be managed from a central location or a decentralised location. WMNs can easily integrate into other networks like the internet, wireless sensor networks or wired networks.

There are three different architectures: backbone or infrastructure mesh networks, client mesh networks and hybrid mesh networks [1].

- **Backbone or Infrastructure WMNs** are the most common. Mesh routers form a backbone for mesh clients to connect to, and the routers can connect to the internet if it doubles as a gateway as shown in Fig. 2.1.
- **Client WMNs** consist only of wireless clients that communicate with each other as shown in Fig. 2.2.
- **Hybrid WMNs** are a combination of client and backbone mesh networks as shown in Fig. 2.3.

### 2.1.2 Application Fields

WMNs can be applied to numerous fields including [1] :

- **Military applications** - WMNs can be deployed in the field of battle, allowing for easier and faster communication between personnel.
- **Remote monitoring** - WMNs can be deployed in cities to allow for remote monitoring of traffic lights, railways, electricity etc.
- **Community Networks** - WMNs can be deployed in communities such as universities and neighbourhoods, allowing shared internet access without a physical connection or satellite uplink.
- **Emergency Response** - Hospitals can use WMNs to respond quicker to emergencies or it can allow for quick access to patient charts via the network.

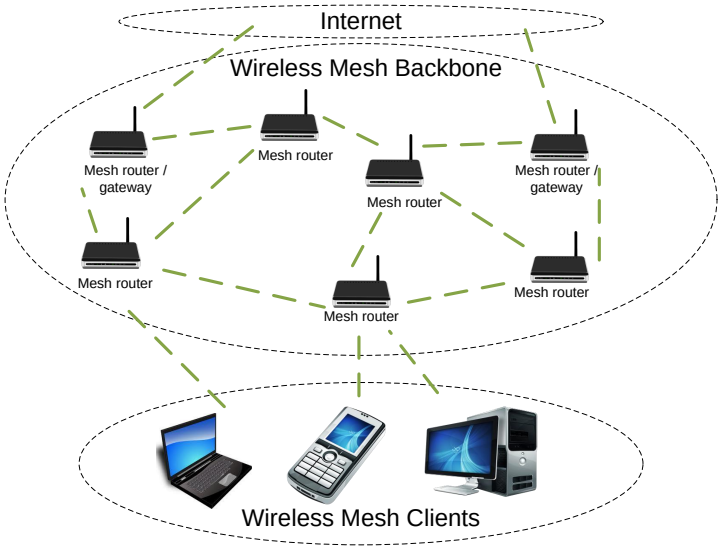



Figure 2.1: Backbone Wireless Mesh Network

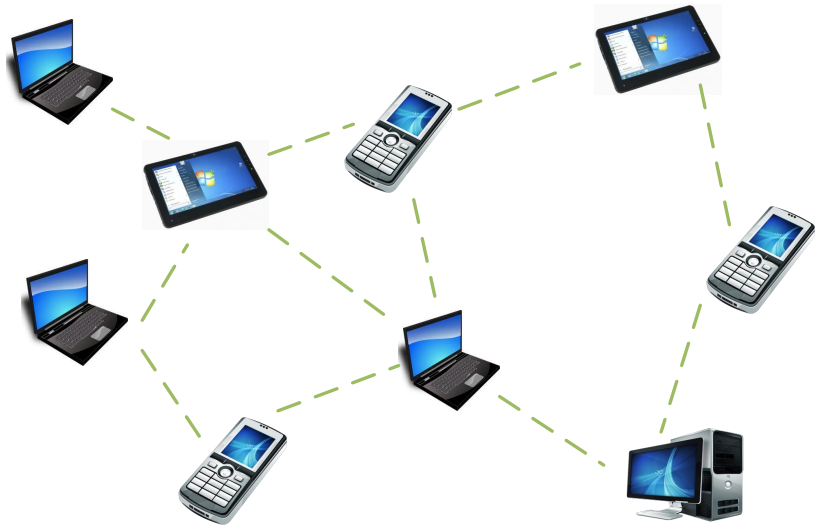


Figure 2.2: Client Wireless Mesh Network

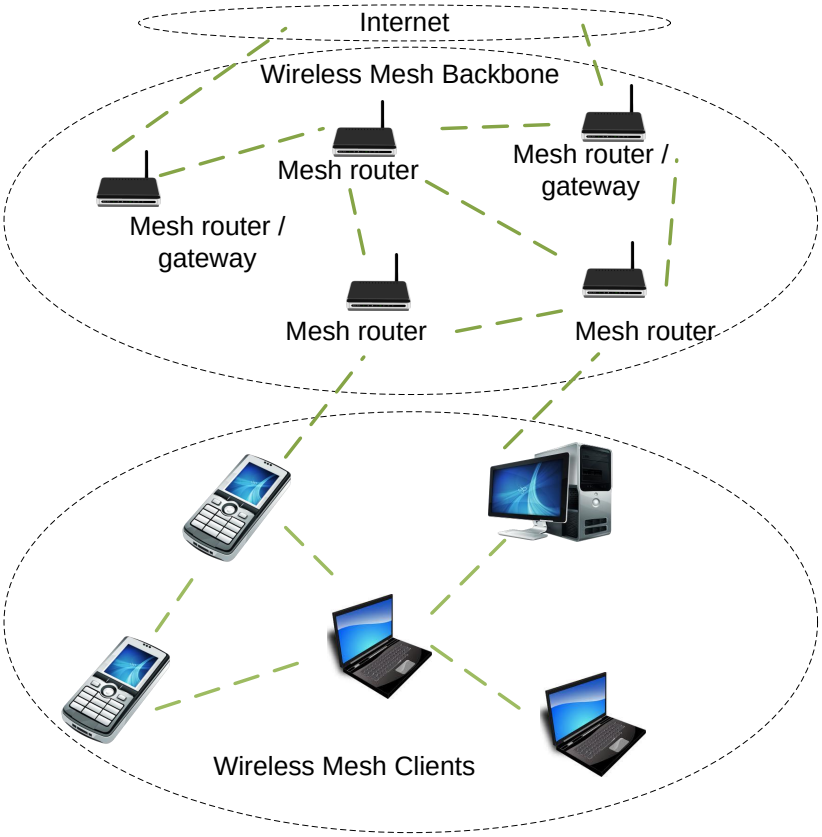



Figure 2.3: Hybrid Wireless Mesh Network

### 2.1.3 Advantages of Wireless Mesh Networks

WMNs have the following benefits [1,13] :

- **Network robustness** - The multiple path nature of WMNs can overcome link failures. When a link fails another path can be selected to route information because there is more than one path to a node.
- **Reliable service coverage** - The multiple hops of WMNs provide redundant paths that expand the coverage area.
- **Easy maintenance** : WMNs can be deployed relatively fast in comparison to normal fixed lines. They can also be extended incrementally and are easily assembled and disassembled.
- **Low installation costs** - WMNs have very low upfront installation costs as there is no costly infrastructure that has to be implemented to set up the network.

### 2.1.4 Disadvantages of Wireless Mesh Networks

Despite all the benefits, WMNs still have their own kind of challenges namely:

- **Routing** - Most of the routing protocols employed in WMNs are not developed for WMNs. Existing protocols are either extremely complex or too simple.
- **Network management** - The decentralised nature of WMNs can make managing the network more difficult.
- **Delay** - The multiple hop nature of WMNs can increase the delay of packet delivery.
- **Security issue** - The decentralised nature and the openness of the medium creates a security risk, because of no authentication in the network. Also, multiple hops can put user data at risk while it is travelling through the network.

## 2.2 Network Coding

NC was first introduced in [2] by Ahlswede *et al*. NC can be described as a technique that optimises the throughput of a network by combining the received packets and sending these combined packets on to the next node. It provides several benefits and is well suited for implementation in the wireless environment because it takes advantage of the wireless broadcast and opportunistic listening properties.

### 2.2.1 How does Network Coding work?

NC can easily be explained by the butterfly network model in Fig.2.4 and Fig.2.5. Regard both nodes A and B as source nodes and nodes E and F as receiver nodes. If both source nodes have a packet they want to send to both receiver nodes, a bottleneck is created at node C. The receiver node E can receive both packets, where packet $p_1$ will travel along the edge{AE} and the packet $p_2$ will travel along the edges {BC, CD, DE}. Similarly, node F can also receive both packets. Assuming that a node can only send one packet at a time node C will first send packet $p_1$ and at the next timeslot will send packet $p_2$. This bottleneck can be alleviated by implementing NC at node C. This is achieved by combining both packets with a *XOR* operation. Thus, a new packet $e_1 = p_1 \oplus p_2$ is created. Node C then forwards the combined packet and that packet is then decoded at the receiver nodes using the messages already received.

### 2.2.2 Types of Network Coding

There are two types of NC, namely Deterministic NC and RLNC which will be discussed below.

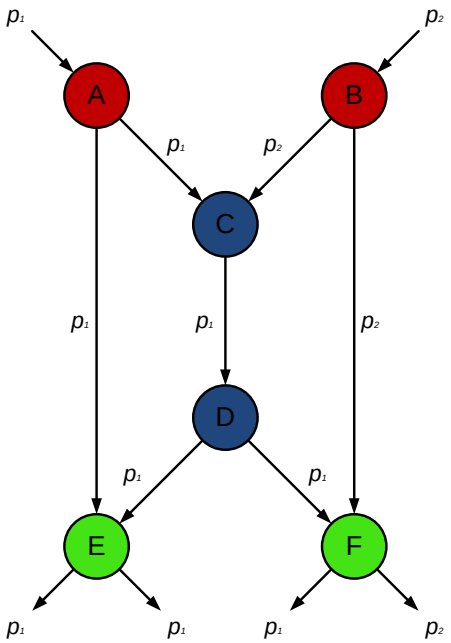



Figure 2.4: Butterfly Network without Network Coding

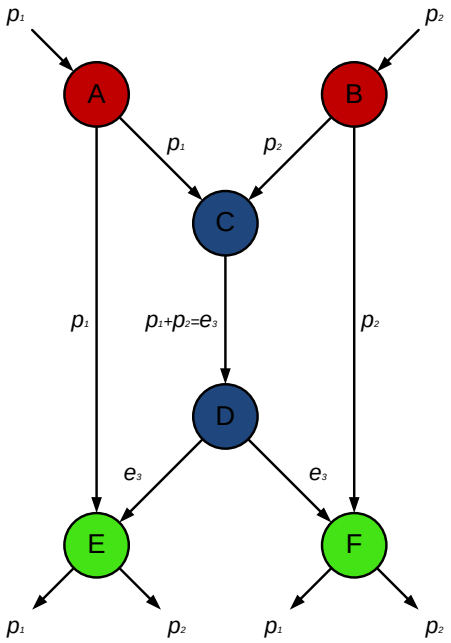



Figure 2.5: Butterfly Network with Network Coding

### Deterministic Network Coding

In deterministic NC, an algebraic approach is taken to the combining of packets. The packets are combined with a bitwise XOR. The process is the same as described in section 2.2.1.

Each packet $p_i$ consists of $m$ codewords, with m an element of a finite field. The size of $m$ is determined by the finite field used for coding. In the case of $\mathbb{F}_{2^8}$ the size of $m$ is one byte. Thus, a packet can be described as a column vector with $m$ symbols:

$$\vec{p_i} = (p_{i1}, p_{i2}, ..., p_{im})^\top \in \mathbb{F}_q \tag{2.1}$$

where q is a positive power of a prime number.

When packets are combined, a new coded packet $\vec{e}$ is created,

$$\vec{e_1} = \vec{p_1} \oplus \vec{p_2} \in \mathbb{F}_q \tag{2.2}$$

with all operations taking place in the finite field, an example of this can be seen in Fig. 2.6. It is necessary to remember that when packets are combined, their length stays the same. With this type of NC, it is necessary for the nodes to know the network topology. The topology is used to determine how the packets were combined, thus knowing how to decode them.

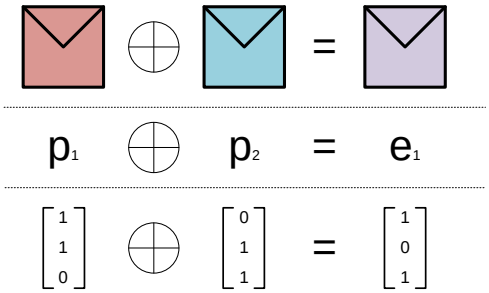



Figure 2.6: Network Coding Example

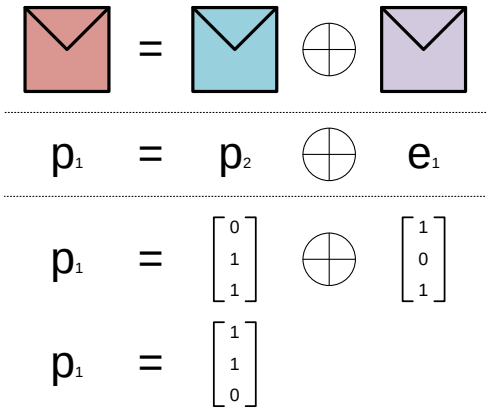



Figure 2.7: Network Coding Decode Example

To decode the packets, Gaussian elimination is used. As seen in the case discussed in Fig.2.5 at node E the $p_2$ packet can be computed by $p_2 = p_1 \oplus e_1$ as seen in Fig. 2.7.

**Random Linear Network Coding**

In practical WMNs packets are send randomly, because the nodes are not always aware of the topology. RLNC works well with decentralised operations. RLNC was first introduced by Ho *etal*. in [3].

The information that has to be send is divided into batches called generations to make the encoding and decoding process simpler. When the nodes performing NC creates new coded packets, they only use packets from the same generation. When a receiver has received enough linearly independent packets of a generation, it can decode. A generation $G$ can be described as a *(m x n)* matrix

$$G = [\vec{p_1}, \vec{p_2}, ...\vec{p_n}], p_{ij} \in \mathbb{F}_q \tag{2.3}$$

where $n$ is the size of a generation, the packets $p_i$ are column vectors as described in equation 2.2 and $q$ represents the number of elements in the finite field.

In RLNC, a coding vector $\vec{c}$ is added to the coded packet to ensure correct decoding at the receiver. The elements in the coding vector are chosen randomly from a finite

field. The finite field $\mathbb{F}_{2^8}$ consists of the elements 0 to 255. It has been proven in [3] that if the field is sufficiently large the coded packet will be linearly independent from the other coded and normal packets. The coding vector $\vec{c}$ has a size of $n$, and is described as follows:

$$\vec{c} = c_1, c_2, ...c_n \in \mathbb{F}_q \tag{2.4}$$

where $c_i$ is an element from the finite field $\mathbb{F}_q$.

After the elements were selected, the packets are combined according to the coding vector. The new coded packet $\vec{e}$ can be described as,

$$\vec{e} = \sum_{i=1}^{n} c_i \vec{p}_i, i = 1, 2, ..., n \tag{2.5}$$

The packet that is forwarded consists of the coding vector and the coded packet $(\vec{c}, \vec{e})$. Packets do not have to arrive in sequence because they are coded. When the receiver has enough linearly independent packets of the current generation, in this case $n$ packets, it decodes them by solving the set of $n$ linear equations.

### 2.2.3 Application Fields

NC can be used in [4] distributed storage, peer-to-peer networks, file sharing, WMNs, and ad-hoc sensor networks.

### 2.2.4 Advantages of Network Coding

The benefits of NC are :

- **Improving throughput** - NC can reduce the effects of a bottleneck in the network. [2, 4, 5].
- **Minimises the energy per bit** - It takes less energy to send through the same amount of packets in the network. [4, 5]

- **Minimises the delay** - Packets move faster through the network. [4]
- **Improves network robustness and adaptability** - All coded packets are equal, and the receiver node only has receive enough linearly independent packets (no matter which) to decode. [5]

### 2.2.5 Disadvantages of Network Coding

The combining nature of NC makes it vulnerable to attacks. When a packet is polluted with incorrect information, a whole generation of packets can be lost, because the decoding process will give incorrect answers. These threats to NC will be discussed in section 2.4.

## 2.3 Security in networks

According to [14] there are five security services for network security. They are message-confidentiality, integrity, authentication and non-repudiation and entity authentication.

**Message confidentiality/privacy** - means that the data that is sent over the network needs to be confidential.

**Message integrity** - means that the data that is sent over the network must be valid and not have been tampered with.

**Message authentication** - means that the receiver must be sure of the sender's identity.

**Message non-repudiation** - means that the sender and receiver cannot deny sending or receiving the message.

**Entity authentication** - is when the user or entity's identity is verified.

## 2.4 Security in Network Coding

Secure NC was introduced in 2002 by Cai *etal*. in [15]. According to [16] there are three approaches to security namely computational-, physical- and information-theoretic security.

- **Computational** - when it is computationally infeasible to break the system.
- **Physical** - when using physical properties to prevent or detect attacks.
- **Information Theoretic** - determines the maximum transmission rate necessary to make it impossible to break the system.

There are a lot of approaches to security in NC. The solutions for security are focused on only three of the five security services namely message confidentiality and message integrity in [9,16] and message authentication in [16].

Most of the solutions focus on eavesdropping attacks and Byzantine (malicious node) attacks. The security schemes that focus on eavesdroppers include, [17] which uses secret sharing to combat it, [9] that uses secure key checksums and [10] that uses secret key distribution. The security schemes that focus on Byzantine attacks aka malicious nodes in the network, include, [18–20] who uses distributed signature schemes and [21] that uses homomorphic Message Authentication Codes.

In NC, the information that is send over the network is divided into chunks called generations. Usually each generation consists of 32 packets and generations are numbered or marked.

Security in NC has two general frameworks, inter-flow NC and intra-flow NC. Inter-flow NC is when packets are combined over different generations. For example in Fig. 2.8 the coded packet $E1$ consists of linear combinations of packets $a$, $e$ and $h$ which come from generations 1, 2 and 3 respectively. Thus NC was performed over multiple generations. Intra-flow NC mixes packets within individual flows or generations as

seen in Fig. 2.9. The packets $E1$ and $E2$ consists only of packets from generation 1 and packet $E3$ consists only of packets from generation 2.

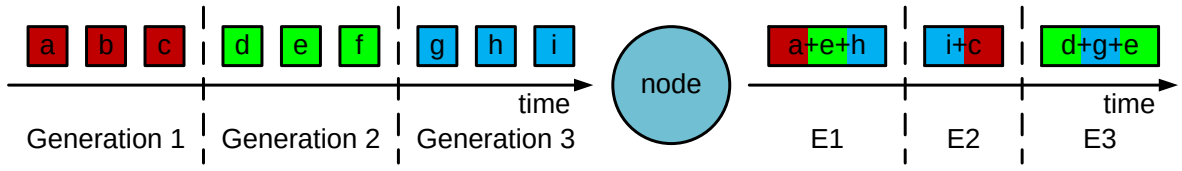



Figure 2.8: Inter-flow Network Coding

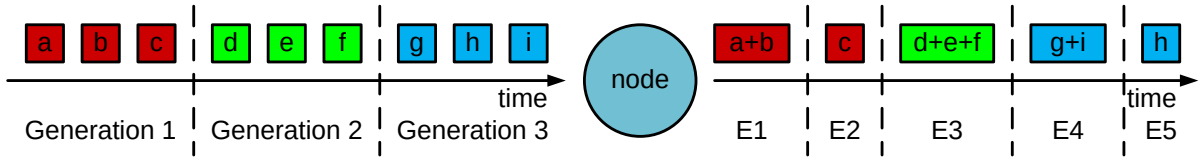



Figure 2.9: Intra-flow Network Coding

### 2.4.1 Intra-flow Network Coding

With intra-flow NC packets are combined within individual flows while with inter-flow NC packets are combined across multiple flows.

For security purposes, it is better to implement intra-flow NC rather than inter-flow NC. In 2009 [7] analysed NC on WMN and identified various security threats for both inter-flow and intra-flow NC. They divided the system into its various components namely, forwarding node selection, data packet forwarding and acknowledgement delivery.

The threats identified were:

**Forwarding node selection and rate assignment:**

- Link quality falsification or modification;
- Wormholes.

**For data packet forwarding:**

- Packet pollution;
- Packet dropping.

**For acknowledgement delivery:**

- ACK injection or modification;
- ACK dropping;
- ACK delay.

In the case of NC, the data packet forwarding component is very important. As can be seen form the list there are numerous threats and the solutions described in section 2.4 only focussed on the packet pollution threat. Packet dropping is also a significant threat and therefore needs to be addressed. If there are malicious nodes in the network that drops packets, the receiving node cannot decode the packets. This causes an increase in the delay of the network because some of the packets then has to be send again.

