MSc (Res)

Enhancing Bio-inspired Intrusion Response in Ad-hoc Networks

Maryamosadat Kazemitabar A.

August 2013

ABSTRACT

Practical applications of Ad-hoc networks are developing everyday and safeguarding their security is becoming more important. Because of their specific qualities, ad-hoc networks require an anomaly detection system that adapts to its changing behaviour quickly. Bio-inspired algorithms provide dynamic, adaptive, real-time methods of intrusion detection and particularly in initiating a response. A key component of bio-inspired response methods is the use of feedback from the network to better adapt their response to the specific attack and the type of network at hand.

However, calculating an appropriate length of time at which to provide feedback is crucial - premature feedback or delayed feedback from the network can have adverse effects on the attack mitigation process. The antigen-degeneracy response selection algorithm (Schaust & Szczerbicka, 2011) is one of the few bio-inspired algorithms for selecting the appropriate response for misbehavior that considers network performance and adapts to the network. The main drawback of this algorithm is that it has no measure of the amount of time to wait before it can take performance measurements (feedback) from the network. In this thesis, we attempt to develop an understanding of the length of time required before feedback is provided in a range of types of ad-hoc network that have been subject of an attack, in order that future development of bio-inspired intrusion detection algorithms can be enhanced.

Aiming toward an adaptive timer, we discuss that ad-hoc networks can be divided into Wireless Sensor Network (WSN), Wireless Personal Area Network (WPAN) and Spontaneously Networked Users (SNU). We use ns2 to simulate these three different types of ad-hoc networks, each of which is analysed for changes in its throughput after an attack is responded to, in order to calculate the corresponding feedback time. The feedback time in this case is the time it takes for the network to stabilise. Feedback time is not only essential to bio-inspired intrusion response methods, but can also be used in network applications where a stable network reading is required, e.g. security monitoring and motion tracking.

Interestingly, we found that the network feedback time does not vary greatly between the different types of networks, but it was calculated to be less than half of what Schaust and Szczerbicka used in their algorithm.

Keywords

Security, Bio-inspired, Immune System, Intrusion Detection, Intrusion Response, Ad-hoc Networks

DECLARATION

I hereby declare that this thesis together with work contained herein was produced entirely by myself, and contains no materials that have been accepted for the award of any other degree or diploma in any university. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due acknowledgment to others has been made.

Maryamosadat Kazemitabar

ACKNOWLEDGEMENTS

I would like to take the opportunity to thank my supervisors Emma Hart and Bill Buchanan, for their support and guidance throughout the course of my research. Furthermore, I would like to give my appreciation to my husband for providing advice and support throughout my research. Finally, this research would not have happened without the love and support of my parents and family.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Context

Society is becoming increasingly reliant on functions of computer systems, benefiting from e-health, e-commerce, e-government and many more systems. From a personal perspective each member of society carries and interacts with many computing devices. Considering the sensitivity of computer system applications in our lives, guarding the confidentiality, integrity and availability of these systems seems vital. Ad-hoc networks are becoming more popular and so the need for ad-hoc network security is becoming more apparent.

Security mechanisms can be divided into attack prevention, detection and recovery (Stallings, 2009). In order to remove the vulnerabilities of attack prevention methods, intrusion detection can be used as a second wall of defence. Generally, intrusion detection systems (IDS) can be categorised into anomaly detection and misuse detection based on their detection methods. One disadvantage of misuse detection is that it can only detect previously known attacks, based on their signatures; whereas in anomaly detection, attacks are detected through deviation from *normal behaviour* (Timmis, Lemos, Ayara, & Duncan, 2002). Although anomaly detection can have its drawbacks and producing too many false positives is undesirable.

The biological immune system may be considered a rich source of inspiration for anomaly detection and recent advancements in theoretical immunology endorse this idea. Biological systems are self-healing, self-maintaining, self-organising systems and are self-aware in the way that they have an internal self-image and interact with their environment accordingly. The successful and collaborative operation of biological systems based on a limited set of rules and with global intelligence which is larger than superposition of individuals (Dressler & Akan O, 2010) can contribute to many applications especially in distributed networks.

Bio-inspired methods, usually try to detect anomaly by building an adapting self-image. This is performed by extracting elements of the system that represent the system behavior most and then creating a self-image using one of many bio-inspired algorithms. Ad-hoc networks especially require an anomaly detection system that adapts to its changing behaviour quickly and distinguishes the normal changes in the network from abnormal anomalies that might be the result of an intrusion or misbehavior. The various proposed methods of bio-inspired intrusion detection are investigated in section3.2. Intrusion response, which is the goal of any intrusion detection algorithm, is then outlined briefly in section 2.4.2. Part of any bio-inspired intrusion response algorithm is taking measurements from the network to provide feedback to the algorithm. Network feedback time is a sensitive issue and can affect the outcome of the response algorithm profoundly. A measurement taken from the network prematurely after a response, while it is in the process of rerouting, does not show the real effects of that response, also waiting too long to measure the effects of a response can hinder the workings of a network for un-needed lengths of time. Different types of network with different sizes can behave differently, and so as a case study, we investigate the shutdown intrusion response in three types of network: WPAN (Wireless Personal Area Network), SNU (Spontaneously Networked Users) and WSN (Wireless Sensor Network).

Although having a feedback time is essential to bio-inspired algorithms, it can be used in other ad-hoc network applications, where stability is an important issue and network measurements have an impact on the outcome. An application could be in ad-hoc network stability analysis, in which we can determine the fault tolerance of different types of ad-hoc networks against node failures. Another similar application is the inspection of the ad-hoc network resilience to the set of attacks, in which nodes are destroyed such as DOS attacks.

1.2 Contributions

The main contributions of this research are:

- Calculation of response feedback time for three main types of ad-hoc network: WSN, WPAN and SNU,
- Development of a method of calculating response time that could be used to improve the bio-inspired *antigen-receptor degeneracy* method (Schaust & Szczerbicka, 2011) for selecting the most suitable intrusion response for the current state of the ad-hoc network,
- Enhancing the implementation of the AODV routing protocol in ns2 simulator to support the node shutdown feature.
- Simulation of an attack response in different types of ad-hoc networks.

This work can also benefit:

- Methods that require feedback after a change to the ad-hoc network either bio-inspired or non-bio-inspired.
- Stability analysis of ad-hoc networks in case of node failure.
- Analysis of ad-hoc network resilience against the set of attacks, in which nodes are destroyed such as DoS (Denial-of-Service) attacks.

1.3 Overview

Ad-hoc networks have various applications from military to rescue to monitoring and in many cases they provide information that influences the safety of human beings. Having a secure and reliable ad-hoc network where all nodes take the same share of the work load is crucial to the application using the network as an infrastructure. A system that detects abnormalities and malicious activity and responds accordingly puts up a line of defence for that application which would have deemed extremely vulnerable without.

An autonomous bio-inspired response selection approach seems to be the promising solution for ad-hoc networks security. The *antigen-receptor degeneracy* method by Schaust and Szczerbicka (2011) is the prevalent bio-inspired intrusion response method. We aim to overcome the main weakness of this algorithm. In this research we try to work out the time after a shutdown response to an attack in an ad-hoc network after which the network readings are dependable and show the effects of the response correctly.

We are going to address the following research questions:

- To what extent is it possible to predict the appropriate time-lag in an ad-hoc network following an attack before feedback should be provided to a response algorithm?
- To what extent does the feedback vary with the type and size of network and the type of attack?

Having answered these questions, it should be possible to design better bio-inspired algorithms for intrusion response in ad-hoc networks.

1.4 Scope

We investigate the different immune inspired intrusion detection and response algorithms, and try to enhance intrusion response by calculating the feedback time used to evaluate response effects on the network. In the implementation part of the research we focus on AODV as the best overall performing routing protocol in ad-hoc networks. We consider three main categories of ad-hoc network: WSNs, SNUs and WPANs; and by simulating these networks we investigate the effect of ‘node shutdown’ as a response to misbehaviour on the network and calculate the time to wait until the network reaches a state of stability where it either cannot recover or is nearly recovered.

The aim of the research is to calculate feedback time in three main categories of ad-hoc networks, following a node shutdown response to misbehaviour, and the objectives are:

- Simulate a shutdown response to misbehaviour in the different categories of ad-hoc networks: WSNs, SNUs and WPANs, using an appropriate simulator.
- Conduct a series of experiments in order to calculate the time taken for the network to return to an acceptable level of throughput following an attack response.
- Analyse the results over a range of scenarios with different percentages of misbehaviour.

1.5 Thesis Structure

This thesis is organised as follows:

- **Chapter 2** introduces background information on ad-hoc network security and discusses categories of ad-hoc networks, Ad-hoc routing protocols, attacks against ad-hoc networks and intrusion detection and response in ad-hoc networks.
- **Chapter 3** presents the literature review and related work on bio-inspired intrusion detection and response in ad-hoc networks. In particular, an energy aware immune-inspired misbehaviour detection method, which is the predecessor of the *antigen-receptor degeneracy* technique, is analysed.
- **Chapter 4** describes the enhancement of the *antigen-receptor degeneracy* intrusion response technique by calculating the response feedback time and explains the implementation and simulation details.
- **Chapter 5** explains the evaluation methodology and the experiment design. It then illustrates and discusses the experiments results.
- **Chapter** 6 finally concludes this thesis by summarising the contributions and discussing future work.

Chapter 2

Technology Overview

2.1 Introduction

In this chapter we introduce ad-hoc networks as the underlying technology for our work. The main categories of ad-hoc networks which are then used in our experiments are described. Routing protocols in ad-hoc networks are discussed and our choice of protocol explained. The challenges and specific needs of ad-hoc networks like adaptability, limited computation and transmission resources are explained so that we can understand the reasons behind solutions devised specifically for ad-hoc networks. Finally the security of ad-hoc network is considered and attacks and vulnerabilities of these networks are discussed.

2.2 Ad-hoc Networks

Ad-hoc networking or multi-hop relaying is not a new concept and roots back to 500 B.C. to the time of Darius I, the king of Persia, who devised a communication method between his capital and the remote provinces of his empire (Byers & Bourgoin, 2005-2006); he used a line of men positioned on high structure shouting. This method was 25 times faster than normal messengers available at that time. Nodes in an ad-hoc network cooperate to communicate beyond their wireless transmission range; this and the reduced amount of energy in transmitting data from the source to a destination using a multi-hop method is the basis for wireless ad-hoc networks. Together with situations where it is not possible to have any other kind of infrastructure network between the nodes, like setting up the infrastructure would either be too time consuming or expensive, or where the terrain is inaccessible, the need for developing such networks with the reassurance of security is clear.

Wireless ad-hoc networks can have many different uses in our everyday lives for example, traffic data that is collected and distributed by vehicle navigation systems, or researchers in a conference sharing resources, or the shared use of hardware and software on nodes available to different people, or the detection of fire using sensors in a building; It can also have uses in military operations, to sense activity in enemy grounds, or disaster areas where communication infrastructure have been destroyed, although like many other developments these networks were initially devised for military uses.

The PRNET project was one of the first distributed wireless multi-hop communication systems. This Defence Advanced Research Projects Agency (DARPA) funded project initiated in 1973, was a robust, reliable, operational experimental network (Jubin & Tornow, 1987). The nodes in this network were mobile, and had a single communications channel, which gave the benefit of simpler channel management techniques and the ease of supporting mobility. The main challenges in PRNET were error and flow control over the wireless link, handling link breaks, distributed channel sharing and processing and storing of information in each node.