**Packet Pollution**

Packet pollution occurs when a malicious node injects corrupt packets into the network. This is done by either fabricating a packet, or by changing the contents of an

existing packet in the network. When a NC node receives a corrupted packet and it is used in creating a new coded packet, the corruption spreads quickly through the network. At the receiver node, a whole generation of packets can be corrupted, because of one corrupt packet. This can cause loss of data, a delay in the network and a significant decrease in the throughput of the network. Packet pollution can also occur accidentally when the packets get corrupt because of channel errors.

The packet pollution security threat has been addressed by other security schemes [6, 21, 22].

These schemes all incur high overhead and are computationally intensive. The DART scheme proposed by [8] addresses packet pollution but does not incur as much overhead as other schemes.

**Packet Dropping**

Packet dropping occurs when a malicious node in the network deliberately drops received packets. There are two types of packet dropping attacks, black hole attacks and grey hole attacks. Packet dropping attacks where the malicious node drops all the packets received are known as black hole attacks. With grey hole attacks, packets are dropped selectively at random intervals. Grey hole attacks are usually more difficult to detect than black hole attacks.

NC has inherent resilience to packet dropping, but because NC system are optimised very carefully, any interference can cause problems. It was shown by [7] that packet dropping can have a severe effect on NC systems.

Packet dropping attacks are not as severe as packet pollution but still has a negative effect on the throughput of the network.

There are three approaches to packet dropping in WMN, reputation-based, credit based and acknowledgement based.

**Reputation based [23]**

- Identifies packet dropping on a per node basis.
- Collects accurate observations of node behaviour.
- Computes a reputation metric.
- Challenges:
  - Consumes a lot of bandwidth with the collection of information.
  - The algorithm must be correct.
  - It is easier detect black hole attacks, than grey hole attacks.

**Acknowledgement based [24]**

- Downstream nodes send acknowledgements upstream.
- Suitable for unicast traffic.
- Challenges:
  - The packet dropping node can still send authentic acknowledgements.
  - Cannot identify selective broadcast dropping nodes.

**Credit based [25]**

- Creates an incentive for selfish nodes to forward packets.
- When it forwards, the credit counter increments and uses credit to send its own packets.
- Challenges:
  - Selective dropping attacks are difficult to detect.
  - Malicious nodes can still collect enough credit(grey hole attacks).
  - No identification mechanism to detect malicious node.

## 2.5 DART security scheme

The DART security scheme was proposed by [8]. The scheme focuses on preventing packet pollution in WMNs using intra-flow NC. It is based on signature schemes and time asymmetry. Dong *et al*. [8] claims that their scheme have a much smaller overhead than other security schemes for packet pollution. The same type of scheme was used for NC in peer-to-peer systems in [26]. It is based on TESLA [27] that also used signature schemes and time asymmetry.

### 2.5.1 Description of DART

The DART security scheme uses RLNC and the source divides the data into generations. Coded packets are generated from the active generation and send into the network.

The source generates random checksum packets, at constant time intervals, for the active generation, $G$, and broadcasts it to all the intermediate nodes. The checksum packet $(CHK_s(G), s, t)$ consists of the random checksum $CHK_s(G)$ for the packets in the generation $G$, the random seed $s$, used to create the checksum and $t$ the timestamp when the checksum was created. For authentication the source digitally signs the packet.

The intermediate nodes in the network each have two packet buffers, the verified_set and the unverified_set. These two buffers store verified packets and unverified packets respectively. Only packets that were verified are used for NC and forwarded. When a new coded packet arrives it is put into the unverified_set along with its arrival time. When a checksum packet arrives its digital signature is checked. If it is authentic, the checksum packet is forwarded to the next nodes. Then all the unverified packets that arrived before the checksum packet was created are verified. All the packets that pass verification are put into the verified_set and packets that failed are discarded.

Packets that arrive at the receiver node also go through the same process as the forwarder nodes but the verified packets are stored in a decoding matrix. If the receiver node has enough linearly independent packets of the current generation, it decodes the generation and uses an end-to-end authentication scheme to verify them. The authentication scheme is for extra security in case a polluted packet slipped through the checksum verification.

If a malicious node injects a polluted packet into the network, it will not propagate further than one hop, because of the verification of packets at each node. If an attacker can produce a polluted packet that meets the current checksum's requirements, the packet still will not be verified because the node only verifies packets that were received before the checksum was created. The polluted packet will thus be verified by the next different random checksum and be discarded.

The DART security scheme is not as computationally expensive as other schemes and is practically implemented. The scheme only addresses the packet pollution threat, but has the potential to be expanded to address more security issues like packet dropping.

## 2.6 Simulation Design and Performance Metrics

### 2.6.1 Simulation Design

To simulate a network, a pre-existing simulator can be used or a custom simulator can be written from scratch. Popular network simulators include OPNET, OMNET++, QualNet, NS-2 and NS-3. Any computer language can be used to write a custom simulator including C++, Matlab, Java and

In this implementation we assumed that the timing of the simulation was in the order of nanoseconds implicating that the transfer of packets between nodes are instantaneous.

In order to determine the effectiveness of any implemented scheme, metrics are needed to objectively measure the performance of the scheme.

### 2.6.2 Throughput

Throughput is defined as the amount of data that is successfully transferred from one place to another in a certain amount of time.

Throughput measured at the receiver node in a network using NC, is defined as the number of relevant coded packets that is received in the time it takes to send all the generations.

### 2.6.3 Latency

The latency of the network is the time it takes for a packet to arrive at the receiver node from the source node. In the case with NC the latency is measured as the time it takes for the first decoded packet to arrive at the receiver. This occurs when the first generation is successfully decoded.

### 2.6.4 Malicious node Detection

The probability of detecting the right malicious node is measured by checking if each identified malicious node corresponds to the actual malicious node.

*This chapter provides background literature to the problem of the research topic. Section 2.1 provided background to WMNs which are wireless multi-hop networks that are arranged in a mesh topology. There are three types, Backbone WMNs, Client WMNs and Hybrid WMNs. We focus on Client WMNs that consists on of wireless clients that communicate with each other.*

*Section 2.2 describes NC, a technique that can be implemented in WMNs. NC is a technique that can improve the efficiency of a network. This is accomplished by forming linear combinations of the received packets and forwarding the combined packets, thereby minimising transmissions.*

*There are two general frameworks for NC, inter-flow NC and intra-flow NC as discussed in section 2.4. The threats identified for the data packet forwarding component of intra-flow NC are packet pollution and packet dropping. Packet pollution occurs when a malicious node injects corrupt packets into the network while packet dropping happens when a malicious node drops the received packets.*

*Section 2.5 describes the DART security scheme that addresses packet pollution. This scheme is based on time asymmetry and checksums. It does not incur as much overhead as other security schemes, but does not address packet dropping in a network.*

# Chapter 3

# Experimental Design

---

*In this chapter we describe the design of the experiment. A brief background and description of the schemes is given in sections 3.1, 3.2 and 3.3. The experimental set-up, the assumptions made and the parameters used in the simulation are also discussed.*

---

## 3.1 Network Coding

NC is a technique that can be implemented in a network to increase its efficiency and maximize the throughput. This is achieved by combining packets at the intermediate nodes and forwarding these combined packets. This technique is described in Chapter 2. The NC implemented in this experiment is RLNC. RLNC is a type of NC where the intermediate nodes randomly code packets together, and not in a predefined manner as with Deterministic NC.

The source node divides the data in chunks of $n$ packets, called generations, where there are 32 packets in each generation. A coding vector with the length of 32 elements is generated and the packets in the current generation are combined accordingly to

create a coded packet. These elements are chosen at random from the finite field $F_{2^8}$. The source starts sending these coded packets to the intermediate nodes.

All intermediate nodes in the network are RLNC nodes. Each intermediate node stores the received packets in a buffer called the incoming queue. When it is the intermediate node's turn to send a packet, a new coded packet is generated by combining the packets in the incoming queue as explained in Chapter 2. When there is only one packet in the incoming queue that packet is forwarded. The packets in the incoming queue are periodically flushed to ensure that new coded packets are created. Each time the intermediate node sends a packet, half of the same generation packets in the incoming queue are flushed.

The receiver node decodes a generation when it has received 32 linearly independent coded packets, by solving the linear equation, and sends an acknowledgement to the source node.

In this implementation the generations are pipelined. This means that the source sends $n$ packets of a generation and then moves on to the next generation without waiting for an acknowledgement from the receiver. An active window of $k$ generations is maintained and the source cycles through these generations. When a generation is acknowledged the next generation is activated.

## 3.2 DART Scheme

The DART security scheme by [8] was implemented in a NC network that uses generations and RLNC. The source divides the data into generations consisting of 32 packets. The DART scheme is placed on top of a RLNC network. Basically the DART scheme creates extra checksum packets that are used to verify each coded packet at the intermediate nodes before the packets are forwarded.

The simulations that were done in [8] the assumption was made that the source node

knew which nodes were on the forwarding path. This assumption stemmed from the routing protocol MORE [28] that was implemented in their simulations.

The source node generates an additional checksum packet $(CHK_s(G), s, t)$, that consists of the random checksum $CHK_s(G)$ for the packets in the generation $G$, the random seed $s$, which is used to create the checksum and $t$ the timestamp when the checksum was created. For authentication the source digitally signs the packet. Intermediate nodes store received packets in an unverified queue called the unverified_set and the unverified packets are verified when the node receives a checksum packet. The packets that are verified are stored in a verified queue called the verified_set.

### 3.2.1 Checksum packet generation

In [8], a pseudo random function

$$f : \{0,1\}^{\kappa} \times \{0,1\}^{log_2(b)+log_2(m)} \rightarrow \mathbb{F}_q \tag{3.1}$$

is defined, with $\kappa$, which is the size of the key for $f$ and $b$ which controls the size of the checksum and $m$ the number of symbols in a packet. They let $f_s(x)$ to denote the $f$ keyed with key s applied on input $x$.

The source generates a random $b \times m$ matrix $H_s = [u_{i,j}]$ using the function $f$ and a random $\kappa$-bit seed $s$, where $u_{i,j} = f_s(i||j)$. In this implementation the built-in random number generator of MATLAB was used to create the $b \times m$ checksum matrix.

A checksum $CHK_s(G)$, for a generation $G$ with seed $s$ is defined as

$$CHK_s(G) = H_s G \tag{3.2}$$

which is a $b \times n$ matrix because $H_s$ is a $b \times m$ matrix and $G$ is a $m \times n$ matrix. After the checksum is calculated, the source distributes the checksum packet, $(CHK_s(G), s, t)$ to the forwarder nodes in the network. The checksum packet is send after a generation has been sent into the network. When a node receives a valid checksum packet for a

generation $G$, it verifies the coded packets received before the time $t$, when the checksum was created. The validity of a packet $(\vec{c}, \vec{e})$ is checked by determining whether the following equation holds,

$$CHK_s(G)\vec{c} = H_s\vec{e} \tag{3.3}$$

where $H_s$ is the $b \times m$ matrix generated by the node with seed $s$. This can be done, because the checksum matrix does a random linear transformation on the generation.

Consider a valid packet $(\vec{c}, \vec{e})$ where $\vec{e} = \sum_{i=1}^{n} c_i\vec{p}_i = G\vec{c}$ and a checksum packet $(CHK_s(G), s, t)$ where $CHK_s(G) = H_sG$. Then, $CHK_s(G)\vec{c} = (H_sG)\vec{c} = H_s(G\vec{c}) = H_s\vec{e}$, proving the correctness of the validity equation.

When a data packet is received, it goes into the unverified_set and after the packets are validated they are moved to the verified_set. When there are packets in the verified_set, they are either network coded or they are just forwarded to the next node. This ensures that no polluted packets gets forwarded more than one hop. When a node receives a checksum packet it first checks to see if it is not a duplicate checksum packet. If it is an original packet the checksum packet is used to verify the packets, otherwise the checksum packet is discarded. When generations are pipelined a checksum packet contains a checksum matrix for every active generation. Every packet that is part of the active generations is verified.

The security of the scheme is proved in the paper [8]. They also state that choosing the finite field as $\mathbb{F}_{2^8}$ and the security parameter $b = 2$ is sufficient to contain packet pollution.

An example of the DART scheme can be seen in Fig. 3.1. In this example consider a network consisting of a source, receiver and four intermediate nodes. The receiver and intermediate nodes have two incoming buffers, the unverified queue and the verified queue. The intermediate nodes perform RLNC. The source sends one generation at a time and disseminates checksum packets at every time interval T.

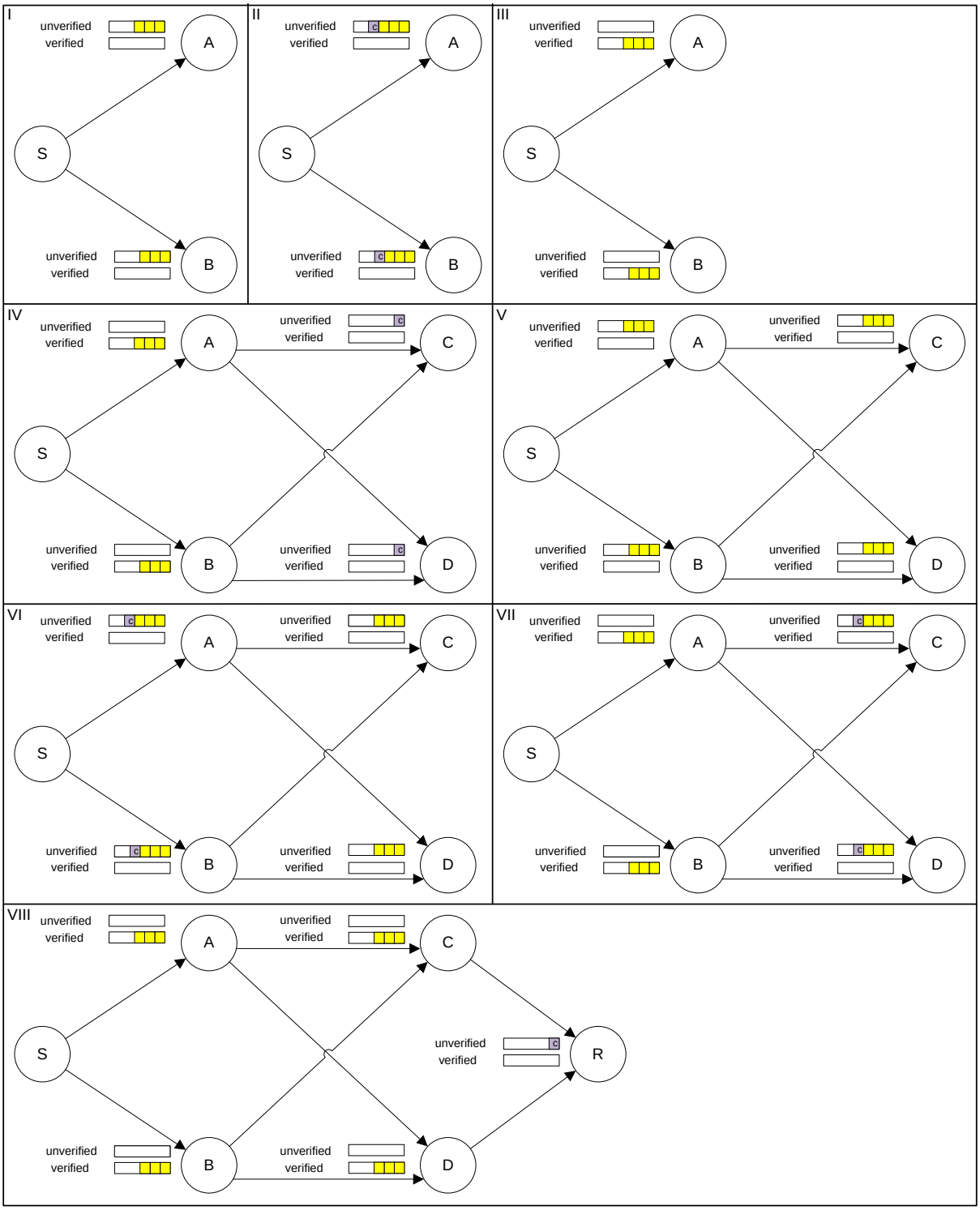



Figure 3.1: Example of the DART security scheme

In Frame I the source $S$ sends out packets to $A$ and $B$. These packets are stored in node $A's$ and node $B's$ unverified queues and each packet's arrival time is also recorded. In Frame II the source sends out a checksum packet at time $T$ that arrives at $A$ and $B$. $A$ and $B$ immediately forwards the checksum packet. $A$ and $B$ uses the checksum packet to verify all the packets that were received before time $T$. When the packets are verified, they are stored in each node's verified queue as seen in Frame III. In Frame IV the checksum packets arrived at nodes $C$ and $D$. Since $C$ and $D$ have no unverified packets they just forward the checksum packet. In Frame V the source continues to send coded packets to $A$ and $B$ and $A$ and $B$ also forward new coded packets to $C$ and $D$. In Frame VI the source sends another different checksum packet at time $2T$. Nodes $A$ and $B$ forward the checksum packet and uses it to verify all the packets in their unverified queues that were received before time $2T$. In Frame VII the checksum arrives at nodes $C$ and $D$. They forward the checksum packet and then uses it to verify the packets in their unverified queues that arrived before time $2T$. Finally in Frame VIII the checksum packet arrives at the receiver node $R$. As there are no packets to verify it discards the checksum packet. Nodes $C$ and $D$ can now forward packet to the receiver and when the next checksum is send into the network the receiver will use it to verify the packets in its unverified queue. This process continues until there are enough packets at the receiver node $R$. When there are enough linearly independent verified packets at the receiver, they are decoded and the generation is acknowledged. When the generation is acknowledged the source moves on to the next generation. The entire process repeats until all the generations have reached the receiver node.

## 3.3 Packet Dropping Detection (PDD) Scheme

The DART scheme as described in section 3.2 addresses packet pollution in NC networks. This scheme cannot detect malicious packet dropping nodes that have a negative effect on the throughput of the network. These nodes can also be compromised nodes that pose a security risk to the network. When packet dropping nodes are detected they can be removed form the network or an alert can be generated to inspect

the node manually to identify any problems with it.

### 3.3.1 Basic DART network flow

The basic network flow is described in the following paragraphs.

The source multi-casts 32 coded packets into the network. After that it broadcasts a random checksum packet to all the nodes it is connected to. The source sends multiple generations into the network. It cycles through the active generations until a generation is acknowledged and then activates the next generation. This process is known as pipelining generations.

Each intermediate node keeps a buffer in which it stores the last 50 checksum packets received. This buffer is used to check for duplicate checksum packets. When a duplicate checksum packet is received it is discarded. Duplicate checksums can be discarded because the original checksum checked all packets that were viable. If it is not a duplicate checksum packet, it is transferred to the outbound checksum queue. When it is the node's turn to send packets it first checks if there are any checksum packets and if there are, the checksum packets are send before the coded packets.

The receiver node is essentially an intermediate node that does not forward the verified packets but decodes them. After a generation is decoded the receiver sends an acknowledgement packet to the source. When the source receives the acknowledgement packet it moves on to the next active generation

The checksum packet essentially travels through the whole network. We saw that by adding additional information to the checksum packet the downstream nodes could gather information about their upstream nodes.

### 3.3.2 The PDD scheme explained:

The PDD scheme can be classified as a type of reputation based approach to packet dropping as discussed in section 2.4.1. We saw that by adding additional information to the checksum packet the downstream nodes could gather information about their upstream nodes. Each node in the network tracks how many packets of each generation it has received. It also keeps track of how many of these received packets were verified. They are stored in a matrix that we denote as the HealthMatrix. The HealthMatrix of a node initially consists of a $(1\ x\ 3)$ matrix where the first column represents the $NodeID$. Each node in the network has a unique $NodeID$. The second column represents all the packets that were inserted into the unverified queue while the second column represents all the packets that were successfully verified. Each time a packet is received or verified by a node the appropriate element in the node's matrix is incremented. This can be seen in Fig. 3.2 where the source multicasts 32 coded packets and then broadcasts a checksum packet.