Modern ad-hoc networks are multi-hop peer to peer networks in which each node manages both the routing and processing of the data in a store and forward way (Hubaux, Buttya´n, & Capkun, 2001). Data is transmitted to its destination through a series of hops between intermediate nodes, and so reducing the required energy for the data exchange according to the inverse square law. Nodes may be mobile or stationary, either way they join and leave the network randomly. A mobile ad-hoc network is an infrastructure-less wireless network that dynamically changes topology in an unpredictable manner since nodes are free to move (Luo, Ye, Lu, & Zhang, 2004). In Latin, *ad-hoc* literally means "for this," further meaning "for this purpose only" and thus usually temporary (IETF Working Group Charter, 1997).

Ad-hoc networks are defined as infrastructure-less networks, as opposed to cellular networks, in which base stations manage the resources and routing of the data, and nodes only communicate with the nearest base station. Base stations themselves manage the routing of the data independently through wired/wireless mediums. Nodes in an ad-hoc network are more complex, compared to nodes in a cellular network, as they have to perform the routing of the data as well as performing processing operations.

Thus Ad-hoc networks are formally defined by (Luo, Ye, Lu, & Zhang, 2004) as: "An Autonomous system of mobile hosts connected by wireless links, the union of which forms a communication network modelled in the form of an arbitrary communication graph."

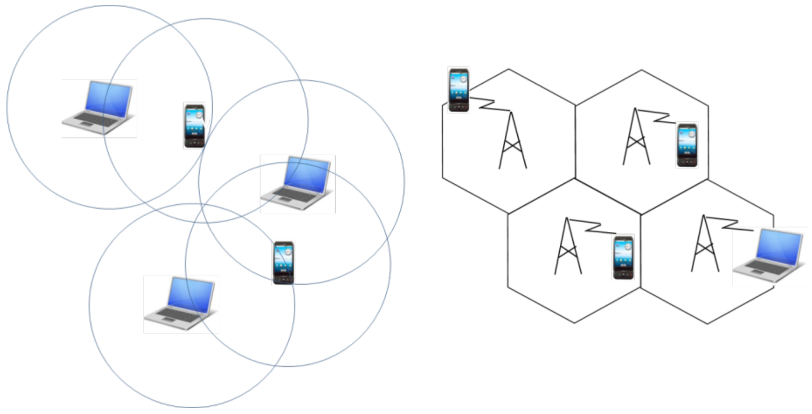


Figure 1- A diagram showing the differences between cellular and ad-hoc architecture, and a hybrid one

An intermediate solution between cellular and ad-hoc is hybrid communication networks, the proposed solutions like Multi-Hop Cellular Networks (MCNs) (Ananthapadmanabha, Manoj, & Murthy, 2001) and self-organising packet radio ad-hoc networks with overlay (SOPRANO) or integrated cellular ad-hoc relay (iCAR) networks (Wu, Qiao, De, & Tonguz, 2001); these show an improvement in the capacity of the system significantly. In these networks mobile ad-hoc nodes can function in the presence of infrastructure and the architecture combines the benefits of cellular and ad-hoc networks. Although this architecture seems beneficial the ad-hoc networks' unsolved problems keep it from being commercially deployed: "QoS provisioning and real-time applications, pricing, cooperative functioning, energy-efficient relaying, load balancing, and support for multicast traffic” (Hofmeyr, 2000).

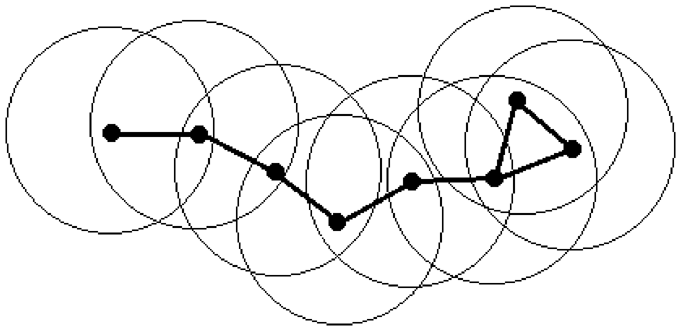


Figure 2- A diagram showing a typical ad-hoc network as a graph

In an ad-hoc network where infrastructures like base stations, routers and switches don’t exist, wireless nodes connect using the neighbouring nodes as connecting nodes. In Figure 2a typical ad-hoc network is depicted. Each node is shown using a small circle, with its transmission range as the bigger circle. Nodes located within the transmission range of other nodes are able to send and receive data from that node and therefore neighbours, this is illustrated by connecting the two nodes in the graph with a line.

2.3 Categories of Ad-hoc Networks

Wireless ad-hoc networks can be categorized in many ways, according to the area they cover like PAN, LAN, MAN, WAN or RANs (Cordeiro & Agrawal, 2006), according to their applications like military applications, collaborative and distributed computing, emergency operations, wireless mesh networks, wireless sensor networks, Hybrid wireless networks (Murthy & Manoj, 2004), but generally the below classification (Dressler, 2006) fulfils our purposes.

- **WSN**- Wireless sensor networks are a type of ad-hoc network in which the devices are cheap, small and stationary, and able to sense some physical characteristic of the environment and transmit it to a base station or another node. The nodes have limited computational and communication capabilities and (rechargeable) batteries are the main source of power. These devices could be used in battlefield surveillance of enemy territory where they could be scattered from a plane and activities could be detected and monitored. Another application area is the health of large bridges and structures; bio sensing, machine prognosis and environmental monitoring could be commercial application fields (Cordeiro & Agrawal, 2006).

- **WPAN**- Wireless Personal Area Networks are localized networks in which nodes are associated with a given person and aim at enabling wireless communication between the devices a user is carrying. The nodes vary between a few centimetres to a few meters in range, and usually utilize a single hop point to point wireless link to exchange data. Bluetooth and infrared data association and ZigBee are the main technologies used to implement WPANs. PDAs and mobile phones are the main platforms for WPAN.

- **SNU**- Spontaneously networked users are networks formed on demand to exchange data, have multimedia communications or use entertainment applications like multiuser games. The nodes in this network are usually devices with higher processing power and greater bandwidth capabilities, like laptop computers and PDAs. Dynamic multi-user games and groupware and collaborative tools are some of the applications in this category of ad-hoc networks (Dressler, 2006). Table 2-1 describes the differences in the three categories of ad-hoc networks.

Table 2-1- Differences in the three categories of ad-hoc networks; WSN, WPAN and SUN (Dressler, 2006)

Requirement	WSN	WPAN	SNU
Energy	High	high	low
Storage	High	medium	low
Mobility	Limited	low	high
Heterogeneity	Limited	high	low
Processing Power	High	medium	low
Group Formation	No	high	high
Surrounding Conditions	High	low	low
Security	Low	high	medium
Varying user demands	None	medium	high

2.4 Routing Protocols in Ad-hoc Networks

Routing in ad-hoc networks is different from traditional fixed routing protocols; it has to be capable of dealing with the dynamic nature of these networks. As mentioned before ad-hoc networks can be implemented in many different areas such as military, emergency, conferencing and sensor applications. Based on the implemented category of ad-hoc network the appropriate routing protocol is used. For example in a military application a low probability of detection and interception is a key factor, while in a sensor application minimum energy usage is most important (Kuosmanen, 2002).

There are two main types of routing protocols in ad-hoc networks, proactive or table driven and reactive or on-demand routing protocols. Proactive routing protocols need to maintain an up-to-date list of destinations and their routes which would mean high data maintenance and slower reaction to changes in the network, although packet forwarding is faster as the route is already present. Destination Sequenced Distance Vector (DSDV) is an example of a proactive routing protocol. Reactive routing protocols determine the route when needed, and have smaller route discovery overheads. These protocols use a flooding approach for route discovery, which is more time consuming and could lead to network clogging. The reactive methods of routing are more dynamic and have less data maintenance overhead. Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA) and Ad-hoc On Demand Distance Vector Routing (AODV) are the main protocols of this type (Zou, Ramamurthy, & Magliveras, 2002).

AODV is experimentally shown to have the overall better performance over DSR and TORA (Gupta, Sadawarti, & Verma, 2010), while each protocol has advantages and disadvantages of its own and better for some applications over others. DSR works better with applications of moderate mobility, having lower overhead it is better suited to networks with lower bandwidth and power. TORA supports multiple routes and multicasting and so it is better suited to larger networks of dense population. Below is a table showing a comparison between these three routing protocols in terms of network metrics with “1” rated as best and “4” worst numerical value.

The AODV routing protocol was chosen for experiments in our project because of its overall prevalence amongst other reactive routing protocols. As seen in Table 2-2 AODV acts moderately in all aspects and performs better in the throughput and packet-drop measurements. AODV is also a popular choice for bio-inspired solutions for intrusion detection and response and can be seen in (Sahu & Shandilya, 2010).

Table 2-2 –A numerical comparison of three reactive routing protocols (Gupta, Sadawarti, & Verma, 2010)

Metrics	AODV	DSR	TORA
Scalability	2	3	1
Delay	3	2	4
Routing overhead	2	1	3
Drop packet	1	2	3
Throughput	1	2	4
Dynamic adaptability	2	3	1
Energy conservation	2	1	3

2.5 Challenges of Ad-hoc networks

The special properties of ad-hoc networks brings about challenges specific to them, while these challenges are the main basis for security threats, they are also what distinguishes them from structured networks, making them useful for their application. In the next section explain how these properties can be compromised and misused to attack ad-hoc networks briefly, but here we look into the properties of ad-hoc networks to better understand them. The main challenges in ad-hoc networks are as follows (Morais & Agrawal, 2006), (Murthy & Manoj, 2004) and (Brutch & Ko, 2003):

Nodes in ad-hoc networks usually rely on exhaustible forms of energy, with the device’s network interface being the largest consumer of energy. This provides the basis for a denial of service attack, by using up the node’s energies and flooding the network or more sensitive nodes with requests. Also this constraint brings selfish behaviour of nodes, dropping packets implying not enough battery to perform a task.

Nodes in an ad-hoc network use a shared wireless medium to access the network, and this in itself brings about some new constraints, specifically the limited bandwidth. Nodes have to wait before they send packets into the channel, and interference causes packet loss. A good routing protocol manages the use of the shared channel in an optimum way, using the maximum amount of resources available to it. The limit in bandwidth is also a basis for denial of service attacks, where more sensitive nodes are targeted with fake packets and other nodes are prevented from accessing the network. Also with wireless communication come other challenges such as noise, fading and interoperation: when two independently formed ad-hoc networks come close to each other physically, their interaction would be a challenge and security may be a major concern.

The dynamic nature of ad-hoc networks means nodes in ad-hoc networks are free to join, move or leave and this indicates the need for support for high scalability. Some routing protocols perform better with fewer nodes and other with more, and so using the right protocol for the application helps nodes better manage their communication tasks. Route acquisition, service location and encryption-key exchange are a few examples of tasks that will require considerable overhead as the network size grows. The free movement of nodes in the network means that so a malicious node can change its position and target a different node, and so it is very difficult to track the malicious behaviour of the compromised node (Li & Joshi, 2006).

Nodes in an ad-hoc network must work together and cooperate to perform the required task efficiently. This might not be very simple when differences in the amount and priority of data exist; e.g. a critical fire box should not be wasting its batteries to relay gaming data. Also a selfish node could compromise the working of the network by not taking its share of the workload, and not relay data it categorises as lower priority.

All the above challenges and constraints must be considered when working on ad-hoc networks, aiming for a high level of quality of service in the network. Parameters such as end-to-end delay, jitter, throughput and packet loss probability are a few measurements that should be explored in these networks.