This matrix is expanded as more generations travel through the node resulting in a $(1\ x\ 3\ x\ n)$ matrix where $n$ represents the number of generations. An example of how the HealthMatrix expands can be seen in equations 3.4, 3.5 and 3.6.

$$HealthMatrix_{1,1-3,1} = \begin{pmatrix} NodeID & UnverifiedPackets(Gen_1) & VerifiedPackets(Gen_1) \end{pmatrix} \tag{3.4}$$

$$HealthMatrix_{1,1-3,2} = \begin{pmatrix} NodeID & UnverifiedPackets(Gen_2) & VerifiedPackets(Gen_2) \end{pmatrix} \tag{3.5}$$

$$HealthMatrix_{1,1-3,n} = \begin{pmatrix} NodeID & UnverifiedPackets(Gen_n) & VerifiedPackets(Gen_n) \end{pmatrix} \tag{3.6}$$

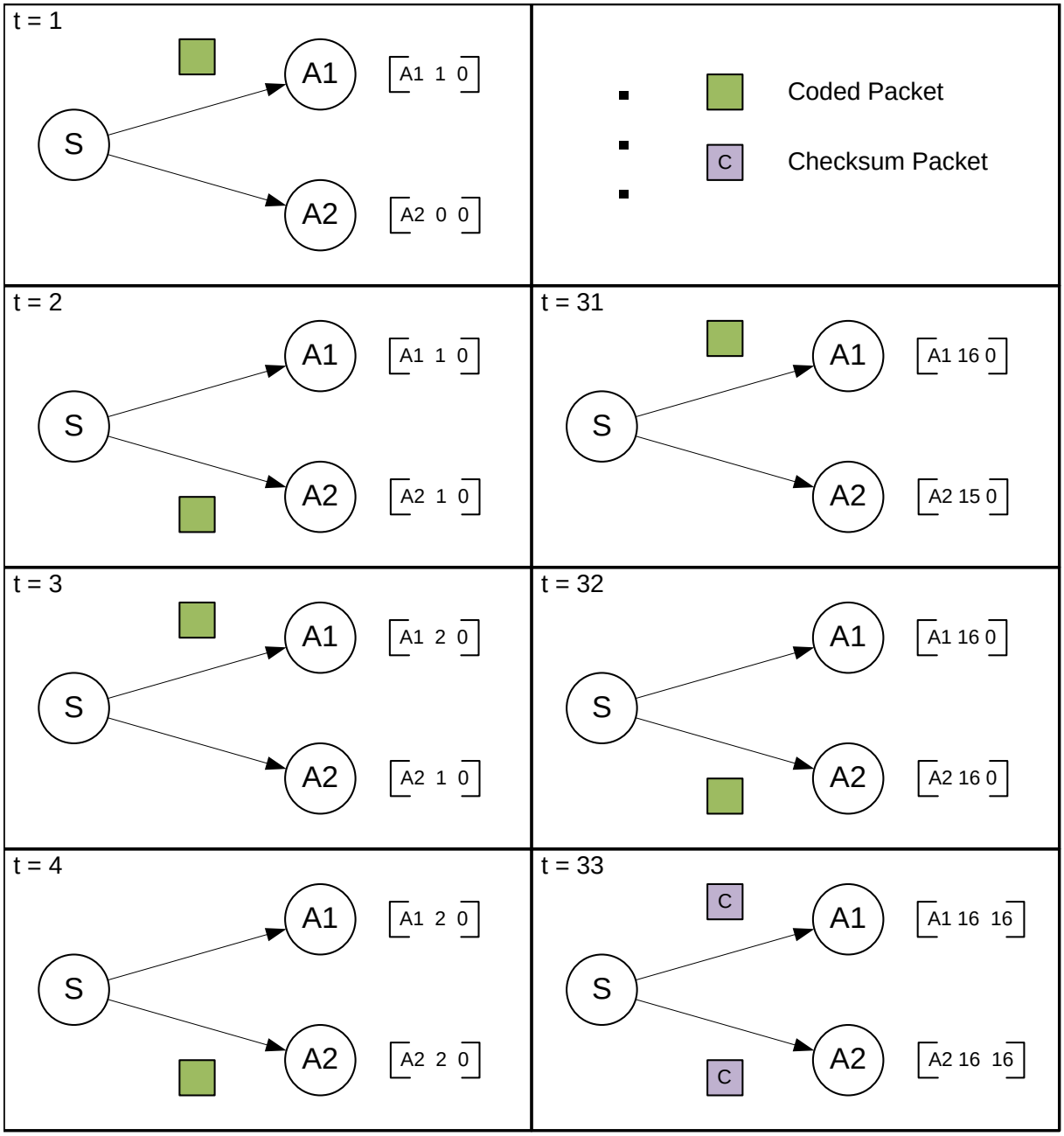



Figure 3.2: Example of how the HealthMatrix increments in the network

When the source initially sends out a checksum packet, an empty $(1\ x\ 3)$ HealthMatrix is attached to the packet as a place holder. The headers for the checksum packet for the DART scheme as used in the implementation is shown in Table 3.1 as well as the additional header for the PDD scheme.

Table 3.1: Checksum Packet Descriptions

| **Field** | **Description** |
|---|---|
| **Security Checksum Packet - DART** | |
| Type | 2 |
| GenID | 1 - n |
| CHK_s | checksum matrix |
| Seed | random seed used to generate checksum matrix |
| Time | time the packet was created |
| | |
| **Security Checksum Packet - PDD** | |
| Type | 2 |
| GenID | 1 - n |
| CHK_s | checksum matrix |
| Seed | random seed used to generate checksum matrix |
| Time | time the packet was created |
| HealthMatrix | matrix containing information about the packets the nodes in the network received |

When the checksum arrives at an intermediate or receiver node it updates that node's HealthMatrix. Only entries that were larger in the checksum's HealthMatrix are updated. The intermediate node then processes the checksum packet as in the DART scheme. The only exception is that before the intermediate node forwards the checksum packet, it updates the HealthMatrix in the checksum packet with the node's HealthMatrix. When an intermediate node receives a duplicate checksum packet, it first updates the node's HealthMatrix before it is discarded. When the checksum packet eventually reaches the receiver, the receiver updates its HealthMatrix. When the receiver acknowledges a generation it also sends its HealthMatrix to the source. When the source node receives the updated HealthMatrix from the receiver node, it checks the HealthMatrix for any intermediate nodes that did not receive any packets or entries that are missing. The source node is aware of the topology of the network and can thus keep a HealthMatrix that consists of all the nodes. As a packet dropping node never sends any packets (including checksum packets), its corresponding entry in the source's HealthMatrix will be $(NodeID, 0, 0)$ indicating that it is a packet dropping node. With this information the source can detect the packet dropping node and

remove it from the forwarder set of nodes. This detection scheme can be used to do identify faulty nodes in the network.

### 3.3.3 How each node handles the checksum packet and HealthMatrix

The source node generates a checksum packet with an empty HealthMatrix as shown in Table 3.2.

Table 3.2: Checksum Packet sent by source

| **FieldName** | **Description** |
|---|---|
| **PDD Checksum Packet** | |
| Type | 2 |
| GenID | 1 |
| CHK_s | (2 x 32) matrix |
| Seed | 45648 |
| Time | 33 |
| HealthMatrix | (" 0 0) |

When an intermediate node $A1$ receives the checksum packet, it is checked for duplicity. The HealthMatrix of the checksum packet is checked for information. As the HealthMatrix is empty in this case the node moves on to verifying the packets in the unverified queue. Then the checksum packet is stored until it is the node's turn to send. Supposing the nodes HealthMatrix consists of $(A1\ 20\ 16)$, before the node sends the checksum packet the HealthMatrix is updated as shown in Table 3.3

Table 3.3: Checksum Packet sent by source

| **Field** | **Description** |
|---|---|
| **PDD Checksum Packet** | |
| Type | 2 |
| GenID | 1 |
| CHK_s | (2 x 32) matrix |
| Seed | 45648 |
| Time | 37 |
| HealthMatrix | (A1 20 16) |

When an intermediate node $A2$ receives the checksum packet, it is checked for duplicity. The HealthMatrix of the checksum packet is checked for information. As the HealthMatrix has an entry for $A1$ that is not in node A2's HealthMatrix, the node's HealthMatrix is updated. The checksum is then used for verifying the packets in the unverified queue. Then the checksum packet is stored until it is the node's turn to send. Say the node's HealthMatrix consists of the matrix in equation 3.7.

$$\begin{pmatrix} A1 & 20 & 16 \\ A2 & 12 & 12 \end{pmatrix} \tag{3.7}$$

Before the node sends the checksum packet the HealthMatrix is updated as shown in Table 3.4

Table 3.4: Checksum Packet sent by source

| **Field** | **Description** |
|---|---|
| **PDD Checksum Packet** | |
| Type | 2 |
| GenID | 1 |
| CHK_s | (2 x 32) matrix |
| Seed | 45648 |
| Time | 37 |
| HealthMatrix | (A1 20 16)<br>(A2 12 12) |

The receiver node handles the HealthMatrix in the same way as intermediate nodes. When the receiver acknowledges a generation it sends it's HealthMatrix to the source node.

**Example 1** :

In this example a brief overview of how the system works is given for a network without any malicious nodes present. Consider the network in Fig. 3.3 consisting of a source node $S$, a receiver node $R$ and five intermediate nodes $A1 - A5$. Each node gets a turn to send. Let $t$ denote the time-step and one packet takes one unit of time to travel between nodes. At time $t = 0$ the source starts multi-casting packets to $A1$ and $A2$ as seen in Frame I. After the source has multi-cast 32 packets it broadcasts the checksum

to $A1$ and $A2$ as seen in Frame II. The checksum packet has an empty HealthMatrix attached to it by the source node. The intermediate nodes $A1$ and $A2$ uses the checksum packet to verify all qualifying packets in their unverified queues.

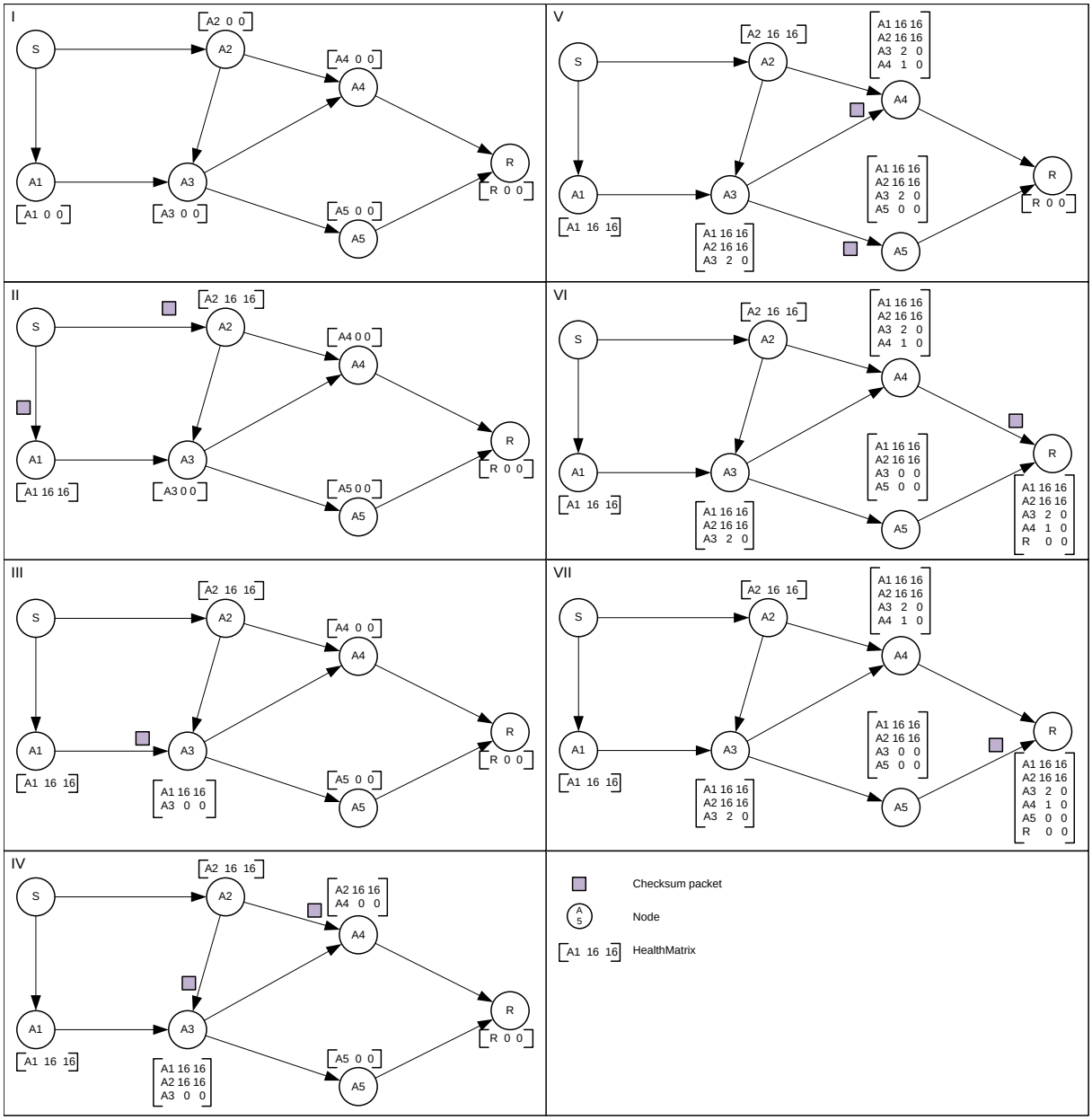



Figure 3.3: Example1 - The PDD network with its HealthMatrices

When it is node $A1$'s turn to send ($t = 34$), it updates the checksum packet with its HealthMatrix. The node then sends the checksum packet to node $A3$ as well as a new

coded packet as seen in Frame III. When the checksum packet arrives at node $A3$, the node checks if the packet is a duplicate and then it updates its HealthMatrix from the checksum. The node expands its HealthMatrix to include an entry for node $A1$. The coded packet that was received is placed in the unverified queue of the node.

When it is node $A2$'s turn to send ($t = 36$) it updates the checksum packet with its HealthMatrix. The node then sends the checksum packet to nodes $A3$ and $A4$ as well as a new coded packet as seen in Frame IV. When node $A3$ receives the checksum packet it checks if it is a duplicate packet. If it is a duplicate packet it the node updates its HealthMatrix before the checksum packet is discarded. The coded packet the node received is placed in the unverified queue and the arrival time is recorded. When node $A4$ receives the checksum packet it is checked for duplicity and the node's HealthMatrix is updated. As there are no packets in nodes $A3$ and $A4$ that were received before the checksum was created, no packets were verified. At this stage node $A3$'d HealthMatrix has entries for itself as well as nodes $A1$ and node $A2$. Node $A4$ will have entries in its HealthMatrix for itself and node $A2$.

At time $t = 38$ node $A3$ updates the checksum packet's HealthMatrix and sends the checksum packet to node $A4$ and $A5$ as seen in Frame V. As there are no packets in node $A3$'s verified queue no coded packet is send. When node $A4$ receives the checksum packet it is checked for duplicity and the node's HealthMatrix is updated. As the entries in the HealthMatrix for node $A2$ will be the same, they are not updated but the entries for nodes $A1$ and $A3$ will be updated. When node A5 receives the checksum packet it checks for duplicity and updates the node's HealthMatrix.

At time $t = 39$ it is node $A4$'s turn to send. It updates the checksum packet and send it to node $R$ as seen in Frame VI. As there are no packets in the verified queue no coded packet is send. When node $R$ receives the checksum packet it checks for duplicity and updates the node's HealthMatrix. The HealthMatrix of node $R$ now has entries for itself, node $A1$, node $A2$, node $A3$ and node $A4$.

When it is node $A5$'s turn to send ($t = 40$) it updates the checksum packet and sends it to node $R$ as seen in Frame VII. As there are no packets in the verified queue no coded packet is send. When node $R$ receives the checksum packet is checked for duplicity. The node's HealthMatrix is updated and the checksum packet discarded. As the entries in the checksum's HealthMatrix for nodes $A1$, $A2$ and $A3$ are the same as the entries in the node's HealthMatrix, they are not updated. Only an entry for node $A5$ is added to the node's HealthMatrix. Thus node $R$'s HealthMatrix contains information about all the nodes in the network except the source node. This process repeats until node $R$ collects enough linearly independent verified packets to decode a generation. When the generation is decoded node $R$ sends an acknowledgement packet to the source $R$ along with the node's updated HealthMatrix. When the source $S$ receives the HealthMatrix, the matrix is checked for missing entries and entries that are zero. As the source knows the topology of the network and the identity of the forwarder nodes it is easy to identify packet dropping nodes as they would not have any entries in the HealthMatrix.

**Example 2** :
In this example a brief overview of the PDD scheme is given for a network with a malicious packet dropping node present. Consider the network is Fig. 3.4. Let node $D1$ be a malicious packet dropping node. On time 0 the source $S$ starts sending a packets to $D1$ and $A2$. The intermediate node $A2$ stores its received packets in its unverified queue and records the amount of packets in its HealthMatrix. Node $D1$ drops all the packets it receives. At time $T$ the source broadcasts a checksum packet. Node $D1$ discards its checksum packet while node $A2$ uses the checksum to verify its packets in the unverified queue. Node $A2$ updates the checksum packet with its HealthMatrix and forwards it to nodes $B1$ and $B2$. When the checksum arrives at nodes $B1$ and $B2$ they update their HealthMatrices. When the receiver receives the checksum packet the node also updates it's HealthMatrix. The HealthMatrices for the nodes can be seen in Fig 3.4.

This process repeats and the HealthMatrices for the nodes at time $2T$ is shown in Fig. 3.5. It can be seen that the entries for $B1$ and $B2$ has been updated.

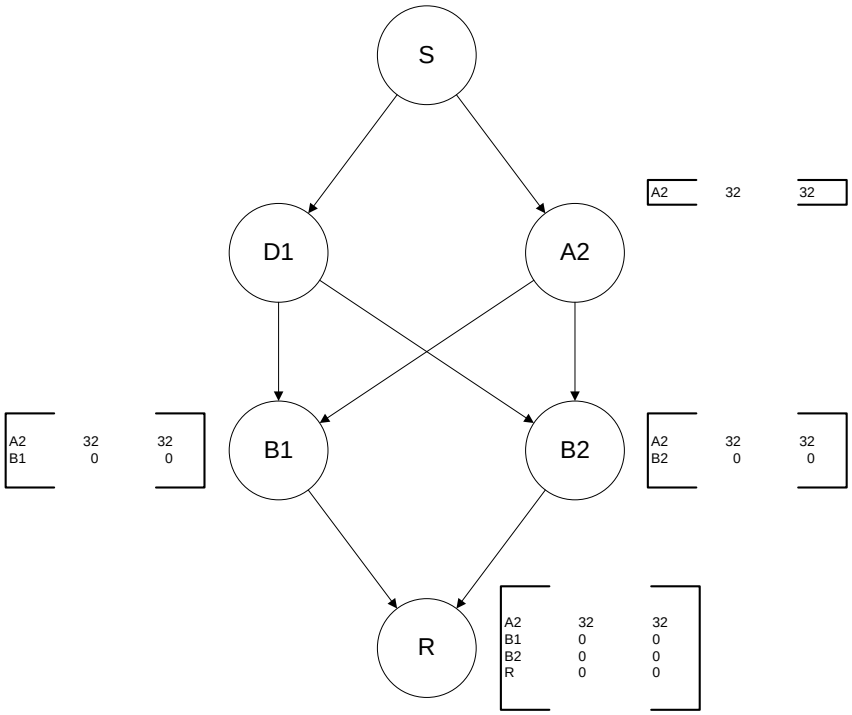



Figure 3.4: Example2 - The network with its HealthMatrices at time T

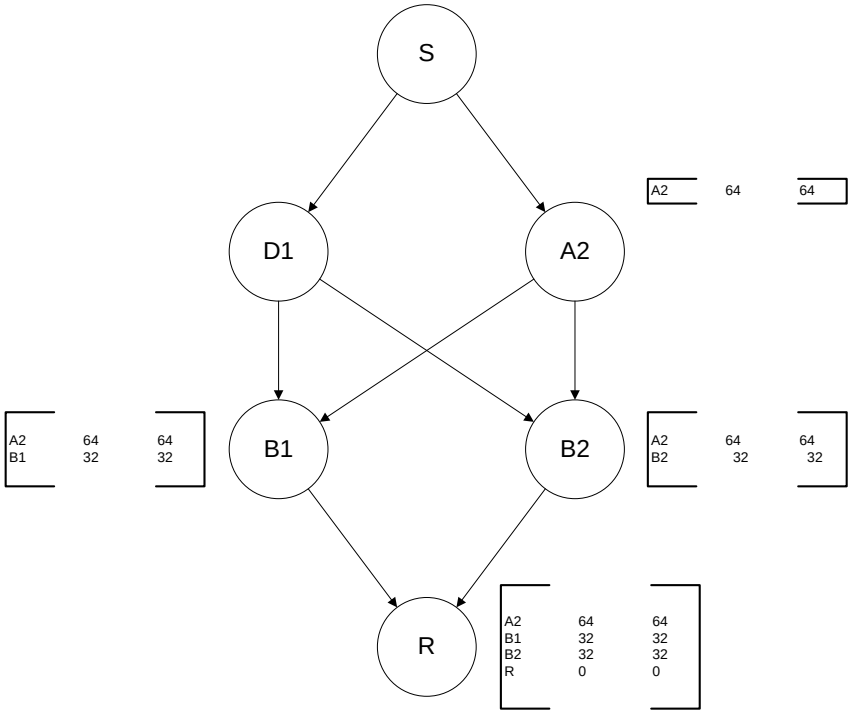



Figure 3.5: Example2 - The network with its HealthMatrices at time 2T

The process is repeated a third time and the HealthMatrices for time $3T$ is shown in Fig. 3.6. In this case the generation is decoded and the HealthMatrix at the receiver is send to the source. The source examines the received HealthMatrix. As there is no entry for $D1$ and the source knows that there should be it indicates that $D1$ is a packet dropping node. The source knows the topology of the network and thus knows which nodes entries should be in the HealthMatrix.