2.6 Security of Ad-hoc Networks

In order to communicate, nodes in an ad-hoc network need communication protocols that consider the nature of ad-hoc networks, being dynamic and noting the limit in bandwidth, they provide the only infrastructure to ad-hoc networks. Traditional routing protocols have one weakness that underlies the multitude of attacks defined for ad-hoc networks; they all *trust* the neighboring node as being *self* and non-malicious (Djenouri, Khelladi, & Badache, 2005). Encryption may help the confidentiality of the data transmitted through the network, but most encryption methods have their vulnerabilities and considering the limited resources in ad-hoc networks their use is limited. Attack that target the availability of the network and the validity of the paths cannot be mitigated by encryption, as usually the control parts of packets are modified to perform such attacks; or false signals are fabricated and sent into the network. In addition, eavesdropping is a serious attack that cannot be detected (Djenouri, Khelladi, & Badache, 2005), and the attacker may collect data and then try to break the encryption.

2.6.1 Attacks on Ad-hoc Networks

Ad-hoc networks are more vulnerable compared to other networks, because of their dynamic and distributed ad-hoc topology, multicast transmission and location awareness (Chow & Yeung, 2002). Ad-hoc networks are becoming more and more useful in our everyday lives as well as advances in their commercial/military uses. In these wireless networks the overall energy usage of communication is decreased significantly according to the inverse square law, and this allows for devices with limited energy resources to enter into a cooperative network in which each node cooperates to allow for the benefit of low energy communication for the whole network.

There are a few issues regarding the quality of service and security of ad-hoc network that need to be addressed to provide for a better experience in the use of ad-hoc networks. There is no physical barrier for ad-hoc networks and so they are extremely exposed. Also as there is no central point of access in an ad-hoc network key management in encryption/authentication methods is a major problem as is also the issue of an unprotected routing in an ad-hoc network. Also the cooperation of the nodes in an ad-hoc network is important as some nodes might try to take advantage of the network in an unfair way.

In an ad-hoc network we could have *malicious* or *compromised* nodes. These nodes perform attacks with the intention of damaging the confidentiality, availability, integrity, authentication or repudiation of a node or network (Kong, Luo, Xu, Gu, Gerla, & Lu, 2002). Also *selfish* nodes are nodes that drop packets, or fail to forward them to save their own battery.The following list gives a broad view of the possible attacks in an ad-hoc network (Djenouri, Khelladi, & Badache, 2005), (Li & Joshi, 2006), (Singh & Yadav, 2007), (Komninos, Vergados, & Douligeris, 2007) and (Kong, Luo, Xu, Gu, Gerla, & Lu, 2002):

- **Routing Attacks**: Compromised or unauthorized nodes in a network can fabricate or modify routing signal in order to reroute the data through themselves and get unauthorized access to the data. These attacks include worm hole, black hole, replay, routing table poisoning and breaking the neighbour relationship.
- **Eavesdropping:** Eavesdropping is a passive attack that cannot be detected, as it does not affect the operation of the routing protocol. An intruder which has gained unauthorized access using one of the other attacks can then start to gather and analyse data in order to break the encryption applied to the data.
- **Denial of Service:** Denial of service attack tries to stop the operation of the entire network by disrupting the routing function; this attack takes advantage of the limited battery resources of the nodes in the network and tries to consume these resources prematurely. Sleep deprivation, rushing attacks (Hu, Perrig, & Johnson, 2003) and routing table overflow are examples of denial of service attack.
- **Sybil Attack:** These attacks are possible where a repudiation system exists to state the legitimacy of nodes in the network. An attacker can cause the isolation of legitimate nodes by fabricating messages to blacklist those nodes. Black mail is a kind of Sybil attack.
- **Masquerading:** During the neighbour acquisition process an attacker might act as another node, capturing messages and replaying or modifying them posing as a legitimate node.
- **Location Disclosure:** By analysing the traffic in a network, the attacker can discover the location of a node, which is sensitive in some applications.

2.6.2 Intrusion Detection and Response in Ad-hoc Networks

After looking into the structure, challenges and attacks in ad-hoc networks we would need to look for methods of counteracting these attacks and vulnerabilities. We could try and tackle this from many different angles which could be either signature based or anomaly detection. Signature based methods identify an attack using pre-identified signatures specific to that attack while anomaly based methods try and identify deviations from normal behaviour. Signature based methods benefit from having lower false positives and anomaly detection methods benefit from being able to identify unknown attacks. Signature based methods suffer from higher false negatives which is very dangerous.

Among traditional Intrusion Detection Systems (IDS), two main categories of IDS exist: Distributed and Centralised IDSs. Centralised IDSs are deployed in structured networks and on more central points of the network like switches, routers and gateways and process data real-time for the entire network. This is not possible in ad-hoc networks where central points of access don't exist and we have to rely on partial and localized data. So the IDS is usually based on the node in ad-hoc networks rather than having a network based one. We look into host based techniques for anomaly detection in ad-hoc networks.

Many traditional techniques for anomaly detection exist such as Classification Based, Clustering Based, Statistical Techniques and Information Theoretic (Chandola, Banerjee, & Kumar, 2009). Classification techniques use labelled (training) data to learn a model and test a case instance using the learnt model. In clustering based anomaly detection methods similar data instances are grouped into clusters. In this unsupervised or semi-supervised technique new data is then considered normal if it belongs to a normal cluster and anomaly if it does not. In statistical anomaly detection technique a stochastic model is fitted to the given data and normal data instances are assumed to occur in high probability regions of the model, whereas anomalies are assumed to occur in low probability regions of the model. Information Theoretic techniques assume that anomalies cause irregularities in the information content of the data which is measured using information theoretic measures like entropy or Kolomogorov complexity.

Bio-Inspired techniques can be viewed as a relatively new approach to anomaly detection and response, which provide an effective distributed, lightweight adaptive solution to anomaly detection. In short these techniques are usually based on creating a self-image using feature selection, and differentiating non-self from self, using different algorithms. The self-image is an adaptive one, meaning that it evolves with the system. This concept of an adaptive self-image along with the lightweight and distributed nature of bio-inspired methods make them great candidates for use in ad-hoc networks. We will discuss the bio-inspired approach to the detection and response to attacks in ad-hoc networks in the next section.

2.7 Conclusions

In this chapter we briefly discussed ad-hoc networks, their specific challenges and properties. We were acquainted with the three main types of ad-hoc networks: WPAN, SNU and WSN. We then looked at the security of ad-hoc networks and how bio-inspired solutions could be the best answer to the many challenges security of ad-hoc networks poses. In the next chapter we will expand the idea of immune-inspired security in ad-hoc networks and talk about the need of an adaptive autonomous response.

Chapter 3

Immune-inspired Security in Ad-hoc Networks

3.1 Introduction

Biological systems have been thriving for millions of years and have intricate ways to achieve goals both simply and effectively. Many biological systems like those of ant colonies or bee hive behaviour exist where each member of the system has a small role but a greater goal emerges out of the combined accomplishments of each member. This concept has received much attention in the ad-hoc community as it resembles the distributed nature and needs of ad-hoc networks. Ant Colony inspired routing protocols for Ad-hoc networks such as Sensor-driven Cost-aware Ant Routing (Chen, Guo, Yang, & Zhao, 2006), Improved Adaptive Routing (IAR) algorithm (Aghaei, Rahman, Gueaieb, & Saddik, 2007) and ANT Colony Optimization (ACO) method (Saleem, 2011). BeeAdHoc is also a swarm based routing protocol which has had an immune based solution developed for its security which is explained in detail below. In addition to that the most promising immune-based IDSs are discussed and reasons are given into why we chose the feedback timer as a tool to help give better performance in these algorithms.

3.2 The Biological Immune System

The biological immune system is the basis of all inspiration for Artificial Immune Systems (AIS). The purpose of the immune system is to protect the body from threats posed by toxic substances and pathogens and to do so in a way that minimises harm to the body and ensures its continued functioning. The immune system in different organisms has different complexity, and the mammalian immune system is the most intricate. The human immune system as one of the most studied immune systems has many properties; it is distributed, diverse, dynamic, error tolerant, self-protecting and adaptable. This system in the lower level is composed of single cells, each working according to a simple set of rules, but the aggregation of these cells (of which there exists many types) gives the complex behaviour we observe. This means each cell has only a small set of tasks to perform, and so we think of light weight distributed agents collaborating in a non-central configuration.

The human immune system is layered; it has three lines of defence to pathogens: the skin, the innate immune system, and the adaptive immune system. Physiological conditions like temperature and ph also contribute to the elimination process of the pathogens.

The innate immune system is the first line of defence after the skin has been ruptured. Some form of innate immune system exists in all organisms. The innate immune system does not change during the lifetime of an organism, it is inherited and so it evolves over the life time of a species. It produces a generic response to all pathogens. Its main task is to do the housekeeping of the body and as well as clearing the body of the remains of dead cells and other unwanted substances, it senses any damage to the body and the presence of bacteria or virus signatures it knows of innately. It can then initiate an inflammatory response and inform the adaptive immune system as well as trying to destroy the pathogen by itself. The time in which the innate immune system deals with the pathogens is crucial to the adaptive system, while it prepares a specific response to tackle the identified pathogen.

The adaptive immune system develops during the life time of an organism; specifically no two organisms of the same species have exactly the same adaptive immune systems. This is because the adaptive immune system responds to pathogens the organism has encountered during its lifetime and as the environment for each organism is different to the other, the repository of responses for each is different. The adaptive immune system has memory; this means if a pathogen is detected a second time the response time would be much shorter and more effective than the first. This is because the more effective cells in fighting the pathogen turn into memory cells that can be summoned to multiply next time it is activated. There is another property within the adaptive immune system which is the need for co-stimulation; this means if there is no confirmation from other cell types, the cell will not be able to implement its defence mechanism (Hofmeyr, 2000), (Forrest & Hofmeyr, 2001), (De Castro & Timmis, 2002).

The immune system has many other properties; it is distributed, diverse, dynamic, error tolerant, self protecting and adaptable. This system in the lower level is composed of single cells, each working according to a simple set of rules, but the aggregation of these cells (of which there exists many types) gives the complex behaviour we observe. This means each cell has only a small set of tasks to perform, and so we think of light weight distributed agents collaborating in a non central configuration.

Ad-hoc networks have properties that remind us of a network of cells interacting with each other. They are a set of processing nodes that work independently and without a central point of control. They also interact with each other and send messages in the form of radio waves. These resources restrained nodes need lightweight methods deployed on them to manage the interactions between them with their changing topology. This leads to the idea of using similar interaction models as those in the human immune system, for ad-hoc networks to go towards a more secure, while robust and adaptable interaction system.

3.3 Immune Based Algorithms

Up to now a number of general purpose algorithms have been developed in AIS. These are mainly based on the discrimination of self from non-self. The positive selection algorithm ensures that only useful cells are selected to be used in the immune system. By useful we mean those that actually respond to pathogens. The negative selection algorithm is responsible for eliminating detector cells that detect self as pathogen, and so after the elimination a set of cells that only recognise non-self will remain. The Idiotypic network theory is another bio-inspired network model used to simulate immune networks. The co-stimulation of B-cells and T-cells has also been reflected in immune inspired algorithms while the clonal selection algorithm is inspired from the way antibodies are produced in a stage of hyper mutation, and those antibodies with the best response to the antigen are then selected as memory antibodies (Timmis, Hone, Stibor, & Clark, 2008).

The danger theory and the following Dendritic Cell Algorithm (DCA) is more recent and has been named as a second generation algorithm. The DCA takes advantage of positive and negative feedback loops from the signals produced in the tissue regarding the safe or dangerous context of the tissue, and the detection of pathogen related signatures. It then uses all these inputs and presents a state of suspicion to pathogen or safe. These states from a set of cells performing the DCA in different time frames are accumulated to initiate a detection alarm. This algorithm has shown to be very light weight, while performing quite well in producing fewer false positives compared to other anomaly detection methods (Greensmith & Aickelin, 2008).