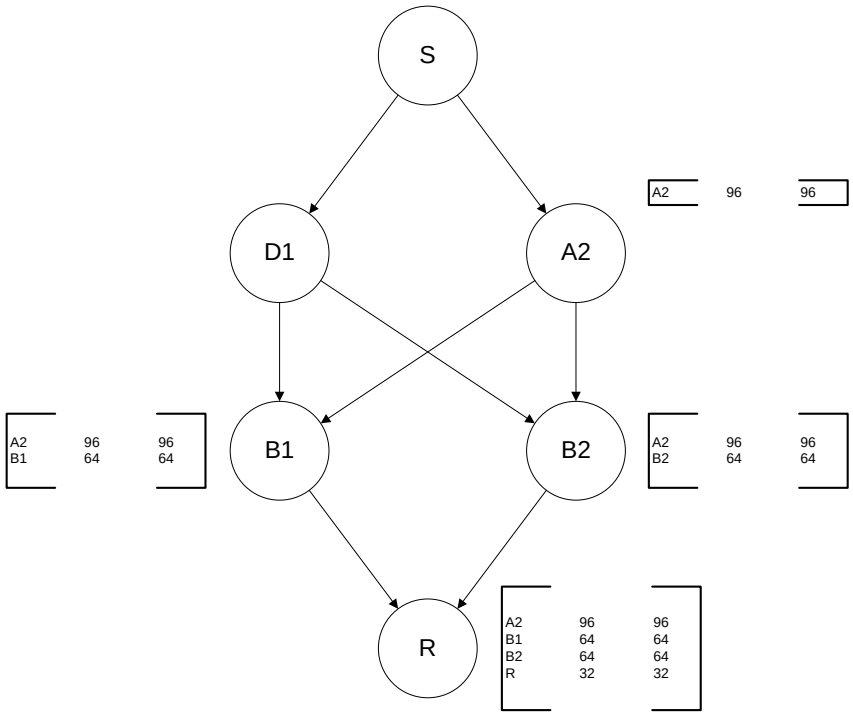



Figure 3.6: Example2 - The network with its HealthMatrices at time 3T

## 3.4 Experimental Set-up

### 3.4.1 Simulation Design

In this implementation we assumed that the timing of the simulation was in the order of nanoseconds implicating that the transfer of packets between nodes were instantaneous. Thus packets were moved from the outgoing queue of a node directly into the

incoming queue of the other node.

As stated in Chapter 1, WMNs consist of nodes that form a mesh topology. This can be seen in the example WMN that was used as representation network in Fig. 3.7. The green nodes in the figure represents the source and receiver nodes.

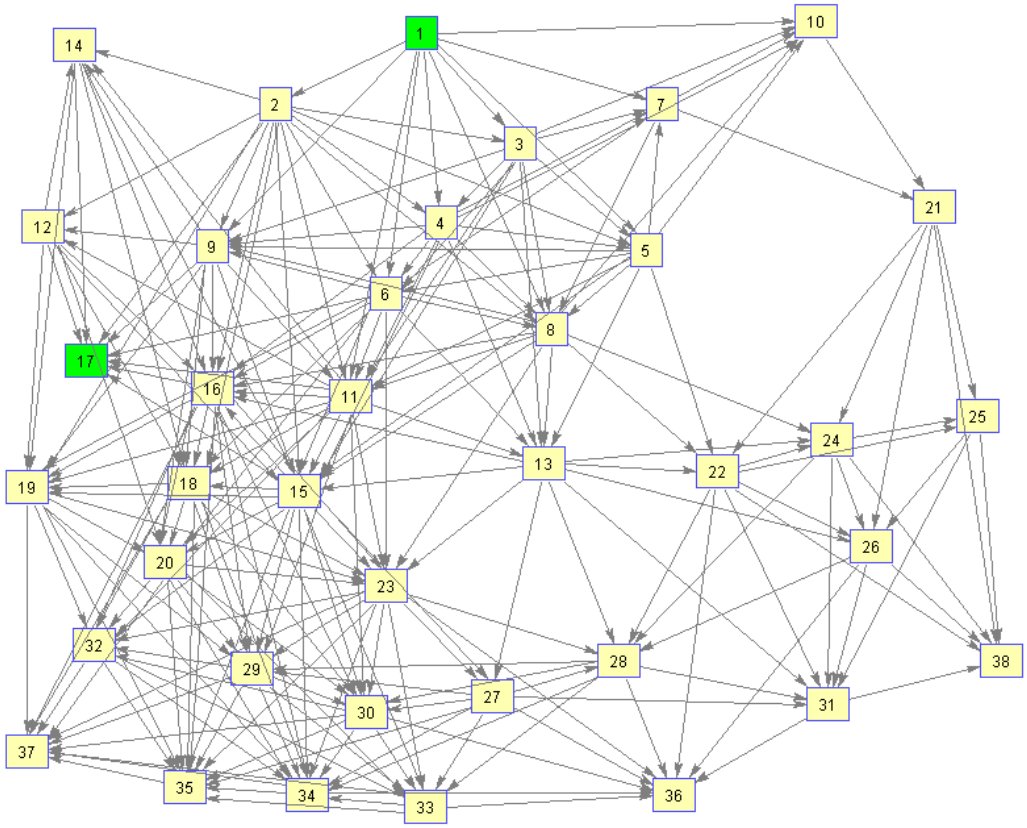



Figure 3.7: The representation of the WMN used in this model.

The network topology used in these experiments can be seen as a directed acyclic graph $G = (V, E)$ where $V$ represents nodes and $E$ the edges of the network. The network consists of a single source node, a single receiver node and intermediate nodes that perform RLNC.

The network had the following parameters:

- Area of 250 x 200 m
- Connection distance $d_{max} = 90$ m.
- Total number of nodes $numNodes = 38$.

For practical purposes the timing of the simulation was implemented according to the 2-hop interference model as described in [29]. We chose 38 nodes in the network as it is the same number of nodes used in the paper by [8]. The area was chosen to give the network good connectivity and the maximum indoor connection distance was used as in [30].

The data transmitted from the source to the receiver consisted of a 216 KB file that was divided into generations where each packet had a size of 1500 Bytes and a generation consisted of 32 packets. NC was performed over the $\mathbb{F}_{2^8}$ field, meaning that a symbol has the size of 1 Byte. We based the parameters in the experimental set-up on those that were used in [8].

The method followed for the experiments was the method of independent replications as described in [31]. This was done to ensure that the results that were obtained were not influenced by different network scenarios. For the simulations, 40 networks were generated with different seeds. For each network there was 5 instances with different source and receiver pairs for each instance. For the packet pollution and packet dropping scenarios, 5 different malicious nodes were chosen for each of the sub-instances. In this Monte-Carlo simulation 6300 independent simulations were run, 200 for each baseline and 1000 for each malicious node simulated in each of the NC, DART and PDD schemes.

## 3.5 Experiment Assumptions

In the development of this experiment, some assumptions were made in order to simplify the experiment.

**Assumptions with regard to the nodes in the network:**

- It is assumed that all nodes in the network are equal and have the same capabilities.
- The nodes in the network are not mobile.
- There are no stand-alone nodes in the network.
- The source node and receiver node are not malicious.
- All nodes have at least two connections.
- The link quality is 100% for all nodes and is not influenced by reception and transmission strength.

**Assumptions with regard to the network:**

- All nodes have the same chance to be the source or receiver node, but it cannot be both at the same time.
- A node can only send or receive in a given time unit.
- No specific routing protocol was implemented.
- The source knows the identity of the forwarder nodes.

## 3.6 Simulation Description

### 3.6.1 Node Descriptions

There are five different kinds of nodes used in this experiment, a source node, a receiver node, an intermediate node and a malicious node. The malicious nodes can be divided into packet pollution nodes and packet dropping nodes.

**Source Node**

The source node creates coded packets by performing RLNC and sends the packets into the network. The source node knows which nodes are in the forwarding path. The source can send checksum and coded packets and can receive acknowledgement packets.

**Receiver Node**

The receiver node receives coded packets and decodes the packets when enough linearly independent packets has arrived. The receiver node can receive coded and checksum packets, and can send acknowledgement packets.

**Intermediate Node**

The intermediate nodes are nodes that perform RLNC on the received packets and sends these new coded packets to their destination node. The intermediate nodes can send and receive all packets.

**Malicious Nodes**

Packet pollution nodes are intermediate nodes that corrupt or change the coded packets they receive. The malicious nodes forward these corrupt packets. Packet dropping nodes are intermediate nodes that drop all the packets they receive.

### 3.6.2 Simulation Set-up

In this section, the simulation set-up procedure that was followed is explained. The simulation flow is described in Fig. 3.8

**Node placement**

A total $(numNodes)$ of 38 nodes were placed randomly in a 250 m by 200 m area, to form a WMN. Each node's (x,y) coordinates were stored in a nodes-coordinate vector, $VNC$, along with an identification number unique to each node. A maximum connecting distance $d_{max}$ was defined as 90 m. The following variables were used in the Node Placement Algorithm, Algorithm 1:

Vertices Matrix - $VNC$

Number of nodes in network - $numNodes$

Connectivity Matrix - $AdjMatrix$

Maximum transmission distance - $d_{max}$

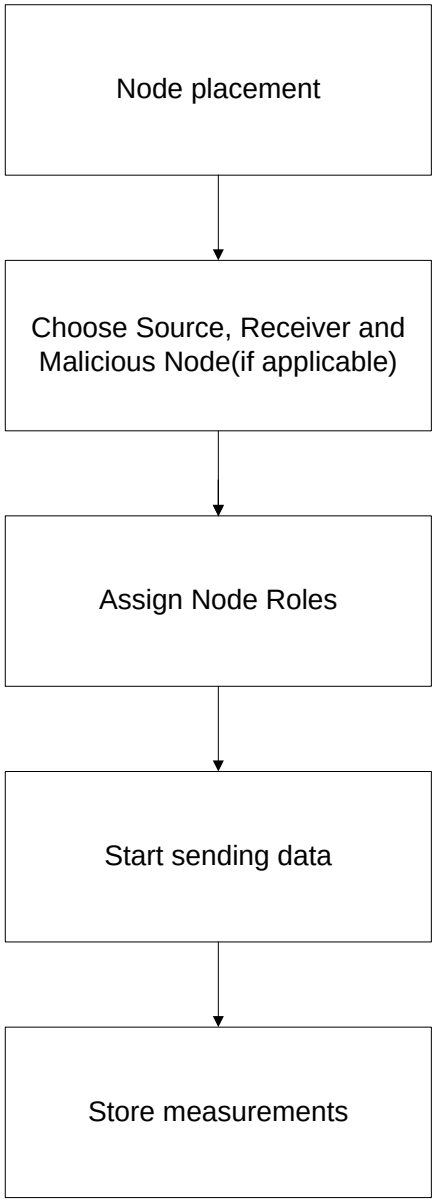



Figure 3.8: The Flow of the Simulation

**Algorithm 1** Node Placement and network set-up

```
Place nodes in VNC.
Define d_max = 90
Create a connectivity matrix AdjMatrix.
for i ∀ ∈ V do
  for j = 0 TO numNodes do
    if i ≠ j then
      Calculate the distance d_(i,j) between the nodes
      if d_(i,j) < d_max then
        AdjMatrix(i, j) = 1
      else
        AdjMatrix(i, j) = 0
      end if
    end if
  end for
end for
```

The connectivity matrix, $AdjMatrix$, in which a 1 represents a connection and 0 no connection, was constructed. This was determined by using a distance algorithm where if the distance between two nodes was less than $d_{max}$ the element was a 1 and if the distance between them was greater than $d_{max}$ the element was a 0.

**Choose source and Receiver**

The source and receiver nodes were chosen randomly from the nodes in $VNC$. Their IDs were stored in the $SourceID$ and $ReceiverID$ variables respectively. In the case of the presence of a malicious node, its ID was stored in the $MalID$. The source, receiver and malicious node never had the same ID in an instance.

**Assign Node Roles**

Each node in the network was assigned a role. These nodes were stored in the *Nodes* matrix. This is shown in Algorithm 2.

**Algorithm 2** Node Roles Assignment Algorithm

```
for i = 1 TO numnodes do
  if i = SourceID then
    Nodes(i) = SourceNode
  else
    if i = ReceiverID then
      Nodes(i) = SourceNode
    else
      if i = MalID then
        Nodes(i) = MaliciousNode
      else
        Nodes(i) = ForwarderNode
      end if
    end if
  end if
end for
```

**Start Sending Data**

The source starts sending the data to all the nodes it is connected to. The source keeps on sending data until all the generations has been acknowledged from the receiver. We denote $t$ as the time-step and assume that one packet is send per time-step. The source multicasts 32 packets to all the nodes it is connected to. After every 32 packets sent, the source broadcasts a checksum packet to all its connected nodes. We assume that transmission of a packet is instantaneous and that it is transferred from the outbound queue of the sending node to the inbound queue of the receiving node. The next node that can send a packet, is chosen according to the 2-hop interference model in [29]. Only nodes that have any packets to send are eligible for a turn to send. For each turn the chosen node can send a checksum packet if applicable and a coded packet while increasing the timestep accordingly. This process continues until the receiver has enough verified packets to decode a generation. The receiver then sends an acknowledgement packet and a HealthMatrix packet to the source. When the source receives

the acknowledgement packet, it removes the acknowledged generation from the active generation list and activates the next generation. When the source receives the HealthMatrix packet it compares the HealthMatrix to the known node forwarder list. If there are any missing entries the source declares them malicious nodes. The source also checks the matrix for any entries in the HealthMatrix for the acknowledged generation that are $(NodeID\ 0\ 0)$ and declares the nodes malicious. The declared malicious nodes are then removed from the connectivity matrix. This whole process repeats until the receiver has received all the generations.

**Store Measurements**

After the source has stopped sending, all variables are stored to be analysed. After all the variables are stored the simulation terminates. We measured the total time that the simulation ran in timesteps. For the throughput we measured all the packets at the receiver node that were used for decoding the generations. For the latency of the network, we measured the time it took to decode the first generation at the receiver. We also recorded all the identified malicious nodes to check whether they were detected correctly.

*This chapter provided a description of the experiment.*

*In section 3.1 the RLNC implementation was explained, while section 3.2 explained how the DART security scheme was implemented and how checksum packets were generated. Lastly, in section 3.3 the security enhancement, the packet dropping detection scheme, was described and explained.*

*In section 3.4 the experimental set-up was described and the assumptions concerning the experiment was stated in section 3.5.*

*In section 3.6 the simulation was described briefly. This included the node descriptions in section 3.6.1 where source, receiver, forwarder and malicious nodes were described and the simulation set-up described in section 3.6.2.*

# Chapter 4

# Verification and Validation

*In this chapter we validate the experimental model and verify the implementation of the experimental design by verifying several core modules. The real and expected outputs are compared to verify each module. The simulation's outputs are validated to see if they are accurate enough for operational validation.*

## 4.1 Experimental Model Validation

Conceptual model validation is defined by [12] as

> *'determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is "reasonable" for the intended purpose of the model.'*

In this case the conceptual model is the experimental model described in Chapter 3. The model was validated by creating scenarios to see if the model outputs confirmed the assumptions.

### 4.1.1 Normal Source Receiver Network Scenario

**Description**

In this scenario the network consists of a source node, *S* and a receiver node, *R* as seen in Fig. 4.1. In this scenario there can be no malicious nodes. A 216 kB image was sent through the network from the source to the receiver. There is no security features implemented in the network. The packet size is 1500 B and the generation size 32.

**Assumptions**

We assume that if the image was transferred successfully that the functions (NC coding and decoding, packet transfer) in the simulation was implemented correctly. We also assume that the time it takes to transfer the packets should be in the area of the number of packets sent from the source node along with the time it takes an acknowledgement packet to arrive at the source.

**Outcome**

The image was transferred successfully, and was not corrupt. The number of packets sent and received at each node was recorded and is shown in Table 4.1.

Table 4.1: The outputs of the simulated Normal network

| Description | Simulation Outputs | Expected Outputs |
|---|---|---|
| **Transfer Time** | **165** | **165** |
| Number of packets sent by S | 165 | 165 |
| Number of packet received by R | 160 | 160 |
| Number of relative packets received by R | 160 | 160 |
| Number of non-relative packets received by R | 0 | 0 |
| Number of Acknowledgement packets received by S | 5 | 5 |
| **Throughput (packets per tick)** | **0.96** | **0.96** |

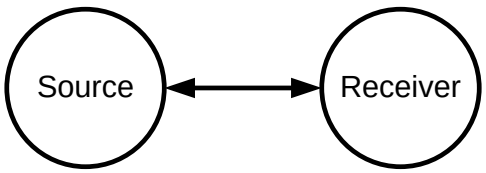



Figure 4.1: The representation of the network for Example 1.

It takes an average of 33 ticks to send a generation of size 32. As simulation results correspond to the expected results it is assumed that the assumptions are correct.

### 4.1.2 Source Receiver with DART Security Scenario

In this scenario the network consists of a source node, *S* and a receiver node, *R* as seen in Fig. 4.1. The DART security scheme is implemented. A checksum packet is sent after each generation(32 packets). In this scenario there can be no malicious nodes. A 216 kB image was sent through the network from the source to the receiver.

**Assumption**

We assume that if the image was transferred successfully that the functions (NC coding and decoding), packet transfer as well as checksum packet verification in the simulation was implemented correctly. We also assume that the time it takes to transfer the packets should be 2 percent more than the scenario described in Section 4.1.1.

**Outcome**

The image was transferred successfully, and was not corrupt. The number of packets sent and received at each node was recorded and shown in Table 4.2.

Table 4.2: The outputs of the simulated DART network

| Description | Simulation Outputs | Expected Outputs |
|---|---|---|
| **Transfer Time** | **170** | **170** |
| Number of coded packets sent by S | 160 | 160 |
| Number of checksum packets sent by S | 5 | 5 |
| **Total number of packets sent by S** | **165** | **165** |
| Number of coded packets received by R | 160 | 160 |
| Number of checksum packets received by R | 5 | 5 |
| Number of relative coded packets received by R | 160 | 160 |
| Number of non-relative packets received by R | 0 | 0 |
| **Total number of packets received by R** | **165** | **165** |
| Number of Acknowledgement packets received by S | 5 | 5 |
| **Throughput (packets per tick)** | **0.94** | **0.94** |

The throughput of the DART system is 2% more than in scenario 4.1.1. As simulation results correspond to the expected results it is assumed that the assumptions are correct.

### 4.1.3 Source Receiver with PDD Security Scenario

In this scenario the network consists of a source node, *S* and a receiver node, *R* as seen in Fig. 4.1. The PDD security scheme is implemented. A checksum packet is sent after every generation. In this scenario there can be no malicious nodes. A 216 kB image was sent through the network from the source to the receiver.

**Assumption**

We assume that if the image was transferred successfully that the functions (NC coding and decoding), packet transfer as well as checksum packet verification in the simulation was implemented correctly. We also assume that the time it takes to transfer the packets should be the same as the scenario described in section 4.1.2.

**Outcome**

The image was transferred successfully, and was not corrupt. The number of packets sent and received at each node was recorded and shown in Table 4.3.

Table 4.3: The outputs of the simulated PDD network

| Description | Simulation Outputs | Expected Outputs |
|---|---|---|
| **Transfer Time** | **170** | **170** |
| Number of coded packets sent by S | 160 | 160 |
| Number of checksum packets sent by S | 5 | 5 |
| **Total number of packets sent by S** | **165** | **165** |
| Number of coded packets received by R | 160 | 160 |
| Number of checksum packets received by R | 5 | 5 |
| Number of relative coded packets received by R | 160 | 160 |
| Number of non-relative packets received by R | 0 | 0 |
| **Total number of packets received by R** | **165** | **165** |
| Number of Acknowledgement packets received by S | 5 | 5 |
| **Throughput (packets per tick)** | **0.94** | **0.94** |

As simulation results correspond to the expected results it is assumed that the assumptions are correct.