3.4 Immune-Inspired Intrusion Detection Systems

Ad-hoc network security has requirements that relate to the biological immune system greatly; any security solution for ad-hoc networks would need to be light-weight, adaptive and real-time. We try to investigate the main solutions proposed so far for bio-inspired methods for misbehaviour detection in ad-hoc networks in this section looking into their positive and negative points.

Sarafijanovic and Le Boudec have worked on misbehaviour detection in ad-hoc networks using immune inspired methods and have produced a series of publications: (Sarafijanovic & Le Boudec, 2004), (Le Boudec & Sarafijanovic, 2004), (Sarafijanovic & Le Boudec, 2005) and (Sarafijanovic & Le Boudec, 2005). They have mainly concentrated on algorithms inspired by negative selection throughout their research on AIS and improved their system by giving it memory and adding a negative feedback loop to make it more dynamic/adaptable.

The AIS in (Le Boudec & Sarafijanovic, 2004) is implemented on each node of the network, with the nodes communicating and distributing signals. Each node has a thymus that keeps a repository of self-antigen which is kept up-to-date and free from non-self antigen. Antigens are produced by the nodes by observing the events of interest of neighbouring nodes for a time limit (every 10s), and sent on the route which has had a packet loss (the source has not had a reply from the destination). Packet loss (delay) is the danger signal, the non-malicious nodes producing self-antigen give a form of positive feedback and therefore reinforces the misbehaviour detection process. The system is devised for the DSR routing protocol. It needs more time to detect misbehaviour than previous work by the authors that do not have a virtual thymus, but this is compensated to some extent by using memory detectors. Clustering is also implemented in this AIS which is a feature that is proven to increase true positives and decrease false positives in other works of the authors (Sarafijanovic & Le Boudec, 2004) which also incorporates negative and clonal selection.

This system is simulated using GloMoSim (Zeng, Bargodia, & Gerla, 1998), with a fixed network size of 40 nodes and so scalability has not been tested. Also all nodes have the same transmission range of 355 m and with random mobility of 1 m/s, so in a way nodes are exactly the same. This system is real-time and nodes have an up-to-date repository of detectors and antigen representing self. The previous version of this system was offline and required a learning phase. The work done by the authors concentrates on two misbehaviours:

1) Non-forwarding route requests and non-answering from its route cache,

2) Non-forwarding data packets (Sarafijanovic & Le Boudec, 2004).

The proposed system's (Le Boudec & Sarafijanovic, 2004) classification rate is affected by the misbehaviour probability of malicious nodes; at very low and very high rates it is very in-accurate. The main problem the author had was the mapping of parameters, which affected results greatly and therefore they propose the automatic generation of parameters.

Farooq (2009) focuses on BeeAdHoc which is a Swarm Based routing protocol, and proposes BeeAIS-DC that overcomes the weaknesses of previous anomaly detection systems for this routing protocol: BeeAIS and BeeSec. BeeSec is not immune inspired and is a cryptographic system which requires key management. This is usually a challenge in ad-hoc networks, because it requires a central point of contact. BeeAIS is an immune based algorithm that discriminates between self and non-self. It has static nodes, needs a 50s training period, is not adaptable and does not have a dynamic self-image. These AISs are all based on the BeeAdHoc routing protocol (Forooq, 2009) which is inspired from the way bees communicate in a hive. BeeAdHoc is specifically designed for ad-hoc networks and has shown better performance and high throughputs in dynamic topologies. In order to mitigate the security problems of BeeAdHoc, the routing algorithm itself has had to be modified in all occasions to allow for the implementation of these security mechanisms.

In BeeAIS-DC, the antigens are the scouts[1] and forager[2] type 1 and forager type 2. The attacks considered here are Forging Forward Scout, which try to install a fake route, and Forging Backward Scout, which spoofs backward scouts. They used ns2 (Issariyakul & Hossain, 2012) and implemented the BeeAIS-DC network, with nine nodes and a simple topology. For attack implementation, they launched two different types of routing attacks, and monitored the traffic in three points of the networks to generate a traffic map. Each scout has a Dendritic Cell (DC) associated with it, that resides in the node, it samples scouts within its lifetime and if the scouts have danger signals higher than a threshold the DC matures and goes to the thymus (the DC has to check its state at the end of each interval), in which a co-stimulary signal is used to assign a high danger level to the scout and the path associated with it. The detectors in the thymus are generated randomly and then using negative selection (NS) those that match the semi-mature antigen collected are known as self and deleted, and the rest remain. They are then updated in a genetic algorithm manner, also when a detector is matched with an antigen in a mature DC it is activated.

In Drozda (2010), an energy aware method has been devised to detect misbehaviour in ad-hoc networks. This is done using a set of features that are applied as cascading classifiers, with the energy cost of the

[1] Scouts are responsible for finding routs in the network, and giving these routs to Foragers

[2] Foragers are the main workers; they receive data packets and deliver them to their destination in a source routed mode. Forager nodes are of two types, delay foragers and lifetime foragers, which gather the delay or battery lifetime of the nodes they visit.

classifier increasing as the need for precision in detection is recognised. The OSI protocol stack is studied, and features are extracted from each level that help keep track of the correct routing behaviour of the neighbouring node with regard to the packets coming from it; the behaviours of nodes downstream are monitored by the nodes upstream in the route and calculated features are sent back towards where the packets came from attached to the acks going to the same node. This feature set propagation is done periodically using time windows of 50, 100, 250 and 500 seconds, which affects greatly the performance of the proposed method. Figure 3 shows the method Drozda uses to calculate weights assigned to each feature shown in Figure 4.

This method benefits from partitioning the feature set into three levels of energy usage, with the energy levels escalading as we go into the lower levels of the OSI; but as with traditional security mechanisms in ad-hoc networks the watchdog features that monitors the forwarding of packets by neighbouring nodes is the most effective one in detecting misbehaving nodes. Misbehaviour in this research was non forwarding of packets, delaying packets and wormholes. The underlying routing protocol was AODV, with UDP for its transport layer, and IEEE 802/11 MAC protocol as the data-link layer protocol. Drozda investigates the effectiveness of the feature set from the local and neighbouring nodes in classifying the misbehaviour and produces a set of weights quantifying these features. This method also shows a reduction in energy usage in comparison with using the watchdog feature alone (Drozda, Schildt, Schaust, & Szczerbicka, 2010).

The method used by Drozda needs a change in the data link layer of AODV, in order to accommodate the features to be sent back. This is said to be performed at no extra cost, but with the feature set size growing greatly with the use of the larger sets, this would need clarification. Also the current implementation of this method was performed implicitly, and without actually implementing the algorithm on the network nodes. The simulation performed was conducted using the Jist/SWANS (Barr, 2012) simulator, which is a java based network simulator that has a better performance speed compared with Glomosim and ns2. They used the random waypoint movement model with 1718 nodes in a 3000m*3000 m area.

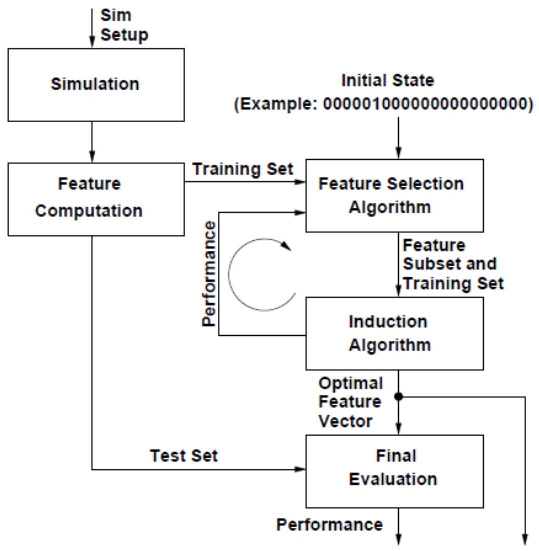
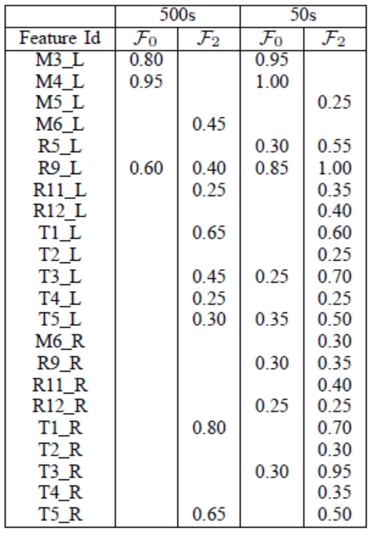


Figure 3- The Wrapper approach (Drozda,



	500s		50s	
Feature Id	$\mathcal{F}_0$	$\mathcal{F}_2$	$\mathcal{F}_0$	$\mathcal{F}_2$
M3_L	0.80		0.95	
M4_L	0.95		1.00	
M5_L				0.25
M6_L		0.45		
R5_L			0.30	0.55
R9_L	0.60	0.40	0.85	1.00
R11_L		0.25		0.35
R12_L				0.40
T1_L		0.65		0.60
T2_L				0.25
T3_L		0.45	0.25	0.70
T4_L		0.25		0.25
T5_L		0.30	0.35	0.50
M6_R				0.30
R9_R			0.30	0.35
R11_R				0.40
R12_R			0.25	0.25
T1_R		0.80		0.70
T2_R				0.30
T3_R			0.30	0.95
T4_R				0.35
T5_R		0.65		0.50

Figure 4- Feature Weights, those greater than 0.25 are shown (Drozda,

Schildt, Schaust, & Szczerbicka, 2010)	Schildt, Schaust, & Szczerbicka, 2010)

Initially 24 features are selected from the OSI protocol stack, which are either performance related like throughput or topology related like node degree; also, the watchdog feature and other features from previous studies in traditional literature are investigated. The majority of the proposed feature-set are averages of primary features[1] for each node in a time window. These features have different costs in terms of energy and they are classified into three sub classes by their energy rating. They are then used in a cascading manner to provide co-stimulation. To find a good feature set to be used by the nodes an optimisation algorithm has been used in a wrapper approach, on the feature set. The outcome of this optimization method is a weight assigned to each of the features showing their effectiveness, and those that are greater than 0.25 are shown in the table in Figure 4.

Table 3-1–A Classification of Immune-based Anomaly detection in ad-hoc networks

	Routing Protocol	Immune Feature	Simulator	Misbehaviours	Results	Lack
Drozda 2010 **An Immuno-Inspired Approach to Misbehaviour detection in Ad-hoc Networks**	AODV	Co-Stimulation (features from other nodes are considered when something is suspicious)	JiST/ SWANS	Packet Dropping, Packet Delaying, Worm holes	Good feature selection method, cascading classifiers reduces energy usage, Many nodes in a large area simulated	The data is extracted from a simulator (not real) and the computational overhead of each node is not considered.
Foroog 2008 **A Sense of Danger Dendritic Cells Inspired AIS for MANET security**	BeeAdHoc	Danger signals, CSM in thymus in a sort of NS method, with the addition of randomly generated detectors or T-cells.	ns2	Forging forward scouts and Forging backward scouts	Claims a better performance in terms of throughput and detection rates	The danger signal is not clear, the detectors (T-Cells) are based on the same input as the danger signals – a loop seems to exist.
Hortos 2003 **An Artificial Immune System for Securing Mobile Ad-hoc Networks Against Intrusion Attacks**	Secure Routing Protocol (SRP)	Negative selection and Genetic algorithms	ns2-SRP*	Route manipulation attack	The NC method uses 8.25% of the PC method's memory, with 1% false positives	Does not have much immune inspiration.
Sarafijanovic 2004 **An AIS for Misbehaviour Detection in MANETs with Virtual Thymus, Clustering, Danger Signal and Memory Detectors**	DSR	Virtual thymus, clustering, danger signals, memory detectors	Glomosim	Not forwarding data packets Not answering route requests	Reduced number of false positives by adding danger signals	The comparisons made do not show the effect of each immune feature or the clustering separately. The danger signal could get misused itself
Sarafijanovic 2005 **An Artificial immune system approach with secondary response**	DSR	NS, Clonal selection, innate (No DCA – only a repudiation system) &	Glomosim	Non-forwarding route requests and non answering from its route cache,	Better response time and ability to detect new attacks reduced false positives	No distributed detection, no local or cooperative response to misbehaving node

[1] Features already available to the node, without any processing performed on data packets.

for misbehaviour detection in mobile ad-hoc networks		adaptive components		Non-forwarding data packets		

* ns2-SRP is simulated by manipulation of DSR protocol

3.5 Immune-Inspired Intrusion Response

As discussed before all nodes in an ad-hoc network are responsible for relaying data, to enable communication between far away nodes. Considering the challenges in ad-hoc networks and the various possible misbehaviours associated with them, Drozda et al. (2010) propose an autonomic, adaptive detection method that detects misbehaviour with low false positives and limited human intervention. After looking into the different methods of bio-inspired intrusion detection for ad-hoc networks, this technique stood out for its low false positives, better detection of misbehaviours while using very little energy. We assume this technique is now in place and we look into the response to the detection of misbehaviour.

The problem of selecting a suitable response mechanism in an ad-hoc network can also be solved in a bio-inspired way although typically this part of attack mitigation is performed by human intervention and no automatic response algorithm exists for WSNs (Schaust & Szczerbicka, 2011). Intrusion response systems can be categorised in different ways (Stakhanova, Basu, & Wong, 2007); one of which is by the degree of automation. They are categorised into automatic response systems, manual response systems and notification systems where only an alert is issued.

Automatic response systems are themselves categorised into static and adaptive systems, in ad-hoc networks where the system is dynamic itself, an adaptive response system would suit it better. Bio-inspired algorithms benefit from a property that gives them superiority: adaptiveness. We conclude that devising bio-inspired methods for adaptive response would be most logical as we focus on the enhancement of one of these methods. In this method the ability to have an evolving self-image is provided by a continuous self-feedback which corrects the self image using positive or negative updates. Schaust and Szczerbicka (2011) propose a bio-inspired response algorithm to intrusions detected. Response methods can be proactive or have a delayed response. Here we focus on shortening the delay caused by the response system.

Node shutdown is a response suited to some attacks such as packet injection (Schaust & Szczerbicka, 2011). Generally response systems aim to lessen network loss by depressing the attacker (Ahmed, Samad, & Mahmood, 2006). In (Ahmed, Samad, & Mahmood, 2006), the suggested responses to misbehaviour in MANETs are removing the misbehaving node from routing tables, blocking traffic to and from it or reducing its trust level. Other possible responses are restarting the misbehaving node, flashing the misbehaving node’s operating system or shutting it down. In the work done by Schaust and Szczerbicka (2011) the proposed bio-inspired solution is used to try and find the best response from 3 misbehaviours and 4 possible responses. They look for the effects of each response on quality of service parameters like received packets, good-put and delivery time with in a specific feedback time period. If the response produced a positive impact the affinity of that response toward that misbehaviour is increased and if a negative impact is seen then the affinity of that response is reduced and that response is undone if possible. This way the system learns which responses are more suitable for which misbehaviours and develops a form of cognitive memory.

A drawback of their work was the non-adaptive response feedback time, which is the time when a set of network features were measured after the response was enacted. They assumed a fixed response feedback time of 5 seconds, which caused a greater number of negative responses mainly due to communication errors, while the actual negative response was much lower. They find that a suitable timer is necessary to revoke responses that cause the QoS to fall, either because of communication errors or lacking to react to the misbehaviour in a reasonable time. Waiting too long to cancel the response is undesirable as it causes unnecessary overhead to the network (Schaust & Szczerbicka, 2011). We propose developing a suitable timer as an extension to their work stating it as a necessity to their algorithm as nearly all negative feedbacks were caused by not receiving an ack.

3.6 Conclusion

In this chapter, we have reviewed immune-inspired security solutions in ad-hoc networks. First the biological immune system has been introduced as a means of inspiration for the design of immune inspired algorithms for the security of ad-hoc networks. We have then looked into the existing immune inspired solutions for intrusion response and discussed their positive and negative points. We have then considered the problem of response to a detected misbehaviour and concentrated on enhancing the Antigen Receptor Degeneracy Algorithm (ARDA) as the only automatic bio-inspired response solution. In the next chapter, we try to devise a feedback timer to be used with the algorithm.

Chapter 4

Improving Immune-Inspired Intrusion Response

4.1 Introduction

In this chapter we look into the *Antigen-Receptor degeneracy behaviour* algorithm (ARDA) in (Schaust & Szczerbicka, 2011), as the prevalent bio-inspired autonomous response study for WSNs. We explain the algorithm which uses the bio-inspired framework and show how it uses feedbacks to adapt its responses to the misbehaviour and network conditions, so that it achieves better quality of service ultimately. As mentioned before the ARDA algorithm assumes that an intrusion detection module is in place and tries to match four different responses to the following three misbehaviours while adapting to the network conditions and trying to maintain a high quality of service. We then discus how we try to improve this method, by finding a more accurate feedback time for the shutdown response.

4.2 ARDA Method

The ARDA algorithm is a bio-inspired algorithm and is designed in accordance to the bio-inspired framework. This algorithm assumes that an intrusion detection module is in place and tries to match 4 different responses to the following three misbehaviours while adapting to the network conditions and trying to maintain a high quality of service. The misbehaviours in the mentioned study are:

1. *Selective packet forwarding*: forwarding data packets from specific routes and therefore compromising on cooperation in the network.
2. *Artificial packet injection*: injecting artificially generated data into the network in order to limit bandwidth or mislead other nodes.
3. *Manipulation of packet content*: falsifying the collected information by changing content of data packets; this misbehaviour is usually associated with a time delay.

The responses implemented for the above misbehaviours which can be used against any of the misbehaviours are:

1. *No Response*: used when the misbehaviour is very similar to normal network behaviour.
2. *Shut down*: the misbehaving node is shutdown using an over the air command.
3. *Flashing*: the misbehaving node's operating system is replaced by a large packet containing a typical OS image of a sensor node.
4. *Blacklisting*: the misbehaving node is temporarily back-listed and unavailable for routing.

The *shutdown* response is chosen as the basic response which can be tested for most types of ad-hoc network devices, and has a considerable unsettling effect on the network which is equal or greater than the other responses.

The ARDA algorithm is illustrated in Figure 6.

```
for each $A$ do
    $m \longleftarrow$ misbehavior specific mask
    for each $\theta \in \Theta$ do
        $d_{cosine}(\theta, A) = \frac{\sum_{i=0}^{n} m_i A_i \times m_i \theta_i}{\sqrt{\sum (m_i A_i)^2} \times \sqrt{\sum (m_i \theta_i)^2}}$
    end
    $\sigma = k$-nearest vectors for $d_{cosine}(\theta, A)$
    if ($\forall \sigma$ the response $res_i$ is available) then
        Select correlated response $res_i$
    end
    else
        $res_i$ = response of $minDistance(\forall \sigma_k \in \sigma_k, A)$
    end
    Initiate $res_i$ and set $OngoingResponse$ = true
    Inform Detection Unit about response activation
    Wait $t_w$ to trigger 2-hop feedback analysis for $res_i$
    Wait $t_{feed}$ for feedback ACK
    if $FeedbackAnalysisIsPositiv$ then
        Increase weights by $w_{pos}$ in $m$ for $A$
        Add $A$ to receptor set
        Connect $res_i$ with $A$ under $m$
    end
    else
        Decrease weights by $w_{neg}$ for $A$ under $m$
        Add $A$ to receptor set and link all responses equally to it
    end
    Inform Detection Unit about response finish
    Set $OngoingResponse$ = false
end
```

Figure 5- Antigen Receptor Degeneracy Algorithm: ARDA. Algorithm requirements are initial receptor set Θ with response matching, k value for kNN and inputs are Time window based antigen vector A with an anomaly indication from the detection unit (Schaust & Szczerbicka, 2011).

The ARDA algorithm begins by assuming a receptor set exists and at least one response is assigned to a receptor, and responses are mapped onto a mask vector. The assignment is currently performed based on expert knowledge and manually. Antigens are collected continuously using a logging mechanism where network features are extracted from the communications channel in a specific time of Δt which is 5 seconds. Antigens are then classified as misbehaviours using an existing classification algorithm. The misbehaviours are then mapped onto the receptor vector and similarity and distance measurements are computed. Then the k-Nearest Neighbour algorithm is used to compute the k nearest receptor set using the distance values calculated before. The receptor set is then evaluated and the most similar and minimum distance response is chosen. This response is enacted and after an initial settling down time, the feedback procedure is started. This procedure is begun by starting a feedback interval timer and initiating a two-hop acknowledgement mechanism, while the quality of service parameters are measured. The authors have used 5 seconds for their feedback interval, but as there was a bug in their chosen routing protocol, DYMO, they were not able to specify the exact reason where an acknowledgment was not received in the time period. They state that if the network quality parameters were better after the response was enacted they would expect a positive feedback acknowledgment. The quality of service parameters they used are received packets, good put and delivery time. They were able to identify a response for each misbehaviour but as they had problems in the feedback procedure they have suggested the use of an adaptive feedback timer in the end.

4.3 Our Approach

We focus on the shutdown response, and the network reaction to this response which could be in three ways: no change, recover after a short time, or unable to recover. We simulate each type of network to find out whether any relationship between the type of network and recovery time exists, or if we can find any other factor that affects the time in our experiment. We aim to simulate the three main types of ad-hoc networks discussed in previous sections and determine their reaction to the response.

The feedback time is the time it takes for the network to stabilise in reaction to a response. After this time, the network parameters are reliable enough to be used in the feature selection part of the *ARDA* algorithm. Earlier collection of data from the network would result in false feedback in the algorithm and later data collection would cause delay in response and inability to act on other detected intrusions.

As mentioned before, the work we aim to do in this study would not only benefit the Schaust algorithm, but any other algorithm that needs to read network features after causing a change in the network. Some applications of ad-hoc need to be very stable and require guaranteed sensing, coverage and connectivity throughout their operation (Al Islam, Hyder, Kabir, & Naznin, 2010) e.g. security monitoring (Chehri, Fortier, & Tardif, 2007) and motion tracking (Gu, Lo, & Niemegeers, 2009).

Other application is in the analysis of ad-hoc network stability, in which we can determine the fault tolerance of WSN, WPAN and SNU against node failures. We can answer questions like what is the probability of a network to be stabilised after a certain amounts of nodes fail. In the stability analysis, we can also investigate the ad-hoc network resilience to the set of attacks, in which nodes are destroyed such as DoS attacks.

4.4 Implementation

In this section, we discuss the used simulation tools are described; ns2, nam and gawk are some of the tools we used in order to simulate the three types of ad-hoc networks and calculate the required measures. We then discuss the implementation details of the simulation and result analysis of the experiment.

4.4.1 Simulation Tools

Ns2 (Network Simulator 2) was used as the simulation tool as it was a powerful open-source simulation tool that had the basic modules needed for our experiments. Opnet, OMNeT++ and ns2 were considered as simulation tools, but as we needed to customise the simulator in order to produce features such as over the air shutdown response command, ns2 was chosen. Ns2 has the capability to simulate many features such as, traffic source behaviour (www, CBR[1]and VBR[2]), routing, packet flow, network topology, multicasting and mobile nodes. Ns2 supports five ad-hoc routing protocols, AODV, DSDV, DSR, TORA and PUMA. The routing protocol Ad-hoc On-demand Distance Vector (AODV) is one of the built in protocols of ns2.