## 4.2 Computerized Model Verification

Computerised Model Verification is defined by [12] as

> *'assuring that the computer programming and implementation of the conceptual model is correct.'*

For verification of the simulation model, each time an action takes place, output in the form of a message to the command line, is produced. These messages are used to visually verify that the packets did indeed arrive at the various nodes and that they were processed correctly. Packet counters were initialised in each node, to verify that the amount of packets sent correspond to the amount of packets received.

**NC Encoding function**

To test the NC coding algorithm, inputs were given to the function as described in Table 4.4. The outputs were calculated by hand to verify the MATLAB function outputs described in Table 4.5.

Table 4.4: Inputs for NC Encoding function test

| **Packets** | **Code Vectors** |
|---|---|
| $\vec{p_1} = (1, 98, 3, 34, 5)^\top$ | $c1 = (140, 181, 74)$ |
| $\vec{p_2} = (85, 8, 6, 57, 30)^\top$ | $c2 = (130, 228, 229)$ |
| $\vec{p_3} = (49, 178, 164, 37, 100)^\top$ | $c3 = (32, 53, 13)$ |

The hand calculation of the first coded packet is described below:

$\vec{e_1} = c1_{(1,1)} * \vec{p_1} + c1_{(1,2)} * \vec{p_2} + c1_{(1,3)} * \vec{p_3}$

$\vec{e_1} = 140 * (1, 98, 3, 34, 5)^\top + 181 * (85, 8, 6, 57, 30)^\top + 74 * (49, 178, 164, 37, 100)^\top$

$\vec{e_1} = (140, 245, 137, 85, 134)^\top + (130, 193, 153, 200, 199)^\top + (43, 27, 110, 202, 247)^\top$

$\vec{e_1} = (37, 47, 126, 87, 182)^\top$

Table 4.5: Inputs for function test

| **Function Output** | **Calculated Output** |
|---|---|
| $\vec{e_1} = (37, 47, 126, 87, 182)^\top$ | $\vec{e_1} = (37, 47, 126, 87, 182)^\top$ |
| $\vec{e_1} = (106, 211, 88, 147, 250)^\top$ | $\vec{e_1} = (106, 211, 88, 147, 250)^\top$ |
| $\vec{e_1} = (23, 208, 153, 169, 50)^\top$ | $\vec{e_1} = (23, 208, 153, 169, 50)^\top$ |

The simulated and calculated values correlate, thus we assume that the implemented function is correct.

### NC Decoding function

To test the decoding function the verified coded packets and the coding vectors are used as inputs to the function. If the output is the same as the packets in Table 4.6, the function works correctly. The output after decoding is seen in Fig. 4.2 and correlates to the packets in Table 4.6.

Table 4.6: Inputs for the NC Decoding function test

| **Packets** | **Code Vectors** |
|---|---|
| $\vec{p_1} = (1, 98, 3, 34, 5)^\top$ | $c1 = (140, 181, 74)$ |
| $\vec{p_2} = (85, 8, 6, 57, 30)^\top$ | $c2 = (130, 228, 229)$ |
| $\vec{p_3} = (49, 178, 164, 37, 100)^\top$ | $c3 = (32, 53, 13)$ |

```
DecodedGeneration =

     1    85    49
    98     8   178
     3     6   164
    34    57    37
     5    30   100
```

Figure 4.2: The output by MATLAB after decoding

### Verifying Packets

The built-in random number generator of Matlab is used to generate the $H_s$ matrix. To verify the packet verifying function, a $H_s$ matrix was created with the $seed, s = 5$ and $b = 2$ and applied to the generation G that consisted of the packets in Table 4.4 to create the checksum matrix $CHK_s$ as seen in Fig. 4.3.

```
G =

     1    85    49
    98     8   178
     3     6   164
    34    57    37
     5    30   100


Hs =

     0     0     0     1     0
     1     1     1     1     0


CHK_s = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          34          57          37
          66          98           2
```

Figure 4.3: The $H_s$ and $CHK_s$ matrices generated by MATLAB for generation G

In order to verify the packets the following equation must hold

$$CHK_s(G)\vec{c} = H_s\vec{e} \tag{4.1}$$

where we define the right side as $RSide = CHK_s(G)\vec{c}$ and the lefts side as $LSide = H_s\vec{e}$.

The output of the equation in 4.1 with the inputs of the code vector $c1 = (140, 181, 74)$ and coded packet $\vec{e_1} = (37, 47, 126, 87, 182)^\top$ is shown in Fig. 4.4. As seen in Fig. 4.4 $RSide = LSide$ thus the packet (c1, e1) is valid. In Fig. 4.5 the coded packet was modified to $\vec{e_1} = (37, 47, 2, 87, 182)^\top$, to show that $RSide \neq LSide$ making the packet invalid.

```
c1 =

   140   181    74


e1 = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          37
          47
         126
          87
         182


RSide = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          87
          35


LSide = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          87
          35
```

Figure 4.4: The result of *RSide* and *LSide* for the code vector c1 and coded packet e1

```
c1 =

   140   181    74


e1 = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          37
          47
           2
          87
         182


RSide = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          87
          35


LSide = GF(2^8) array. Primitive polynomial = D^8+D^4+D^3+D^2+1 (285 decimal)

Array elements =

          87
          95
```

Figure 4.5: The result of *RSide* and *LSide* for the code vector $\vec{c1}$ and the invalid coded packet $\vec{e1}$

### Malicious Packet Dropping

The malicious packet dropping node was implemented to drop coded packets. To test if packets were dropped the source node sent 32 packets to the receiver via a malicious node. The malicious node was set to drop the first 10 packets it received. The output of the nodes is described in Table 4.7. As seen in the table 10 packet were dropped, thus Malicious Packet Dropping is verified.

Table 4.7: The outputs of the nodes after the source sent 32 packets

| **Description** | **Source** | **Malicious** | **Receiver** |
|---|---|---|---|
| Received ENC Packets | | 32 | 22 |
| Received TOTAL Packets | | 32 | 22 |
| Send ENC Packets | 32 | 22 | |
| Send TOTAL Packets | 32 | 22 | |
| Dropped Packets | | 10 | |
| Relative Generation Packets | | | 22 |

**Malicious Packet Pollution**

The malicious packet pollution node was implemented to pollute coded packets. To test if packets were polluted the source node sent packets to the receiver via a malicious node until the receiver decoded a generation. The malicious node was set to pollute the first packet it received. The output of the simulation where the packet was polluted and where the generation was decoded is seen in Fig. 4.6. The generation is polluted as seen on the bottom of Fig. 4.6. While, the output for the nodes is described in Table 4.8. The packets that were dropped by the receiver node was the polluted generation packets. Malicious Packet Pollution is verified.

```
t =

     3

1 Encoded and sending packet >>>>>
22 sending POL packet >>>>>
22 Received an Un/Encoded Packet -----
10 Received Un/Encoded Packet -----
10 Putting into Decoding Matrix
```

```
t =

    37

1 Encoded and sending packet >>>>>
22 sending packet >>>>>
22 Received an Un/Encoded Packet -----
10 Received Un/Encoded Packet -----
10 Putting into Decoding Matrix
10 Decoding Generation
generataion polluted
the end
```

Figure 4.6: The output of the simulation where the packet was polluted and where the generation was decoded

## 4.3 Operational Validation

Operational Validity is defined by Sargent [12] as

> *'determining that the model's output behaviour has sufficient accuracy for the model's intended purpose over the domain of the model's intended applicability.'*

Table 4.8: The outputs of the nodes after the receiver decoded a generation

| **Description** | **Source** | **Malicious** | **Receiver** |
|---|---|---|---|
| Received ENC Packets | | 33 | 32 |
| Received TOTAL Packets | | 33 | 32 |
| Send ENC Packets | 34 | 32 | |
| Send TOTAL Packets | 34 | 32 | |
| Dropped TOTAL Packets | | 0 | 32 |
| Polluted Packets | | 1 | |
| Polluted Generations | | | 1 |

We interpret this as seeing whether the research question is answered. In this case,

Does the method add additional security to the DART security scheme?

As seen from the results in Chapter 4 the Packet Dropping Detection scheme does add additional security to the DART scheme as it detects malicious packet dropping nodes in the network and drops them form the network. Thus, the outputs of the model are validated.

---

*This chapter provided a description of the validation and verification of the research. Section 4.1 described the validation of the experimental model.*

*Section 4.2 described the computerized model verification by verifying core modules of the simulation.*

*The operational validation of the experiment was described in section 4.3.*

---

# Chapter 5

# Simulation Results

*In this chapter the simulation results are discussed. The throughput of the network without security, the network with DART security and the network with PDD is shown and compared to one another.*

## 5.1 Simulation Parameters

The system without any enhancements is referred to as the **Normal** system, the system with DART security is referred to as **DART** and the system with DART security and Packet Dropping node detection is referred to as the **PDD** system.

Time was measured in ticks, where 1 tick represented one time unit. The size of a generation was 32 packets and the size of the file sent through the network was 216 kB. The packet size was 1500 B. For the DART security scheme a checksum packet was sent after a generation was sent into the network. This implies that a checksum packet was sent after 32 data packets were sent into the network. The size of the number of active generations, $k$ was 5. Each simulated network contained 38 nodes. These parameters

were taken from [8].

The throughput was measured by using the equation

$$Throughput = \frac{\text{Number of decoded packets at R}}{\text{The time taken to send the file}} \tag{5.1}$$

where the number of decoded packets received at the receiver, R, refers to the number of faultless coded packets used to decode all the generations correctly.

The latency of the network was measured as the time it took to decode the first generation.

The malicious nodes simulated packet dropping and packet pollution attacks. These attacks represent black-hole type of attacks. In the packet dropping case all the packets the malicious node received were dropped and for the packet pollution case all the data packets that the malicious node sent were polluted.

## 5.2 Throughput of the networks without malicious nodes present

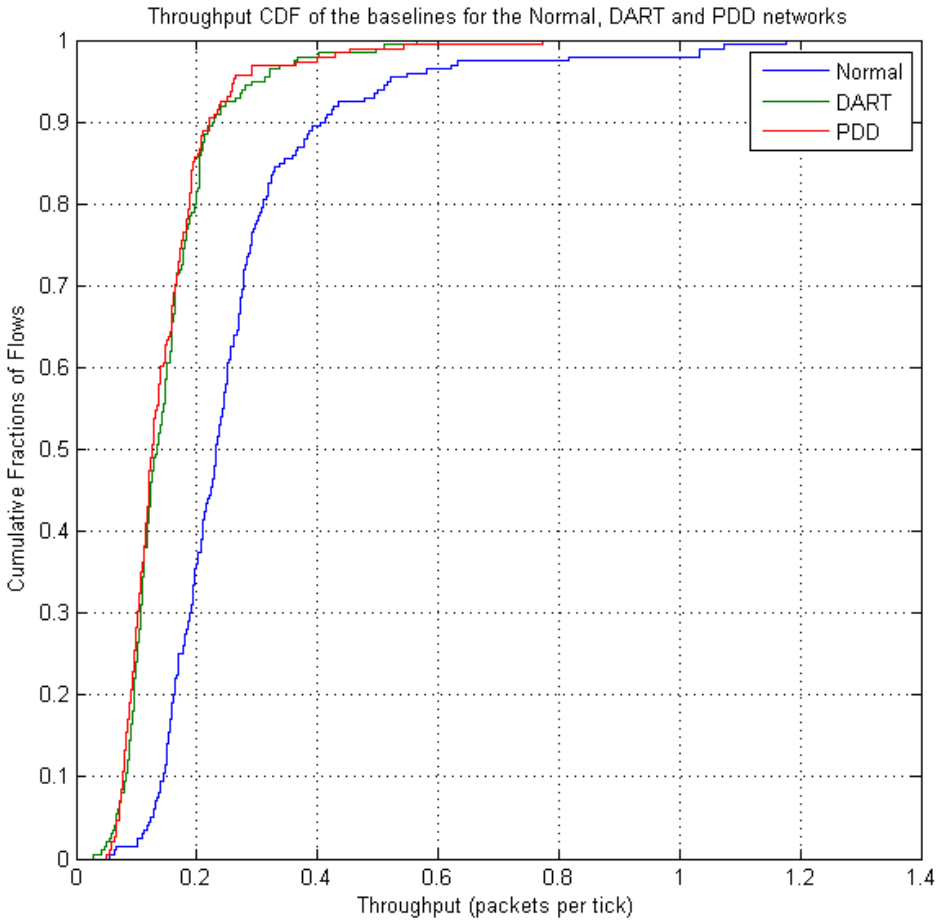



Figure 5.1: The CDF graph for the throughput of the Normal, DART and PDD schemes

A baseline throughput was established for all three schemes (Normal, DART and PDD). As described in Chapter 3, the baseline throughput was measured for 40 different networks and for each network 5 random source and receiver pairs were chosen. The throughput for each scheme is shown in Fig. 5.1. The cumulative fraction of flows in Fig. 5.1 represents the probability of the throughput.

As can be seen in Fig. 5.1 the throughput for the Normal network is much higher than for the DART and the PDD schemes. This is because the DART scheme delays coded

packets at the intermediate nodes for verification. The throughput for the DART and PDD schemes are relatively the same because the PDD scheme is the DART scheme that has modified checksums. The degradation in throughput from the Normal system to the DART and PDD systems is about 43% and 46% respectively when comparing the average throughput.

## 5.3 Throughput of the Normal network with a malicious node present

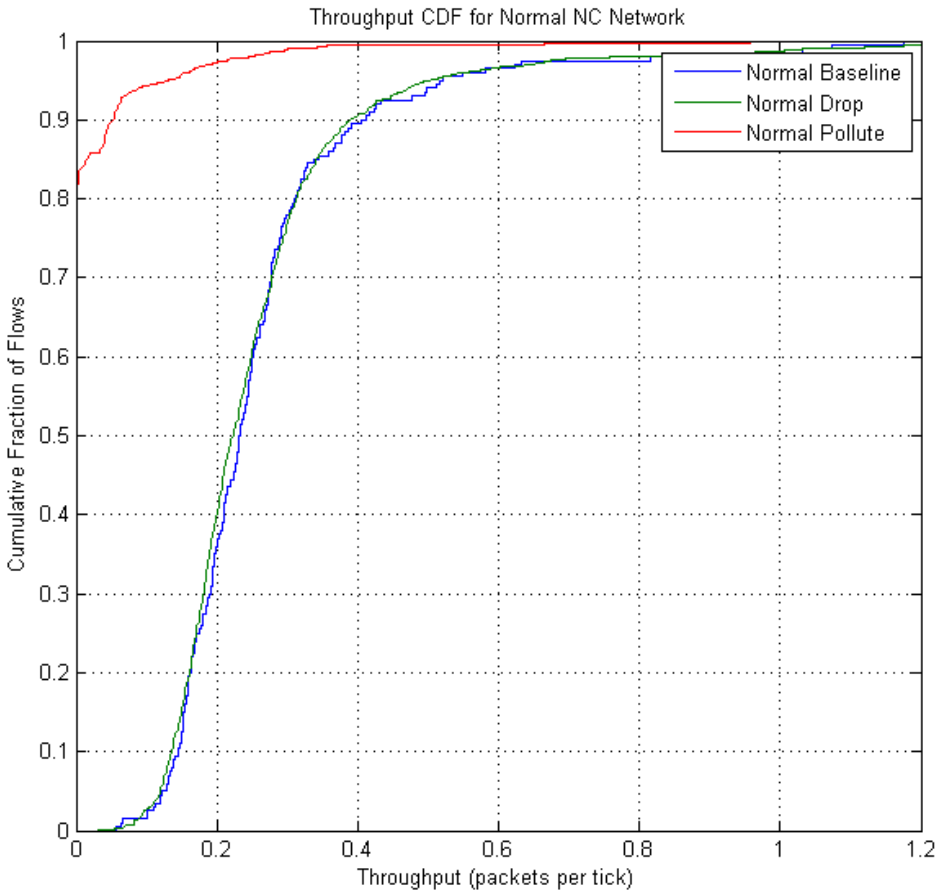



Figure 5.2: The CDF graph for the throughput of the Normal scheme with a malicious node present

The throughput for the case of a malicious pollution node present in the network is shown in Fig. 5.2. Fig. 5.2 shows the high impact that packet pollution can have on a normal NC network. Only one polluted packet can have a devastating effect on the throughput of the network because when that generation is decoded the whole generation is corrupt. When a polluted packet is used in the RLNC process at intermediate nodes those new coded packets are also corrupt. It can be seen that for at least 80% of the flows the throughput was non existent and that 92% of the flows have a throughput of less than 0.06 packets per tick. With a malicious packet dropping node present, the decrease in throughput is not as high as packet pollution. The average difference in throughput is 0.0061 packets per tick which is a degradation of 2% from the Normal system.

## 5.4 Throughput of the DART network with a malicious node present

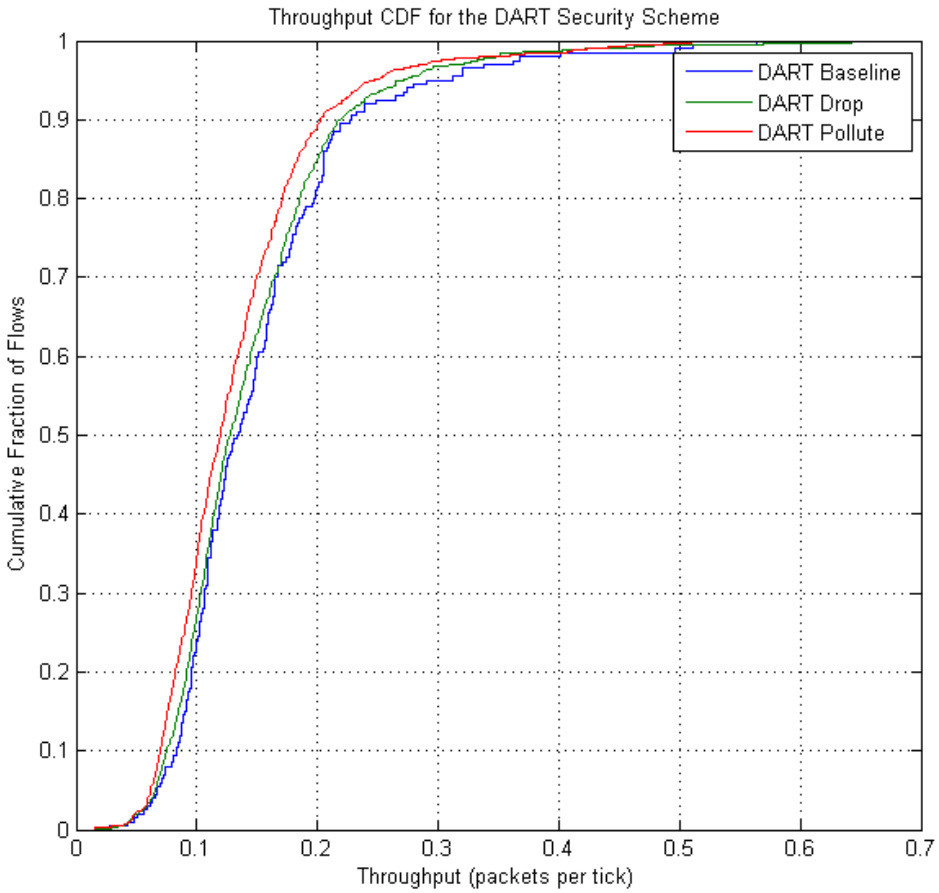



Figure 5.3: The CDF graph for the throughput of the DART scheme with a malicious node present

The throughput for the DART scheme is shown in Fig. 5.3. The effect of packet pollution is much greater than the effect of packet dropping. The degradation of the throughput in the case of packet pollution is 0.0181 packets per tick, while for packet dropping the degradation is 0.0069 packets per tick. From the graph it is evident that the DART scheme has reduced the degrading effect on the throughput that packet pollution has on the network. The DART scheme cannot detect packet dropping nodes in the network.

## 5.5 Throughput of the PDD network with a malicious node present

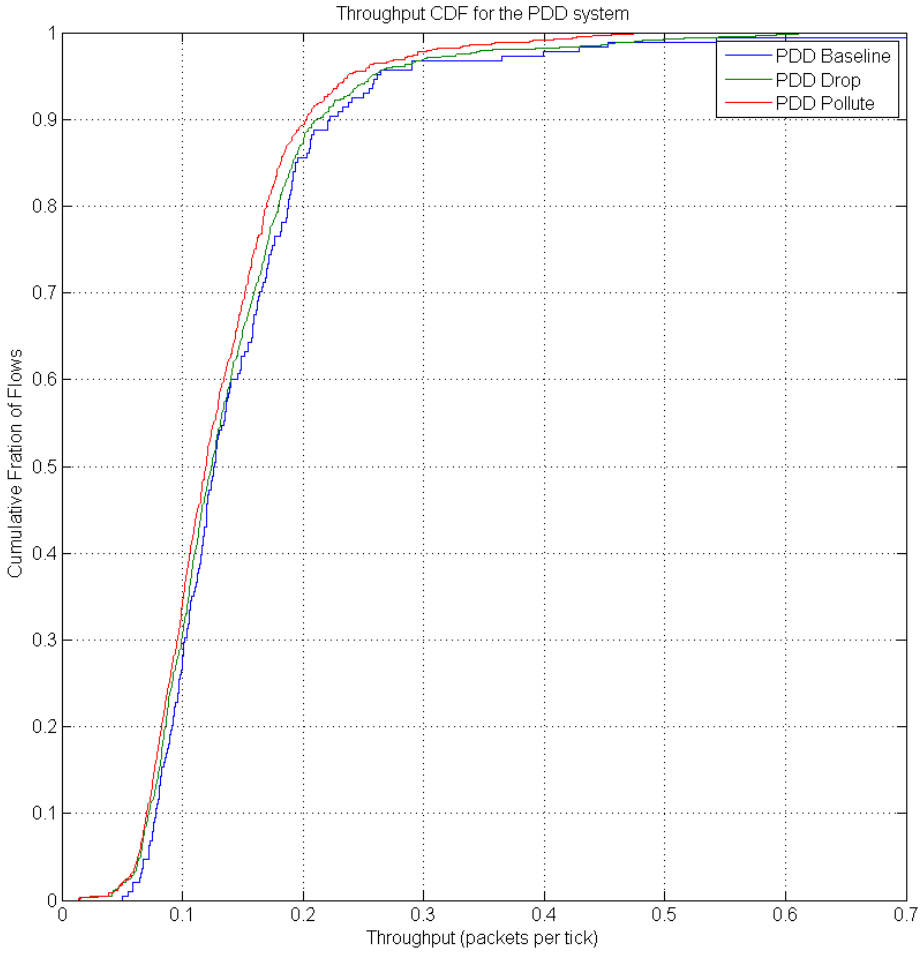



Figure 5.4: The CDF graph for the throughput of the PDD scheme with a malicious node present

The throughput for the PDD scheme is shown in Fig. 5.4. It can be seen that packet pollution has a larger degrading effect on the throughput than packet dropping. The polluted packets fill the unverified queues of the nodes causing viable unpolluted packets to be dropped. When comparing the difference in throughput at the median of the graph in Fig. 5.4, the difference between the baseline case and the packet dropping

case is 0.0021 packets per tick. The difference between the baseline and the packet pollution case at the median is 0.0067 packets per tick. The increase in throughput effect that the DART scheme has on packet pollution can also be seen in Fig. 5.4 and correlates with Fig. 5.3. This indicates that the enhancement made to the DART scheme did not have a severe degrading effect on the throughput of the network. The PDD system can detect the packet dropping nodes while not having a severe negative effect on the throughput of the network.

In the simulation where a malicious packet dropping node was present in the network the malicious nodes that were detected by the PDD scheme, was recorded. These detections are described in Table 5.1. The PDD scheme detected the malicious nodes in all the iterations giving it a 100% detection rate. There were however false positives, where a node was identified as malicious but was not. These false positives consisted of 0.061% of the detections. These false positive detections happened when the malicious node was a distance of one hop away from the receiver and the intermediate nodes only path to the receiver. In other words the malicious node blocked the flow of information for other intermediate nodes. An example of this is described in section 5.8.

Table 5.1: Detection Rate of the PDD scheme

| | **Malicious node detected** | **Node falsely detected as malicious** |
|---|---|---|
| **Detection Rate** | 100% | 0.061% |

## 5.6 Comparing the throughput for the Normal, DART and PDD schemes

The throughput for all 3 three schemes with the presence of a packet dropping node and a packet pollution node is compared in Fig. 5.5 and Fig. 5.6 respectively. The respective statistics for the throughput is shown in Table 5.2.

Table 5.2: Throughput of the networks

| | Mean (packets per tick) | Median (packets per tick) |
|---|---|---|
| **Normal** | 0.2646 | 0.2321 |
| **Normal Drop** | 0.2585 | 0.2225 |
| **Normal Pol** | 0.0200 | 0.0000 |
| **DART** | 0.1512 | 0.1341 |
| **DART Drop** | 0.1443 | 0.1273 |
| **DART Pol** | 0.1331 | 0.1197 |
| **PDD** | 0.1473 | 0.1263 |
| **PDD Drop** | 0.1402 | 0.1242 |
| **PDD Pol** | 0.1320 | 0.1196 |

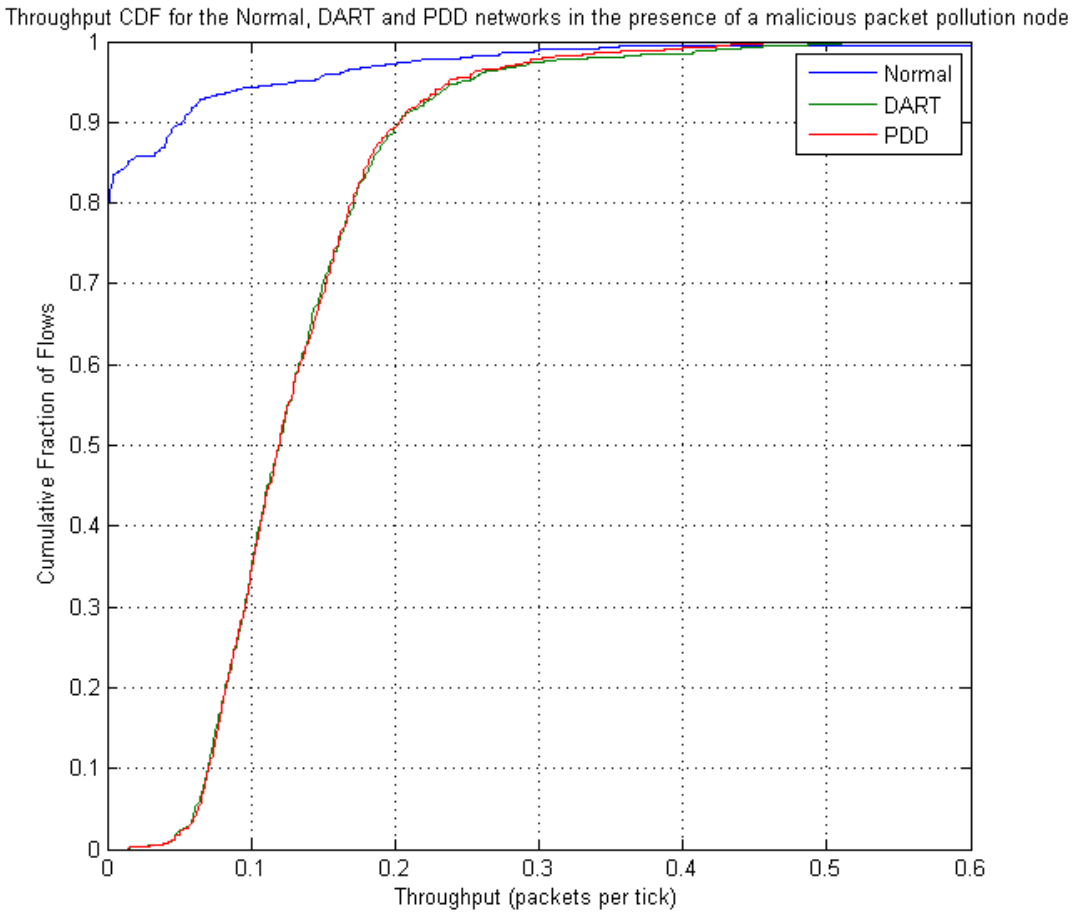



Figure 5.5: The CDF graph for the throughput of the Normal, DART and PDD schemes with a malicious pollution node present

Fig. 5.5 shows that packet pollution has a severe degrading effect on the throughput of the Normal networks and that by implementing the DART and PDD schemes the throughput can be increased. The throughput for the DART and PDD schemes are the same because they handle packet pollution in the same way. The PDD scheme is an enhancement of the DART scheme to detect packet dropping nodes and still handle packet pollution.

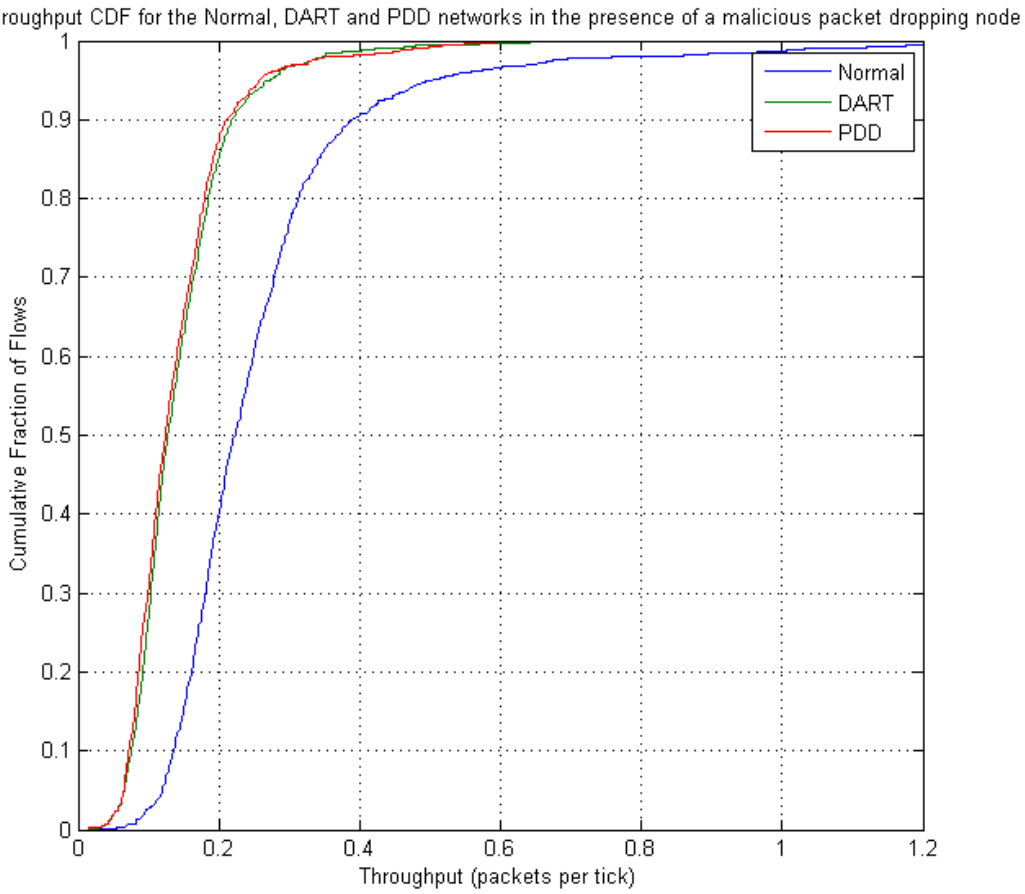



Figure 5.6: The CDF graph for the throughput of the Normal, DART and PDD schemes with a malicious packet dropping node present

In the case where a malicious packet dropping node is present the difference in throughput between the Normal scheme and the DART and PDD schemes is as expected and correlates with the baseline throughputs in Fig. 5.1.

Fig. 5.6 shows that there is a marginal difference in the throughput for the DART and

PDD schemes. The throughput is almost the same because removing the malicious packet dropping node has the same effect as not doing anything at all. The packets that were dropped, were also broadcast to other surrounding nodes, except in the case were a node is only connected to the malicious node. The PDD system can detect these malicious packet dropping nodes and remove them from the network. An alert can be generated to alert technicians so that they can investigate whether the node is compromised or faulty. By identifying a malicious node in the network the security of the network is improved. By improving the security of the network without a significant loss in throughput, the DART scheme was enhanced to detect malicious packet dropping nodes.

## 5.7 Latency of the networks

The latency of the networks is the time it takes for a decoded packets to arrive at the receiver. Fig. 5.7 shows the latency for the three schemes in a network without any malicious nodes present. It can be seen that the latency for a normal NC network is lower than for the DART and PDD schemes. This occurs because the packets are delayed by the verification mechanism of the DART scheme. The difference in the latency between the DART and PDD scheme is 23 ticks where the median for the DART scheme is 584 ticks and the median for the PDD scheme is 623 ticks. The difference between the normal NC system and the DART and PDD systems is 268 ticks and 229 ticks respectively. It can also be seen that the DART and PDD schemes have a larger impact on the latency for networks that already have a high latency. These networks typically have a longer path length than the networks with a higher throughput. The latency is higher because there are more nodes on the path that is subjected to the verification delay mechanism of the DART scheme. 50% of the networks have a latency of less than 584 ticks while 97% of the networks has a latency of less than 2000 ticks.

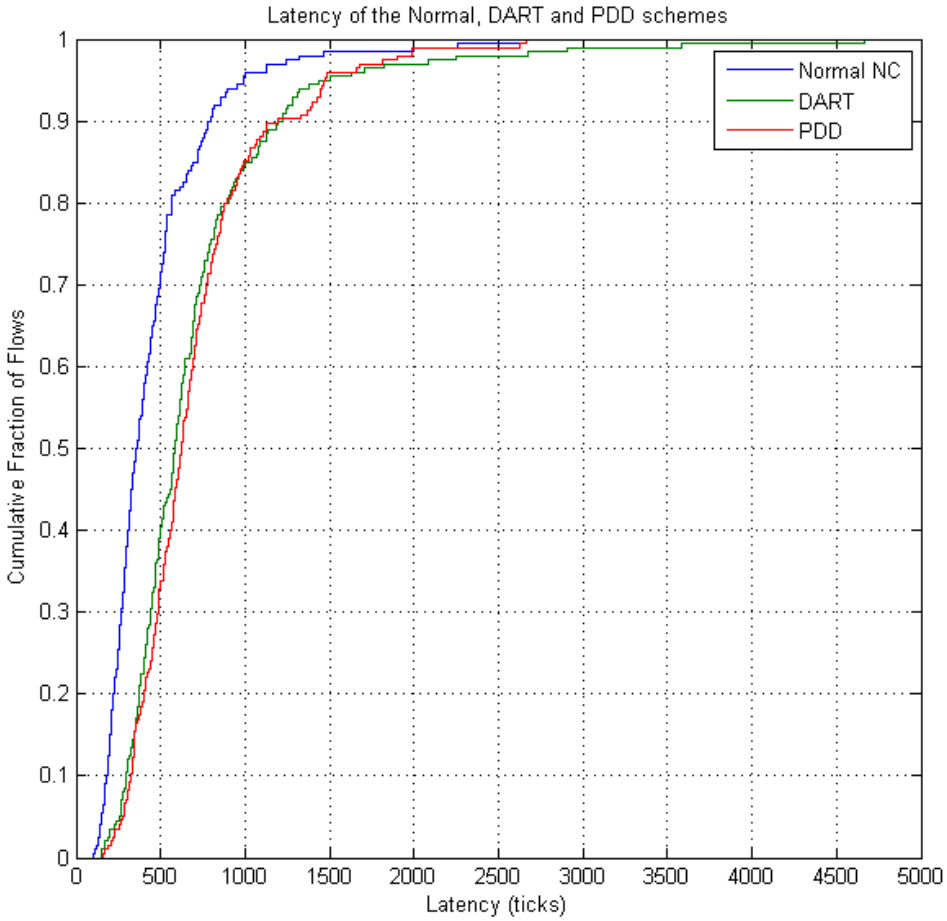



Figure 5.7: The CDF graph for the latency of the Normal, DART and PDD schemes

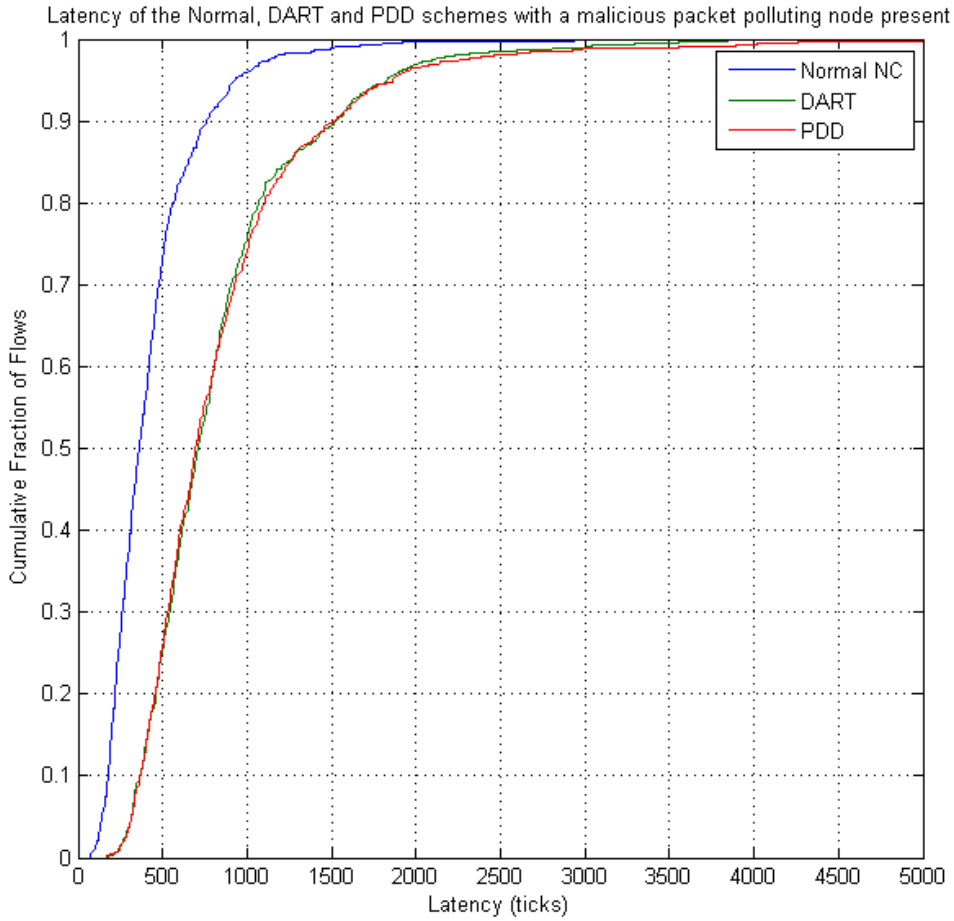



Figure 5.8: The CDF graph for the latency of the Normal, DART and PDD schemes with a malicious packet polluting node present

In Fig. 5.8 the latency for the Normal, DART and PDD scheme in the presence of a malicious packet polluting node is shown. The latency for the DART and PDD schemes are relatively the same where the median latency for DART is 711 ticks and the median latency for PPD is 695.5 ticks. The latencies are the same because the DART and PDD schemes react the same to malicious packet dropping nodes.