Ns2 is a discrete-event driven and an object oriented simulator with a frontend OTcl[3] interpreter and backend C++ modules. The topology of the simulated network and its overall configuration is initially described in OTcl and then a simulator object is defined through the C++ code detailing the network protocols, packet headers, etc. OTcl is more suitable for the configuration part of the simulation compared to C++, as it is faster to change but slower to run.

The basic component used to develop our experiment is the ns2 mobile node module. A mobile node is a basic ns2 node with the added abilities of mobility and transmitting or receiving from a wireless channel. Below is a diagram showing the mobile node structure, which depicts a basic ns2 node connected to an ad-hoc routing agent and a network stack consisting of a link layer object, interface queue, a MAC object and a physical network interface connected to an antenna. We configured each of these layers according to the type of ad-hoc network we were simulating, either WSN, WPAN or SNU.

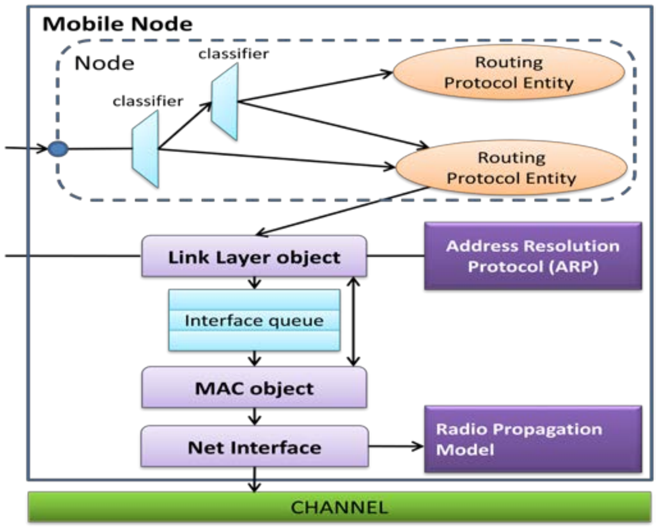


Figure 6 – Structure of a mobile node in ns2

[1] Constant Bit Rate

[2] Variable Bit Rate

[3] Object-oriented Tool Command Language

The main simulation and network configuration in ns2 takes place in the tcl script. This is further explained in the following section. After describing the network to ns2 using the tcl script is run and the output files are analysed for results. There are two main output files, a trace file with all the events such as send, receive, packet drop, etc... and a nam file which can then be used by the network animator to view the simulation in graphical mode. In our experiment we analysed the trace file using gawk and calculated the needed network parameters such as throughput, which is explained in Section 4.4.2. Network throughput as the main feature was then illustrated as a chart in MS Excel.

Nam (Network Animation and visualisation tool) is a separate component used with ns to provide a visual interpretation of the network topology. As stated before ns simulations automatically produce the nam file containing topology information such as nodes and their configurations, as well as trace data.

Gawk (Free Software Foundation, 2008) is a pattern scanning and processing language, which is the GNU implementation of awk. Gawk is suitable for generating reports and extracting pieces of data for processing. In our project Gawk version 4.0.0 was used to analyse ns trace files, in order to produce throughput and other network measures.

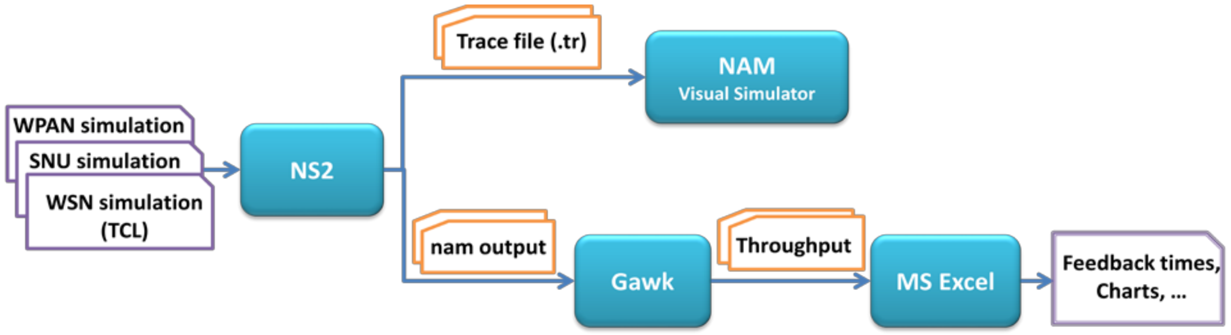


Figure 7- The implemented framework to compute feedback time for immune-based systems

4.4.2 Implementation Details

Our simulation begins by preparing the framework used to simulate the networks (Figure 7). We require a simulation tool that can implement an over the air shutdown or “off” command for its nodes. In our chosen simulator, ns2, this part of the AODV routing protocol was missing from the module *energymodel*. The module *energymodel* is configured to set the amount of energy used to send and receive data packets, the energy usage of a node in sleep mode, and tells it what to do during an off command or restart. The C++ source code for ns2 version 2.35 was adapted and then recompiled to produce this new version of ns2 used for experimentation.

We describe our network to the simulator by creating an instance of a Tcl*class simulator*, then configuring the nodes and describing the network topology to the simulator. The *tcl* script consists of several parts, initial configuration, network setup and end of the simulation. There are also other maintenance parts such as output file specification, which are given in full in the appendix. The initial configuration is show in Figure 8, where node initialisation and topology object configuration takes place.

```
# Here are the set of initial options
# channel type
set opt(chan)              Channel/WirelessChannel;
# radio-propagation model
set opt(prop)              Propagation/TwoRayGround;
# network interface type
set opt(netif)             Phy/WirelessPhy;
set opt(mac)               Mac/802_11;# MAC type
# interface queue type
set opt(ifq)               Queue/DropTail/PriQueue;
set opt(ll)                LL; # link layer type
# antenna model
set opt(ant)               Antenna/OmniAntenna;
set opt(ifqlen)            50; # max packet in ifq
set opt(rp)                AODV; # routing protocol
# X dimension of topography
set opt(x)                 300;
# Y dimension of topography
set opt(y)                 200;
# time of simulation end
set opt(stop)              6;
set opt(energy)            EnergyModel;
set opt(initialenergy)     10000;
set opt(txPower)           0.660;
set opt(rxPower)           0.395;
set opt(idlePower)         0.035;

#simulator setup
set ns          [new Simulator]
# set up topography object
settopo       [new Topography]
# configure the nodes
$ns node-config -adhocRouting $opt(rp) \
       -llType $opt(ll) \
       -macType $opt(mac) \
       -ifqType $opt(ifq) \
       -ifqLen $opt(ifqlen) \
       -antType $opt(ant) \
       -propType $opt(prop) \
       -phyType $opt(netif) \
       -channelType $opt(chan) \
       -topoInstance $topo \
       -agentTrace ON \
       -routerTrace ON \
       -macTrace ON \
       -movementTrace ON \
       -energyModel $opt(energy) \
       -initialEnergy $opt(initialenergy) \
       -txPower  $opt(txPower) \
       -rxPower  $opt(rxPower) \
       -idlePower  $opt(idlePower)
```

Figure 8- Initial network configuration in ns2

For network setup, we stacked the layers as a typical network with the AODV routing protocol, in line with the network implemented in (Drozda, Schildt, Schaust, & Szczerbicka, 2010). Nodes use the AODV routing protocol to find the path, and have a constant bit stream of 50 ms interval, sending packets of 1500 bytes size for the application layer, on top of a UDP link layer agent, with the node using the IEEE 802.11 as its MAC layer protocol (Figure 9).

```
# Set a connection between node_(2i) and node_(2i+1)
for {set i 1} {$i < $opt(flow)} { incr i } {
        set udp_($i) [new Agent/UDP]
        set sink_($i) [new Agent/LossMonitor]
        set cbr_($i) [new Application/Traffic/CBR]
        # 1500 - 20 byte IP header = 1480 bytes
        $udp_($i) set packetSize_ 1500
        # This size INCLUDES the UDP header
        $cbr_($i) set packetSize_ 1480
        $cbr_($i) set interval_ 0.05000
        $cbr_($i) attach-agent $udp_($i)
        $ns attach-agent $node_($j) $udp_($i)
        $ns attach-agent $node_($k) $sink_($i)
        $ns connect $udp_($i) $sink_($i)
        set l [expr ($i+ 0.0)]
        $ns at $l "$cbr_($i) start"
}
```

Figure 9 – Network setup in tcl script

By varying these settings and the number of nodes and the number of data flows the different types of network can be simulated. NS2 runs the *tcl* code and produces a trace file and a *nam* file which can then be viewed using the network animator tool (Figure 12, Figure 17 & Figure 21). Gawk was used to analyse the trace file as stated before and produce an incremental throughput as the result of the simulation. After considering a few network parameters like *throughput*, *end to end delay* and *traffic rate*, *throughput* showed the effect of node shutdown response better than others and so it was selected as the main network parameter for our experiment.

The *end to end delay* is the time taken for a packet to be transmitted across a network from source to destination. This parameter only shows a rough estimate of the time it takes for a packet to be sent over the network. *End to end delay* cannot display network stability because substitute routes may have longer or shorter delays and be perfectly stable. The *traffic rate* parameter is calculated as follows:

$$\sum_{f=1}^{Max\ Flow} \frac{No.of\ sent\ packets * Packet\ size * 8}{flow\ time}.$$

This would disregard whether a packet reached its destination or not, hence not illustrate if a successful reroute had occurred.

Throughput is calculated using the following formula:

$$\sum_{f=1}^{Max\ Flow} \frac{No.of\ received\ packets * Packet\ size * 8}{flow\ time}.$$

We used accumulative *throughput* and a constant bit rate of sent packets, so the average speed of received data remains constant until a response cause it to change. This change is illustrated in our charts instantaneously (chapter 5).

Figure 10 shows a section of gawk code used to calculate throughput for each data flow in a network. It begins by working out the duration of the flow and then calculates the average throughput by dividing the amount of data (in bits) that reaches its destination for that data flow in the duration of that flow. In order to get a more real-time picture of the network throughput, we calculated it incrementally from flow start time to the end of simulation, every 0.1 second. This was done using a Linux shell script calling the gawk parse code and setting the flow end time incrementally.

```
for (f in flows) {
        number_flows++;
        flowend = flow_end_time[f];
        flowstart = flow_start_time[f];
        flowtime = flowend - flowstart;
        if ( flowtime > 1 ) {
        throughput[f] = (packets_received_flow[f]*PKTSIZE*8) / flowtime;
        printf("%s %f %f %f\n", f, flowstart, flowend, rate[f],
                throughput[f]) >> "stats-throughput.dat";
        }
}
```

Figure 10 – gawk code calculating throughput for flows in network

After an overview on the implementation details of our experiment, we go on to discuss the experiment design, results and evaluation of our project.

Chapter 5

Evaluation

5.1 Introduction

In this chapter the design of the experiment and the evaluation method are explained. We hypothesise that the three categories of ad-hoc network take different lengths of time to recover from the shutdown response emitted from an intrusion detection system. In order to evaluate this, we consider an intrusion detection system working and in place, and try to measure the effects of one shutdown response upon our network and the time it takes for it to recover. The three types of network considered are WSN, SNU and WPAN, and the AODV routing protocol is the used ad-hoc routing protocol. In this context the nodes are immobile and scattered in a grid structure in the network (e.g. Figure 12). Throughput is the main network performance measure that's effected and measured in this experiment, although latency and delay are affected by attack.

The three categories of ad-hoc network can behave differently to different responses from Intrusion Detection Systems (IDS). In this research we consider the three network categories and try to evaluate the effect of node shutdown as a response to misbehaviour.