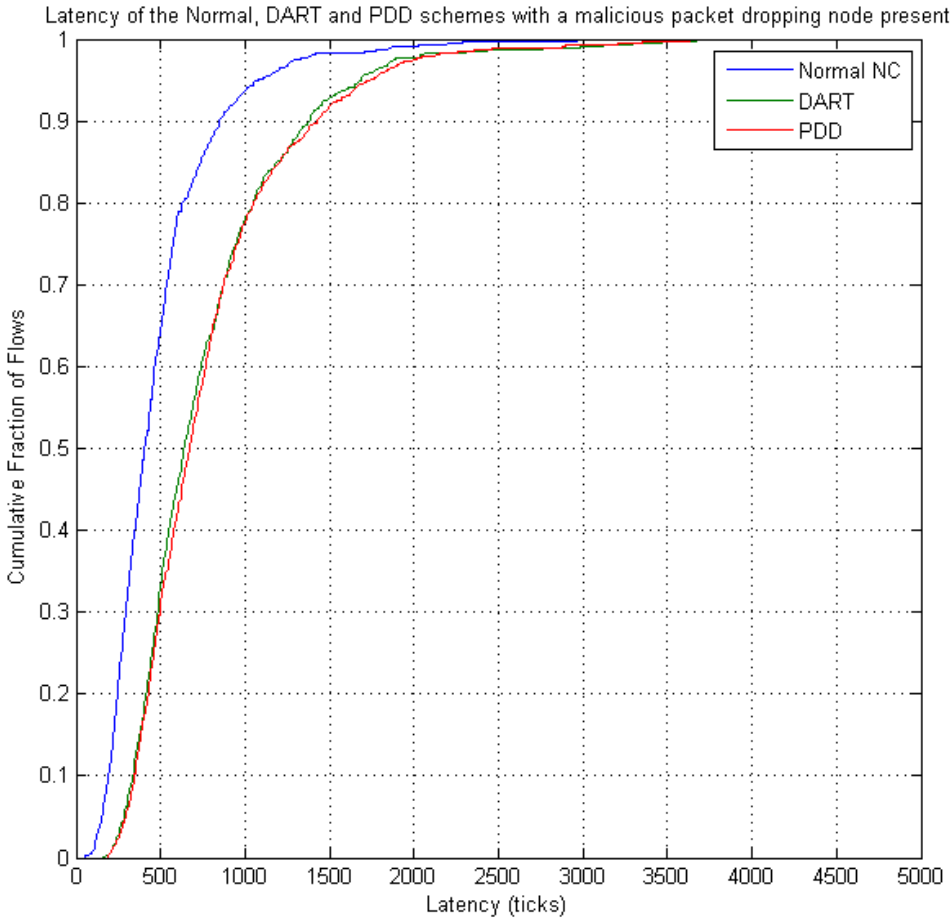



Figure 5.9: The CDF graph for the latency of the Normal, DART and PDD schemes with a malicious packet dropping node present

Fig. 5.9 shows the latency for the Normal, DART and PDD schemes in the presence of a malicious packet dropping node. The latency at the median for the Normal scheme is 405.5 ticks while the median for the DART and PDD schemes are 640 ticks and 672 ticks respectively. The latencies for the DART and PDD are relatively the same except for where the cumulative fraction of flows is between 0.35 and 0.64. where the difference is 32 ticks. This can be attributed to the change in the network topology when the malicious node is detected and removed form the network. The respective statistics for the graphs in Fig. 5.7, Fig. 5.9 and Fig. 5.8 are given in Table 5.3.

Table 5.3: Latency of the networks

| | Mean (ticks) | Median (ticks) |
|---|---|---|
| **Normal** | 447.4 | 355.5 |
| **Normal Drop** | 485.7 | 405.5 |
| **Normal Pol** | 429.5 | 361.0 |
| **DART** | 702.3 | 584.0 |
| **DART Drop** | 773.5 | 640.0 |
| **DART Pol** | 834.5 | 711.0 |
| **PDD** | 706.7 | 623.0 |
| **PDD Drop** | 791.0 | 672.0 |
| **PDD Pol** | 864.0 | 695.5 |

## 5.8 Detection of packet dropping nodes

Although there is a small loss (2.5%) in throughput (in comparison with DART) when implementing the PDD scheme as can be seen in Fig. 5.1, Fig. 5.6 and Fig. 5.5 it manages to detect packet dropping nodes in the forwarder set of nodes. The small decrease in throughput can be attributed to the fact that the checksum packet is larger due to the added HealthMatrix. The PDD scheme detects and drops these malicious nodes form the network. The information about which nodes are malicious can be a helpful diagnostic that can be used to indicate faulty nodes in a network. When faulty nodes are identified they can be replaced, or they can be checked and repaired. By a simple addition to the DART scheme more security was added with the only cost being a small, 2% degradation in the throughput.

In the simulation where a malicious packet dropping node was present in the network the malicious nodes that were detected by the PDD scheme, was recorded. These detections are described in Table 5.1. The PDD scheme detected the malicious nodes in all the iterations giving it a 100% detection rate. There were however false positives, where a node was identified as malicious but was not. These false positives consisted of 0.061% of the detections.

These false positive detections happened when the malicious node was a distance of one hop away from the receiver and the intermediate node's only path to the receiver.

In other words, the malicious node blocked the flow of information to other intermediate nodes. An example of this can be seen in Fig. 5.10. Consider a network with a source, receiver and 7 intermediate nodes where one of the intermediate nodes is a malicious packet dropping node.

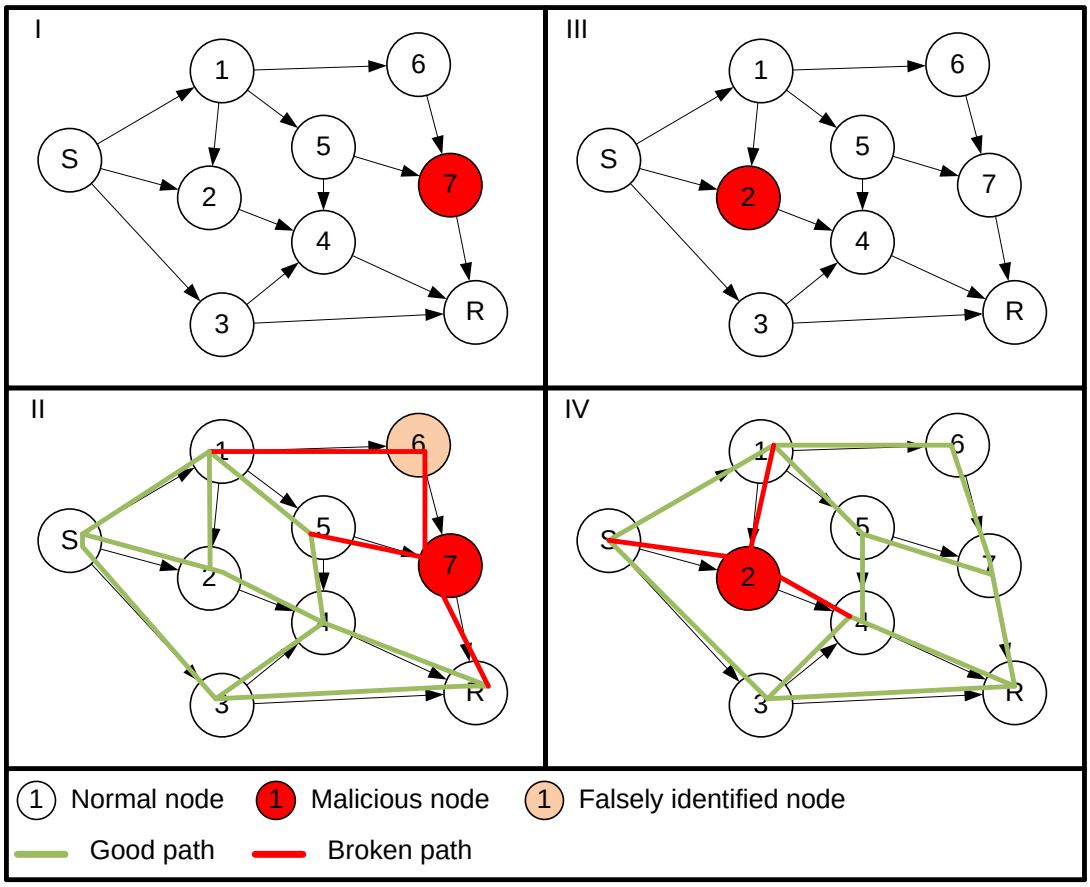



Figure 5.10: Example: Explanation of why a false positive is detected.

In Frame I the red node 7 represents a malicious packet dropping node. This malicious node causes the path {S,1,6,7,R} to be broken as shown in Frame II. When information is send through the network the receiver can receive the packets of nodes 1, 2, 3, 4 and 5, but because node 7 drops all the packets it receives, the receiver node *R* never receives packets from node 6. This also means that the receiver never receives a checksum packet with any information about node 6. Thus when the source receives the HealthMatrix is assumes that node 6 and node 7 are both packet dropping nodes. Thus node 6 is falsely identified as a malicious packet dropping node.

Frame III and IV shows that when the position of the malicious node changes, the malicious node does not block any of the intermediate nodes. All the intermediate nodes are still part of a good path. Thus the topology does not cause the source node to identify nodes falsely as malicious packet dropping nodes.

*This chapter provided a description of the results for the different systems.The simulation parameters were given in section 5.1.*

*In section 5.2 the throughput of the networks with no malicious nodes present was described.*

*Section 5.3 described the throughput of the Normal network in the presence of a malicious node.*

*In section 5.4 the throughput for the DART security system was given. The results described the throughput in the presence of malicious packet dropping and packet pollution nodes.*

*The throughput of the PDD system was described in section 5.5. The simulation was done with the presence of malicious packet dropping and packet pollution nodes.*

*In section 5.7 the latency of Normal, DART and PDD networks were presented and discussed while section 5.8 discussed the detection of packet dropping nodes.*

# Chapter 6

# Conclusion

---

*In this chapter a summary of the research is given in section 6.1 along with the research conclusion in section 6.2. The future work is discussed in section 6.3.*

---

## 6.1 Summary of Research

In Chapter 1 an introduction to the research was given. This included background to the problem and a motivation for the research. The research problem was identified as stated below.

> The goal of this research is to see if an existing security scheme that already addresses packet pollution, can be expanded to address an additional security threat namely packet dropping.

To address the research problem the following research objectives were met:

- The implementation of a WMN.
- To investigate the effects of packet pollution and packet dropping on networks.

- To implement the DART security scheme proposed in [8].
- To expand the scheme to include packet dropping detection.

A literature study was done in Chapter 2 that provided extensive background on the relevant research topics. The topics included WMNs, NC, security in NC, the DART security scheme and the performance metrics. WMNs are wireless multi-hop networks, arranged in a mesh topology, which consist of wireless clients, routers and gateways. NC is a technique that can improve the efficiency of a network. This is accomplished by forming linear combinations of the received packets and forwarding the combined packets, thereby minimising transmissions. NC is easily implemented in WMNs because it exploits the wireless broadcast and opportunistic listening properties of WMNs. Various threats for NC were identified by [7] and this dissertation focussed on the packet pollution and packet dropping threats. The DART security scheme was chosen to be enhanced as it addressed packet pollution but not packet dropping.

Chapter 3 described the experimental design in which the system was designed. The assumptions made about the nodes and the experiment were were given. The implemented NC, DART- and PDD schemes were described and discussed.

In Chapter 4 the simulation results were presented. Throughput and latency graphs for the different systems were presented and discussed. The results were divided by the different systems, (Normal, DART and PDD) that built on one another. Two malicious nodes were simulated - packet dropping and packet pollution nodes.

In Chapter 5 the experimental model was validated by describing scenarios and assumptions for each of the systems. The computerized model was verified by verifying various core modules of the implementation. The operational validity of the model was also tested.

## 6.2 Conclusions

The research done in this dissertation included:

- Creating and setting up a WMN with NC as seen in Chapter 3.
- Investigating the effects of packet pollution and packet dropping on the throughput of the network.
- Implementing the DART security scheme proposed in [8].
- Enhancing the DART scheme to detect packet dropping in the network.

The PDD scheme detected malicious packet dropping nodes that simulated black-hole type of attacks. The detected malicious nodes were identified and removed and thus reduced packet dropping in the network.

There was a small decrease in throughput (when compared to the DART scheme) but the PDD scheme successfully detected malicious packet dropping nodes. The PDD scheme detected the malicious nodes in all the iterations giving it a 100% detection rate. There were however false positives, where a node was identified as malicious but was not. These false positives consisted of 0.061% of the detections. These detections are described in Table 5.1.

The identification of malicious packet dropping nodes can be used as a diagnostic tool. The information about which nodes are malicious can be used to identify faulty nodes in the network. These nodes can then be repaired or removed.

## 6.3 Future work

Future work includes expanding the simulation to include the layers of the Open Systems Interconnection (OSI) stack and including routing protocols. The PDD scheme can be adapted to include detection of grey-hole packet dropping attacks.

# Bibliography

[1] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE DOI - 10.1109/MCOM.2005.1509968*, vol. 43, no. 9, pp. S23–S30, 2005.

[2] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on DOI - 10.1109/18.850663*, vol. 46, no. 4, pp. 1204–1216, 2000.

[3] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Information Theory, 2003. Proceedings. IEEE International Symposium on*, 2003, pp. 442–.

[4] P. Chou and Y. Wu, "Network coding for the internet and wireless networks," *Signal Processing Magazine, IEEE DOI - 10.1109/MSP.2007.904818*, vol. 24, no. 5, pp. 77–85, 2007.

[5] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network Coding: An Instant Primer," *ACM Sigcomm Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.

[6] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on DOI - 10.1109/ISIT.2004.1365180*, 2004, p. 144.

[7] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Secure network coding for wireless

mesh networks: Threats, challenges, and directions," *Computer Communications*, vol. 32, no. 17, pp. 1790–1801, Nov. 2009.

[8] J. Dong and R. Curtmola, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proceedings of the second ACM conference on Wireless network security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 111–122.

[9] P. Ji-Yong, M.-S. Ryu, J. E. Suk, S. Dong-Min, and H.-S. Park, "An integrated security mechanism for network coding combining confidentiality and integrity," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, vol. 01, 2009, pp. 311 –314.

[10] P. Oliveira and J. Barros, "A network coding approach to secret key distribution," *Information Forensics and Security, IEEE Transactions on DOI - 10.1109/TIFS.2008.928538*, vol. 3, no. 3, pp. 414–423, 2008.

[11] H. Gauch Jr, *Scientific method in practice*. Cambridge University Press, 2002.

[12] R. Sargent, "Verification and validation of simulation models," in *Winter Simulation Conference (WSC), Proceedings of the 2010 DOI - 10.1109/WSC.2010.5679166*, 2010, pp. 166–183.

[13] E. Hossain and K. K. Leung, *Wireless Mesh Networks: architectures and protocols.* Springer, 2007.

[14] B. A. Forouzan, *Data Communications and Networking*. New York, NY, USA: McGraw-Hill, Inc., 2007.

[15] N. Cai and R. Yeung, "Secure network coding," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on DOI - 10.1109/ISIT.2002.1023595*, 2002, p. 323.

[16] N. Cai and T. Chan, "Theory of secure network coding," *Proceedings of the IEEE DOI - 10.1109/JPROC.2010.2094592*, vol. PP, no. 99, pp. 1–17, 2011.

[17] N. Cai and R. Yeung, "Secure network coding on a wiretap network," *Information Theory, IEEE Transactions on DOI - 10.1109/TIT.2010.2090197*, vol. 57, no. 1, pp. 424–435, 2011.

[18] K. Han, T. Ho, R. Koetter, M. Medard, and F. Zhao, "On network coding for security," in *Military Communications Conference, 2007. MILCOM 2007. IEEE DOI - 10.1109/MILCOM.2007.4454907*, 2007, pp. 1–6.

[19] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," *Information Theory, IEEE Transactions on*, vol. 54, no. 6, pp. 2596 –2603, 2008.

[20] H. Yao, D. Silva, S. Jaggi, and M. Langberg, "Network codes resilient to jamming and eavesdropping," in *Network Coding (NetCod), 2010 IEEE International Symposium on DOI - 10.1109/NETCOD.2010.5487669*, 2010, pp. 1–6.

[21] A. Le and A. Markopoulou, "Locating byzantine attackers in intra-session network coding using spacemac," in *Network Coding (NetCod), 2010 IEEE International Symposium on DOI - 10.1109/NETCOD.2010.5487673*, 2010, pp. 1–6.

[22] S.-Y. R. Li, Q. T. Sun, and Z. Shao, "Linear network coding: Theory and algorithms," *Proceedings of the IEEE DOI - 10.1109/JPROC.2010.2093851*, vol. 99, no. 3, pp. 372–387, 2011.

[23] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.

[24] Y. Zhang, W. Lou, W. Liu, and Y. Fang, "A secure incentive protocol for mobile ad hoc networks," *Wireless Networks*, vol. 13, pp. 569–582, 2007, 10.1007/s11276-006-6220-3. [Online]. Available: http://dx.doi.org/10.1007/s11276-006-6220-3

[25] K. Liu, J. Deng, P. Varshney, and K. Balakrishnan, "An acknowledgment-based approach for the detection of routing misbehavior in manets," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 5, pp. 536–550, 2007.

[26] A. Le and A. Markopoulou, "Tesla-based defense against pollution attacks in p2p systems with network coding," in *Network Coding (NetCod), 2011 International Symposium on DOI - 10.1109/ISNETCOD.2011.5979096*, 2011, pp. 1–7.

[27] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The tesla broadcast authentication protocol," pp. –, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.5113

[28] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, ser. MobiCom '06. New York, NY, USA: ACM, 2006, pp. 227–238. [Online]. Available: http://doi.acm.org/10.1145/1161089.1161116

[29] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, 2007.

[30] D. Goldsman and G. Tokol, "Output analysis procedures for computer simulations," in *Simulation Conference, 2000. Proceedings. Winter*, vol. 1, 2000, pp. 39–45 vol.1.

# Appendix A

# Conference Contributions

**"Investigating the Effects of Packet Dropping and Packet Pollution Attacks in Network Coding Networks"**

Presented at the Southern Africa Telecommunication Networks and Applications Conference (SATNAC), September 2012, Fancourt Hotel & Country Club Estate, George, South Africa

# Investigating the Effects of Packet Dropping and Packet Pollution Attacks in Network Coding Networks

H.L.H.C. Terblanche, M.J. Grobler and H. Marais
School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus
Tel: +27 18 299 4296, Fax: +27 18 299 1977
Email: {20569807, leenta.grobler, henri.marais}@nwu.ac.za

***Abstract*—Network Coding (NC) has introduced a new way to increase the efficiency of networks, especially Wireless Mesh Networks (WMNs). There are many security threats when using NC in WMNs. We investigate the effects packet pollution and packet dropping attacks in a NC environment. Packet pollution decreases the throughput of the network dramatically, while packet dropping does not have such a big effect. By adding security to NC the effects of these attacks can be decreased.**

***Index Terms*—MATLAB®, Network Coding, Packet Dropping, Packet Pollution, Security**

## I. Introduction

Wireless Mesh Networks (WMNs) are multi-hop networks, with a decentralised nature, that can dynamically form into mesh topologies. They offer many advantages, such as low installation costs, easy maintenance, network robustness and reliable service coverage. As more efficient protocols are developed, these type of networks are becoming more popular. One way to improve the efficiency of these protocols is something called Network Coding (NC).

NC is a method used to increase the efficiency of networks by encoding and decoding data on packet level by means of a logical XOR operation [1]. NC works well in WMNs because it can exploit the broadcast and opportunistic listening properties of this type of network.

With WMN's increase in popularity, the issue of security became prominent. Because NC relies on the combination of valid packets to generate forwarded messages, the network becomes vulnerable to packet pollution attacks. There are numerous ways in which these attacks are addressed, [2]–[4]. Most of these schemes use homomorphic hash signatures which incur a lot of overhead [5] and diminishes the advantage gained by NC.

In 2009 Dong et al. [5] proposed a security scheme for NC in WMNs that claimed not to have as much overhead as previous schemes to address packet pollution. This scheme is based on time asymmetry and checksums. Another security scheme for peer-to-peer networks proposed in [6] is also based on similar principles.

In this paper we look at the effects of packet pollution and packet dropping in networks using NC.

In section II background will be given on WMNs, NC and security in NC, along with a discussion on the DART security scheme. In section III the simulator model and experimental setup will be described. After that, in sections IV and V the results will be discussed and a conclusion will be drawn.

## II. Background

### A. Wireless Mesh Networks

WMNs are wireless multi-hop networks that consist of wireless clients, routers and gateways. They support ad-hoc networking that has self-forming, self-healing and self-organization properties.

There are three different architectures: backbone or infrastructure mesh networks, client mesh networks and hybrid mesh networks [7].

- **Backbone or Infrastructure WMNs** are the most common. Mesh routers form a backbone for mesh clients to connect to, and the routers can connect to the internet if it doubles as a gateway, as shown in Fig. 1.
- **Client WMNs** consist only of wireless clients that communicate with each other.
- **Hybrid WMNs** are a combination of client and backbone mesh networks.

The advantages of WMNs are [7]:

- Network robustness;
- Reliable service coverage;
- Easy maintenance;
- Low installation costs.

Despite these advantages, the decentralised nature and the openness of the medium, creates a security risk because there is no authentication and the network is vulnerable to eavesdropping. NC can address these problems to a certain degree, because the packets are encoded, and an eavesdropper must first get enough packets before it can get any real information.

### B. Network Coding

As stated in the introduction, NC can improve the efficiency of a network. This is accomplished by forming linear

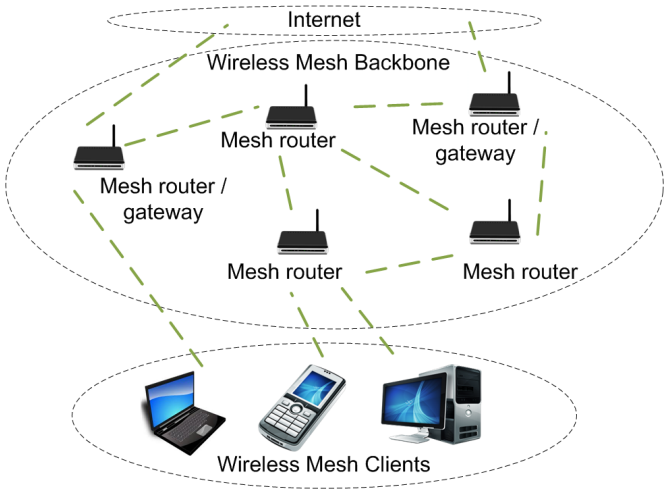



Fig. 1: Backbone Wireless Mesh Network

combinations of the received packets and forwarding only the combinations. A coding vector is attached to the new packet specifying which packets were combined to generate the encoded packet. These combined packets can easily be decoded at the receiver node. This technique was first proposed in 2000 by Ahlswede et al. in [8].

Random Linear Network Coding (RLNC) was first introduced by Ho et al. [9], where the elements in the coding vector are randomly chosen from a finite field, and the packets are combined accordingly. The most common fields that are used are the $G_{2^8}$ and $G_{2^{16}}$ fields. It was proven by Ho et al. [9] that if the field is sufficiently large the resulting encoded packet will be linearly independent from the other native and encoded packets. A native packet is a packet that has not been encoded, while an innovative packet is a packet that is linearly independent from the other received native and encoded packets. In RLNC, the intermediate/forwarder nodes can decide which packets to combine, using the random coding vectors, before forwarding them. When enough innovative packets have arrived at the receiver it is easy to decode them using Gaussian elimination. With RLNC, there is no need for a centralised control mechanism and encoded packets do not have to arrive in sequence at the receiver.

The advantages of NC include robustness, maximizing the throughput, increasing the efficiency and minimizing the delay of the network [1], [10].

RLNC works well in WMNs as shown by Ho et al. in [11]. Although using NC with WMNs has many benefits it also introduces vulnerabilities to the network that has to be addressed.

### C. Security in Network Coding

Security in NC was first addressed by [12] in 2002. There are three approaches to NC security [13]:

- Computational - when it is computationally infeasible to break the system;
- Physical - when using physical properties to prevent or detect attacks; and

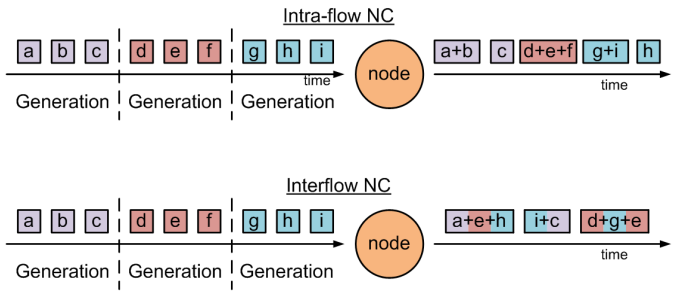



Fig. 2: Intra-flow and Inter-flow Network Coding

- Information Theoretic - determines the maximum transmission rate necessary to make it impossible to break the system.

To ensure that the decoding process is optimised, the information that has to be sent, is divided into chunks of $n$ packets, called generations. There are two general approaches for NC in WMNs - *intra-flow NC* and *inter-flow NC*. Intra-flow NC combines packets within individual generations and inter-flow NC combines packets across different generations as shown in Fig. 2.

In [14], Dong analysed the threats for both general approaches to NC. They divided each approach into different components. The components for intra-flow NC were *forwarding node selection and rate assignment*, *data packet forwarding* and *acknowledgement delivery*.

The threats identified were:

**For forwarding node selection and rate assignment:**

- Link quality falsification or modification;
- Wormholes.

**For data packet forwarding:**

- Packet pollution;
- Packet dropping.

**For acknowledgement delivery:**

- ACK injection or modification;
- ACK dropping;
- ACK delay.

In this paper we focus on the *data packet forwarding* component of intra-flow NC. For this study the other identified threats were not taken into account. Thus, we focus on packet pollution and packet dropping. Of the two, packet pollution has the highest impact on the throughput of a network using NC.

*Packet Pollution:* Packet pollution occurs when a malicious node injects corrupt packets into the network. Packet pollution can also occur accidentally when the packets get corrupt because of channel errors. Packet pollution attacks can cause a significant decrease in the throughput of the network.

Because packets are combined, the pollution can spread quickly down the network.

The packet pollution security threat has been addressed by other security schemes [2]–[4]. These schemes all incur high overhead. The DART scheme proposed by [5] addresses

packet pollution but does not incur as much overhead as other schemes.

*Packet Dropping:* The packets can be dropped by a malicious node or the packets can be lost because of channel errors. Packet dropping attacks where the malicious node drops all the packets received, are known as black hole attacks. With grey hole attacks packets are dropped at random intervals. Grey hole attacks are usually more difficult to detect than black hole attacks. Packet dropping attacks are not as severe as packet pollution but still has a negative effect on the throughput of the network.

### D. DART Security Scheme

When implementing the DART security scheme the source divides the data into generations, and encoded packets are generated from the active generation and sent into the network. The source also generates checksum packets, at constant time intervals, for the active generation and broadcasts it to all the forwarder nodes.

The encoded packets that arrive at the forwarder nodes are stored in an unverified queue. These packets are then verified upon reception of a checksum packet and only packets that arrived at the node before the checksum was created are verified. Packets that pass verification are then put into the verified queue and packets that failed are discarded.

Packets that arrive at the receiver node also go through the same process as the forwarder nodes but the verified packets are stored in a decoding matrix. If the receiver node has enough innovative packets of the current generation, it decodes the generation and sends an acknowledgement packet to the sender to begin with the next generation.

If a malicious node injects a polluted packet into the network, it will not propagate further than one hop, because of the verification of packets at each node. If an attacker can produce a polluted packet that meets the current checksum's requirements, the packet still will not be verified because the node only verifies packets that were received before the checksum was created. The polluted packet will thus be verified by the next different random checksum and be discarded.

This security scheme was implemented in the Glomosim environment [15]. This environment is outdated, therefore it was decided to do a custom implementation in MATLAB®.

## III. Methodology

### A. Network Model

Consider a wireless network with one source node, one receiver node and *(0 - n)* forwarder nodes. Each node can send and receive packets. The network uses RLNC and only decodes packets at the receiver. It is a unicast network with the source connected to the receiver via the forwarder nodes as seen in Fig. 3.

### B. Simulator Design/Model

*1) Basic Network:* Consider the simple network in Fig. 3. If the source wants to transmit the image in Fig. 4 to the receiver node, it has to perform the following steps. The source divides the file into generations. It creates an encoded packet from the current generation and sends it to the forwarder node. When the forwarder node receives the encoded packet, it stores it in a queue until it is ready to send. The forwarder node sends the packet to the next forwarder node where it also gets stored in a queue. The packet is then sent to the receiver. Upon reception the packet gets tested for linear independence and is stored in a decoding matrix. When the receiver node has received enough linearly independent packets of the current active generation, it decodes them using Gaussian elimination. After a generation has been decoded and verified the receiver sends an acknowledgement to the source node. Upon the reception of the acknowledgement packet, the source node begins sending packets from the next generation. This continues until there are no more generations left at the source node. If there were no faults the image should not be corrupt after decoding.

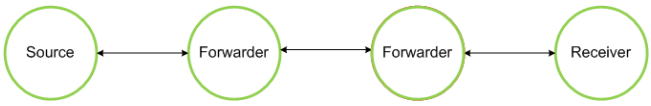



Fig. 3: Example of the network simulated


Fig. 4: The image at the source

*2) Network with security enhancements:* In the case of added security the source node generates a checksum packet for the current generation at fixed time intervals. The checksum packet is broadcast to all the first hop forwarder nodes. The forwarder and receiver nodes then use the checksum to verify packets. If they pass verification they are forwarded or stored in the decoding matrix, otherwise they are discarded.

When each generation is decoded it is verified by end-to-end authentication. If a decoded generation is corrupt no acknowledgement packet is sent and the receiver restarts packet collection.

### C. Metrics

**Throughput** - The throughput of each network was measured to see what the effect of each attack was. The equation used is :

$$\text{Throughput} = \frac{\text{Total relevant encoded packets received}}{\text{Time to send file}} \quad (1)$$

**Packet Loss** - Lost packets are defined as encoded packets that are dropped or polluted by malicious nodes. Packet loss is determined by measuring the total relevant encoded packets received in a malicious environment, to the amount of relevant encoded packets received without any malicious node intervention.


Fig. 5: The image at the receiver after decoding


Fig. 6: Network with a malicious node present

### D. Experimental Setup

The simulation was done in MATLAB® and time was measured in artificial ticks. The field used for NC was $G_{2^8}$, the generation size was 32 packets and the packet size was 1500 Bytes. The queue size for all the forwarder and receiver nodes was 10. For the security scheme the default setup in Dong [5] was used, with a checksum packet sent every 6 packets.

The image in Fig. 4 was sent through the network in Fig. 3 to establish a baseline of the throughput without any malicious nodes. The size of the image was 1.17 MB and was divided into 26 generations.

For an attacker scenario, the image was sent through the network in Fig. 6 with a malicious node present. A malicious node is a forwarder node that was modified to corrupt or drop some of the packets it forwards.

## IV. Results

The effects of packet pollution and packet dropping were investigated for a simple network topology. The following graphs show the throughput and packet loss in networks with NC alone and then with NC with added security.

The image in Fig. 4 was sent through the network with only one polluted packet. When the data was decoded at the receiver the image was corrupt as seen in Fig. 5. From the fact that the picture was corrupt, when it arrived at the receiver, it was derived that the effective throughput was 0.

*1) Baselines:* Fig. 7 shows the throughput for the network in Fig. 3. Implementation of the security scheme in the same network causes a reduction in throughput as seen in Fig. 7. Upon further analysis of the graph it was seen that the reduction is about 30%.

*2) No Security:* Fig. 8 shows the throughput in cases of packet pollution with end-to-end verification. The graph shows that if only a single generation is polluted the throughput falls from the 89% of the baseline to 86%. The throughput decreases further as more generations are polluted. If all the generations have been polluted, by one packet per generation, then the throughput falls from the 89% baseline to 47%. It is assumed that a generation is only polluted once, meaning when it is sent the second time no pollution occurs.

In Fig. 8 the throughput for different cases of packet dropping is shown.

Case 1: 1 packet (3%) of a generation is dropped.

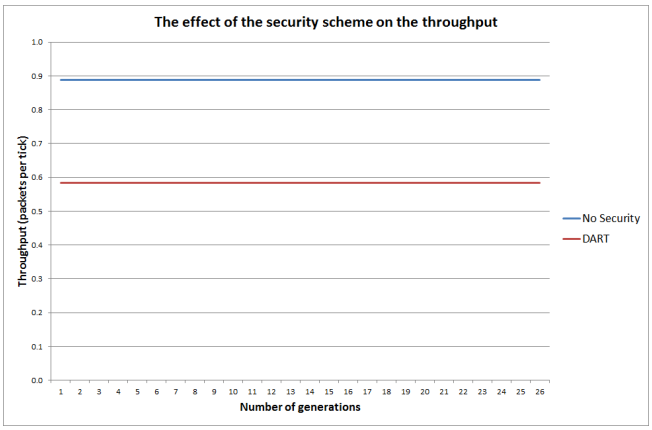



Fig. 7: Throughput of the network in Fig. 3

Case 2: 3 packets (10%) of a generation are dropped.

Case 3: 8 packets (25%) of a generation are dropped.

In case 1 the throughput decreases only slightly. If a packet were dropped in every generation, the throughput would decrease from 89% to 86%. In case 2, the throughput at 26 generations dropped from 89% to 82%. In case 3 the throughput decreased from 89% to 74% at 26 generations.

As can be seen from Fig. 8 and Fig. 10, packet pollution has a much larger effect on the throughput, of the network, than packet dropping.

*3) DART Security:* For attacks on the network with the DART security it was seen that the throughput of the network decreases only slightly as seen in Fig. 9. The same cases as above were applied for packet dropping. In case 1 the throughput decreases only slightly. If a packet was dropped in every generation the throughput would decrease from 59% to 57%. In case 2 the throughput at 26 generations dropped from 59% to 55%. In case 3 the throughput decreased from 59% to 51% at 26 generations. The packet dropping and packet pollution attacks did not have such a big effect on the packet loss, because of the nature of the security scheme. This can be seen in Fig. 11. The packet loss in the case for packet dropping and packet pollution, of 1 packet per generation, is the same. This is because the polluted packets are dropped when detected.

## V. Conclusion

In this paper, we studied the effects of packet pollution and packet dropping, on a simple network using NC. We saw that if there is only one pollution attacker present in the network, the throughput decreases dramatically if each generation is polluted. In the case of a packet dropping attacker, the effect is not as great as with packet pollution. Packet pollution is the biggest threat to networks using NC. This can be seen in Fig. 8 where the packet pollution attack has a much greater effect on the throughput than the packet dropping attack, where more packets are dropped.

By implementing the DART security scheme the throughput was reduced, but no data corruption occurred due to packet pollution. The throughput, in the case where every generation

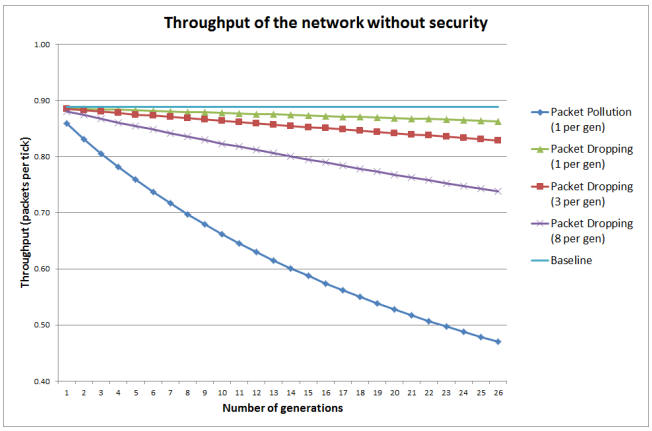



Fig. 8: Throughput of the network in Fig. 6 without security

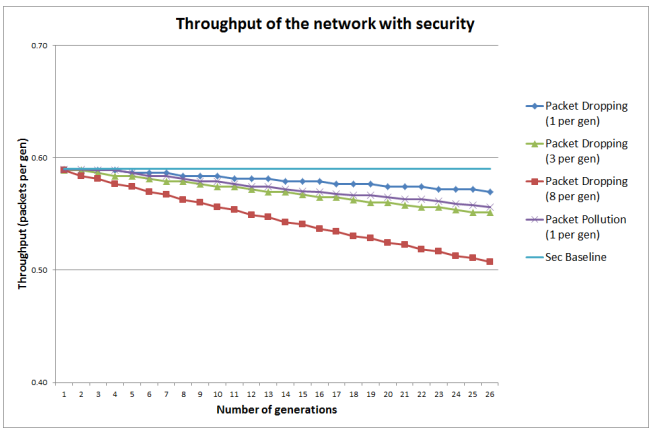



Fig. 9: Throughput of the network in Fig. 6 with DART security scheme

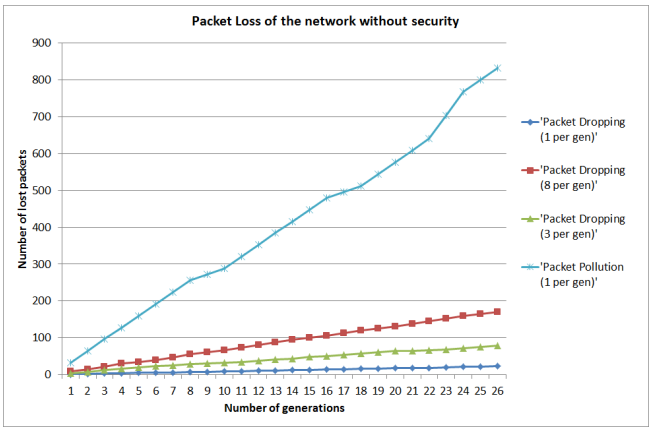



Fig. 10: Packet Loss of the network in Fig. 6 without security

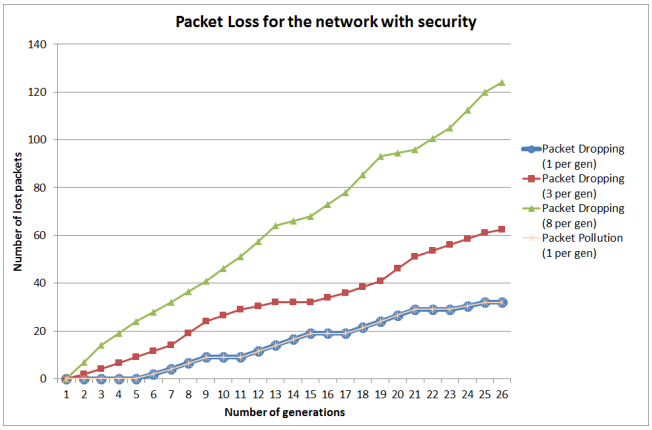



Fig. 11: Packet Loss of the network in Fig. 6 with DART security scheme

is polluted as seen in Fig. 8, is also lower than the effective throughput of the security baseline as seen in Fig. 7. Packet loss is also not as significant in the case of security as can be seen in Fig. 11.

## VI. Future Work

Future work will include expanding the network and investigating how the security scheme, that already addresses packet pollution, can be improved to address packet dropping. This includes determining how the checksum packet can be used along with stored packets at forwarder nodes to recreate dropped packets in the network.

## VII. Acknowledgements

This work was completed with funding from the Telkom Centre of Excellence at the NWU, Potchefstroom Campus.

## References

[1] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network Coding: An Instant Primer," *ACM Sigcomm Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.

[2] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on DOI - 10.1109/ISIT.2004.1365180*, 2004, p. 144.

[3] A. Le and A. Markopoulou, "Locating byzantine attackers in intra-session network coding using spacemac," in *Network Coding (NetCod), 2010 IEEE International Symposium on DOI - 10.1109/NETCOD.2010.5487673*, 2010, pp. 1–6.

[4] S.-Y. R. Li, Q. T. Sun, and Z. Shao, "Linear network coding: Theory and algorithms," *Proceedings of the IEEE DOI - 10.1109/JPROC.2010.2093851*, vol. 99, no. 3, pp. 372–387, 2011.

[5] J. Dong and R. Curtmola, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proceedings of the second ACM conference on Wireless network security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 111–122.

[6] A. Le and A. Markopoulou, "Tesla-based defense against pollution attacks in p2p systems with network coding," in *Network Coding (NetCod), 2011 International Symposium on DOI - 10.1109/ISNETCOD.2011.5979096*, 2011, pp. 1–7.

[7] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE DOI - 10.1109/MCOM.2005.1509968*, vol. 43, no. 9, pp. S23–S30, 2005.

[8] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on DOI - 10.1109/18.850663*, vol. 46, no. 4, pp. 1204–1216, 2000.

[9] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Information Theory, 2003. Proceedings. IEEE International Symposium on*, 2003, pp. 442–.

[10] P. Chou and Y. Wu, "Network coding for the internet and

wireless networks," *Signal Processing Magazine, IEEE DOI - 10.1109/MSP.2007.904818*, vol. 24, no. 5, pp. 77–85, 2007.
[11] T. Ho, B. Leong, M. Medard, R. Koetter, Y.-H. Chang, and M. Effros, "On the utility of network coding in dynamic environments," in *Wireless Ad-Hoc Networks, 2004 International Workshop on*, 2004, pp. 196–200.
[12] N. Cai and R. Yeung, "Secure network coding," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on DOI - 10.1109/ISIT.2002.1023595*, 2002, p. 323.
[13] N. Cai and T. Chan, "Theory of secure network coding," *Proceedings of the IEEE DOI - 10.1109/JPROC.2010.2094592*, vol. PP, no. 99, pp. 1–17, 2011.
[14] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Secure network coding for wireless mesh networks: Threats, challenges, and directions," *Computer Communications*, vol. 32, no. 17, pp. 1790–1801, Nov. 2009.
[15] Glomosim. [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/

**H.L.H.C. Terblanche** is a Telkom CoE student studying towards a Masters degree in Computer and Electronic Engineering at the NWU. She received her B.Eng in Computer and Electronic Engineering in 2010 from the NWU. Her current research interests are wireless mesh networks, network coding and security.