5.2 Methodology

In our study we try to produce a set of response feedback times, with respect to three types of ad-hoc networks: WSN, SNU and WPANs. We test each of these networks by simulating each type of network, considering the shutdown response to misbehaviour (such as packet injection); where by an over the air command the node is shutdown and is not part of the network anymore. Each type of network is tested 30 times, during each run one CBR[1] traffic is set up between a sender and receiver, and a number of misbehaving nodes are shutdown. The throughput is then calculated for each run, and an average is produced over the 30 runs. The throughput usually drops after the response and rises when the network is recovering by rerouting. If it is not able to find a different path to communicate, it will keep dropping. The time by which the network reaches up to 98% of the pre-response throughput is output as the response feedback time.

5.3 Experiment Design

The main characteristics of three types of networks experimented on, are displayed in Table 5-1. As discussed before, in WSN there are greater numbers of nodes, but with less transmission power, while in WPANs the number of nodes and their transmission capability is usually quite small and so they are placed close together, and in SNUs the nodes have higher processing and transmission capability and nodes are placed at a medium

[1] Constant Bit Rate

distance from each other. The number of flows stated in Table 5-1, indicate data flows between a source and destination, sending data in a constant bit rate (CBR). We experimentally noticed that the AODV implementation in ns2 v2.35 shows great instability in the cases where the destination is more than 5 hops away from the source, and so we set our flows to use less than that.

Table 5-1–The three types of networks and their settings in this experiment

	Number of Node	Data Flows	Test Area	Number of misbehaving nodes		
				10%	20%	50%
WSN	50	2	Medium(300*400 m)	5	10	25
WPAN	12	1	Small(150*200 m)	1	2	6
SNU	20	1	Small(150*200 m)	2	4	10

We tested each type of network with three rates of misbehaving nodes, where in each case 10%, 20% and 50% of the nodes were assumed to be malicious, and network recovery time was calculated after 30 runs of each case in each type of network. The end result was calculated as the point where the average of the 30 runs reaches the 98% normal curve.

Each of the three types of network was simulated for duration of 10 seconds with nodes turning on and starting CBR at 1.0 second, initializing and sending data packets of 1500 bytes size, at a rate of one every 0.05 second and the shutdown response occurring at 4.0 seconds. The throughput was calculated by analysing the simulation logs produced by ns2, parsing it using gawk and then the network feedback time was calculated using excel. The control throughput (without response) was considered the basis for the threshold and when the secondary reaches up to 98% of the control throughput, this was counted as the network feedback time. In order to get a more reliable time for each type of network, the experiment was repeated 30 times with random misbehaving nodes.

5.4 Control Experiments

In our analysis of the network reaction to the shutdown response, we began by running a set of control simulations without any responses, and a chart showing the average throughput of the three types of network in the control runs is shown in Figure 11.

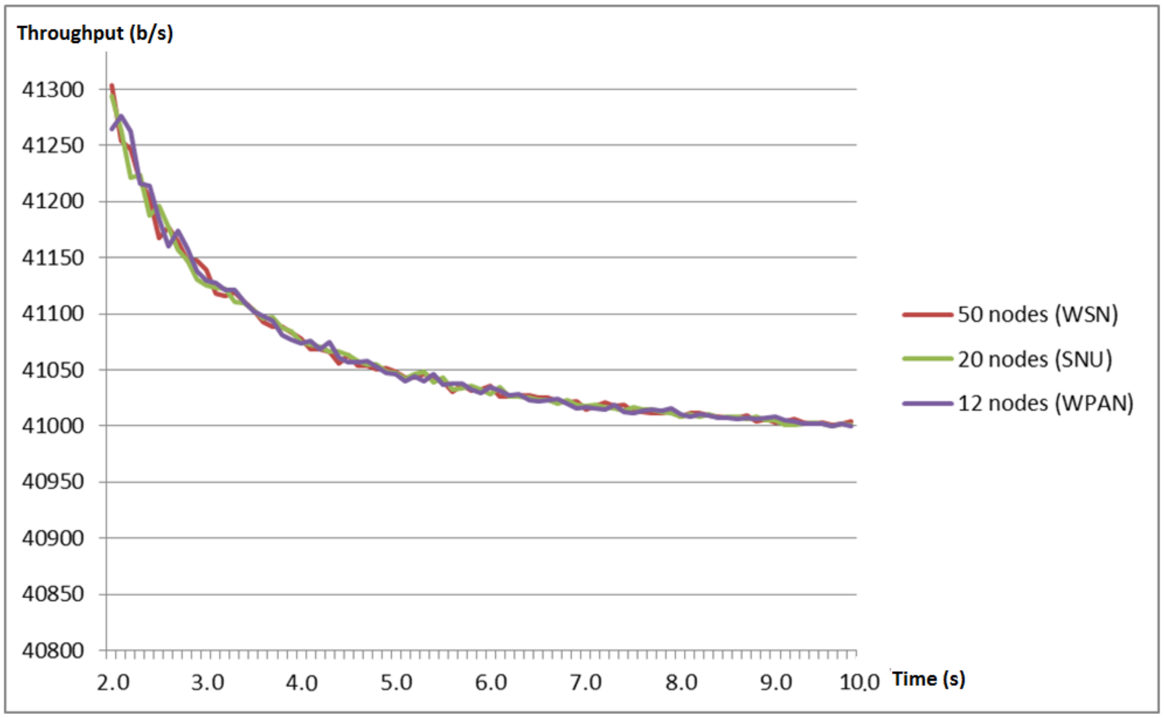
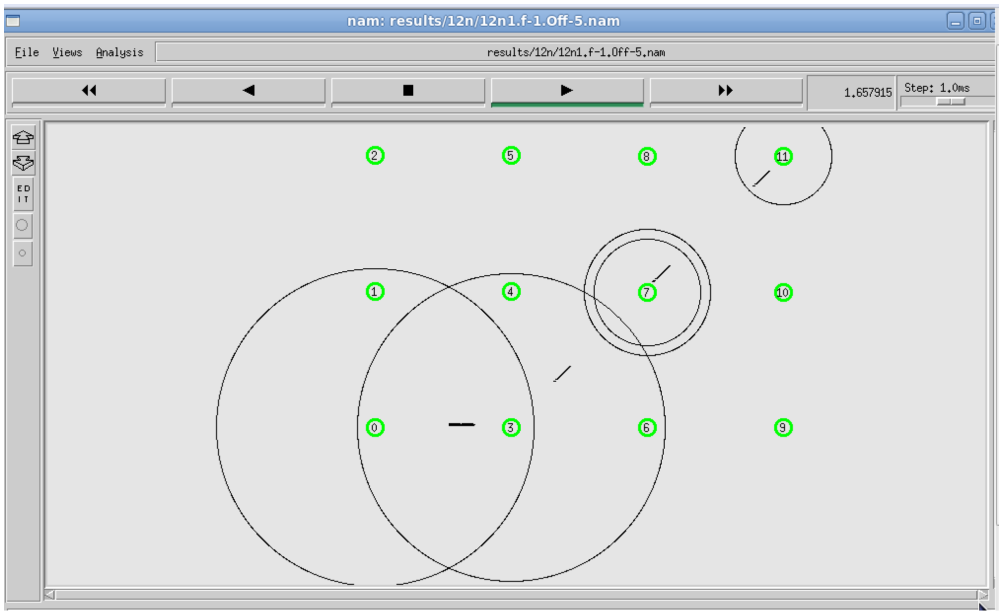


Figure 11- The average throughput in the control runs for the three types of ad-hoc networks

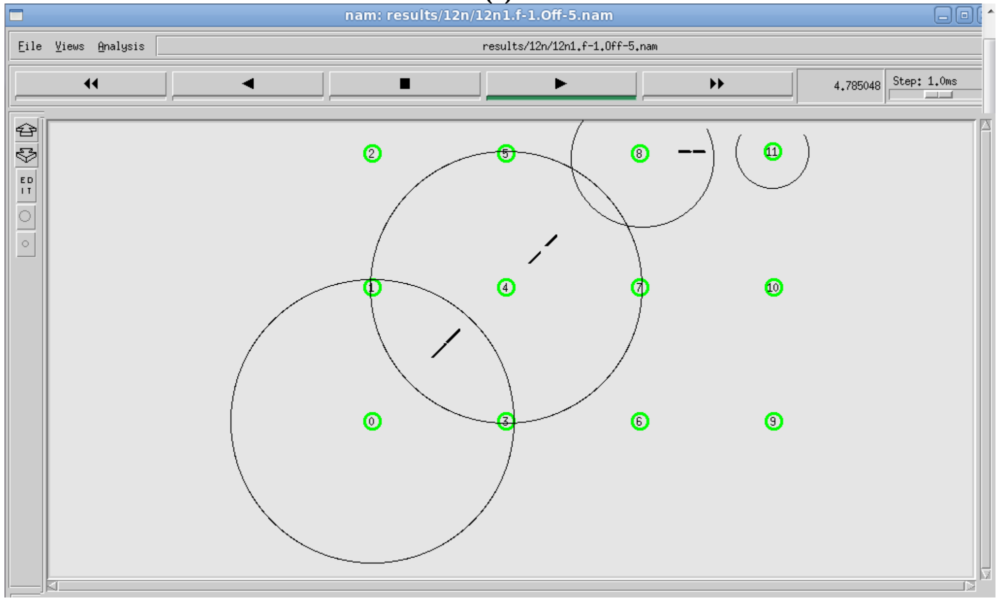
The control chart is showing that all 3 types of networks stabilise at around 410 kbit/s; this is because the constant bit rate data flow, produced by the node application agent, is reaching its destination without any interruptions. The control charts were also used to calculate the final feedback time. When the unstable network’s throughput reaches 98% of 410 kbits/s, 401.8 kbits/s we regard that network as stable.

5.5 WPAN

In order to find the network feedback time for a typical WPAN, we simulated a network of 12 nodes, scattered in an area of 150*200 m^2. The test was repeated 30 times and the position of the misbehaving nodes were varied to form a valid test. A sample run is presented in Figure 12. The 90 runs were then analysed for their throughput and other network measures using gawk and Excel.


nam: results/12n/12n1.f-1.Off-5.nam
File Views Analysis
results/12n/12n1.f-1.Off-5.nam
1.657915
Step: 1.0ms
EDIT
0
1
2
3
4
5
6
7
8
9
10
11

(a)


nam: results/12n/12n1.f-1.Off-5.nam
File Views Analysis
results/12n/12n1.f-1.Off-5.nam
4.785048
Step: 1.0ms
EDIT
0
1
2
3
4
5
6
7
8
9
10
11

(b)

Figure 12- A simulation of a WPAN with 12 nodes. (a) The initial flow between two nodes consists of nodes 0, 3, 7, 11 before the attack. (b) The rerouted path after node 3 is affected in the attack and is shut down by an immune-based intrusion response system.

The visualised examples of WPAN simulations are illustrated in Figure 12. Figure 12-(a) represent initial flow(s) before any interruption or attack. Figure 12-(b) display rerouted paths after some nodes in the initial paths are affected in the attacks and are shut down by the intrusion response system.

After drawing the 30 runs of the simulation with the shutdown response call occurring to random nodes in the network, three sets of charts were visually distinguishable (Figure 13). A set of unaffected runs where the misbehaving nodes were not in the route of the flow and so the response had no effect on the flow. A set of non-recovering runs where the shutdown nodes were in a way that the flow was not able to reroute, and the recovered flows where the shutdown node was in the route but the flow had managed to find a new path and maintain the flow successfully. In Figure 13 the 30 runs are shown, but some of the runs have similar curves and seem to be one. The percentage of these three sets changes in all three types of network, as the number of affected runs changes, this is shown in Table 5-4.

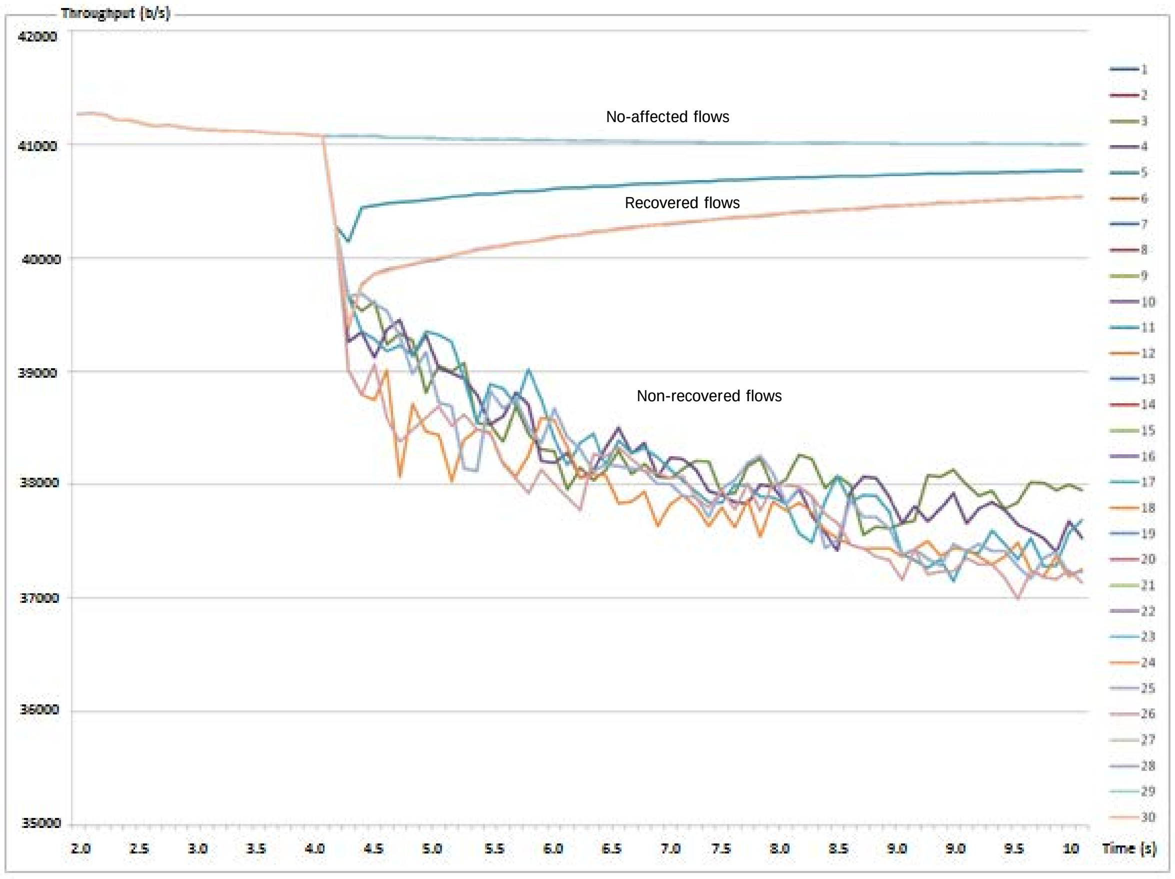


Figure 13- Three reaction to response in WPAN, when50% of nodes are shutdown at time 4.0s: non-affected, recovered, and non-recovered flows.

Table 5-2- The percentage of these three sets changes in all three types of network, as the number of affected runs changes

	WPAN		
⇩Behaviour	**10% off**	**20% off**	**50% off**
Recovered	7%	20%	23%
Not recovered	13%	27%	60%
Not affected	80%	53%	17%

Figure 14, illustrates the average throughput of recovered flows of the 30 runs in a WPAN when some nodes are affected after the shutdown response. The response causes a rapid drop in throughput and as the network recovers it rises, we can safely say that it has recovered when the throughput reaches the 98% normal curve, which is 98% of the control curve. We found out that in a WPAN when 10%, 20% and 50% of nodes are shut down, the throughput reaches 98% of its normal value at 6.1 s (Figure 14), 5.6 s (Figure 15) and 4.7 s (Figure 16), respectively. The 98% mark was calculated from control experiments as the 98% of the average normal WPAN throughput curve.

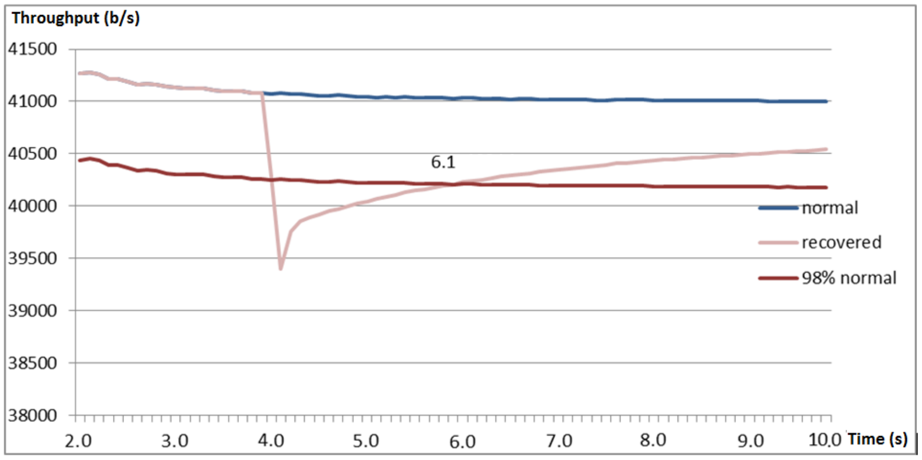


Figure 14- When 10% of nodes misbehaves in a WPAN, the throughput reaches 98% of its normal value at 6.1 s.

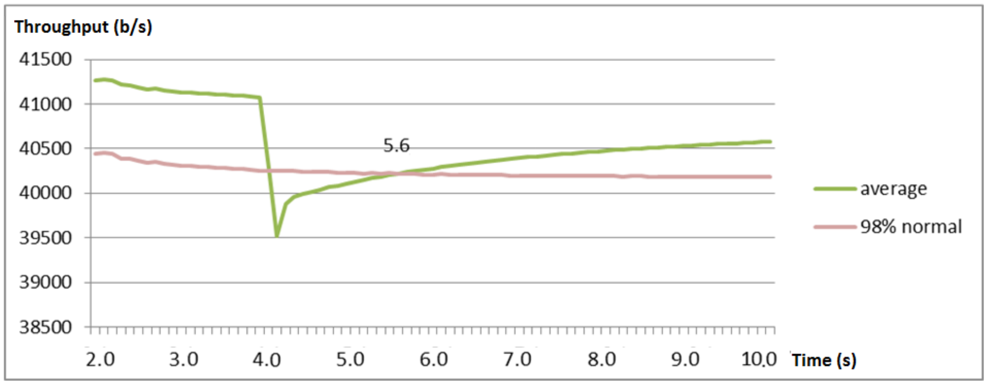


Figure 15- When 20% of nodes misbehaves in a WPAN, the throughput reaches 98% of its normal value at 5.6 s.

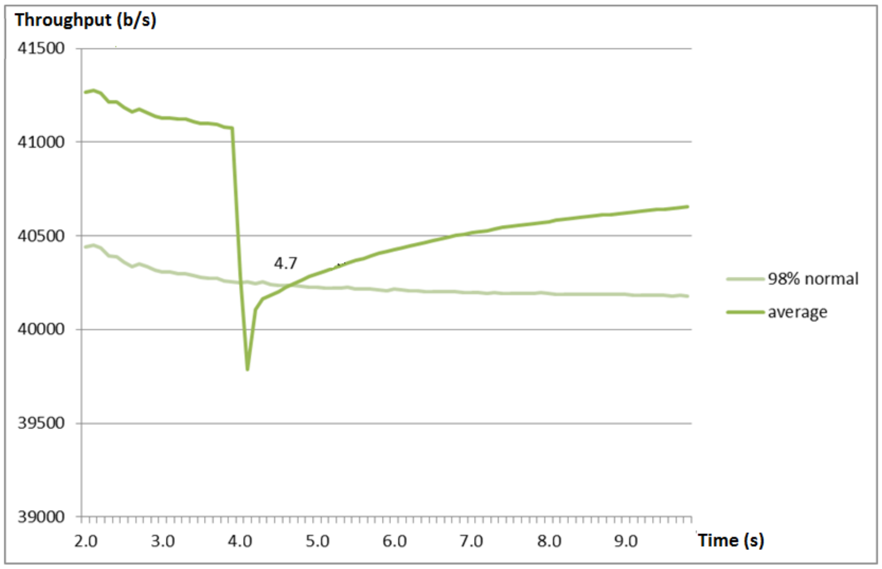
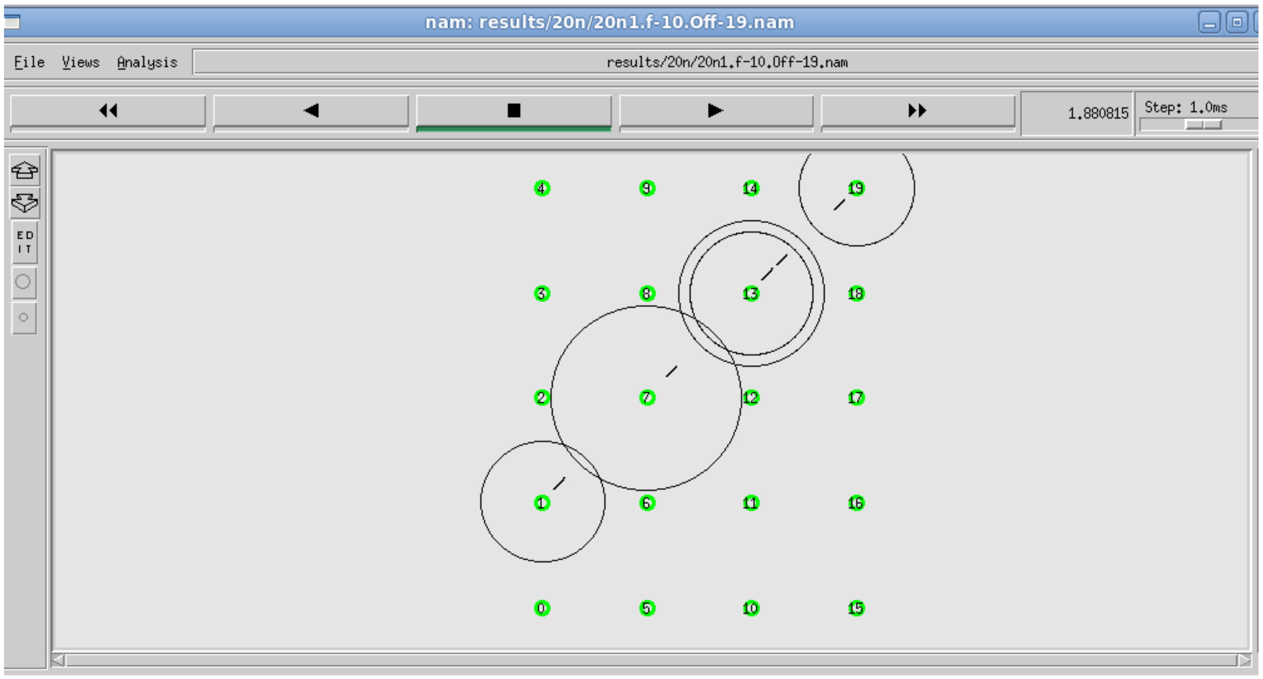


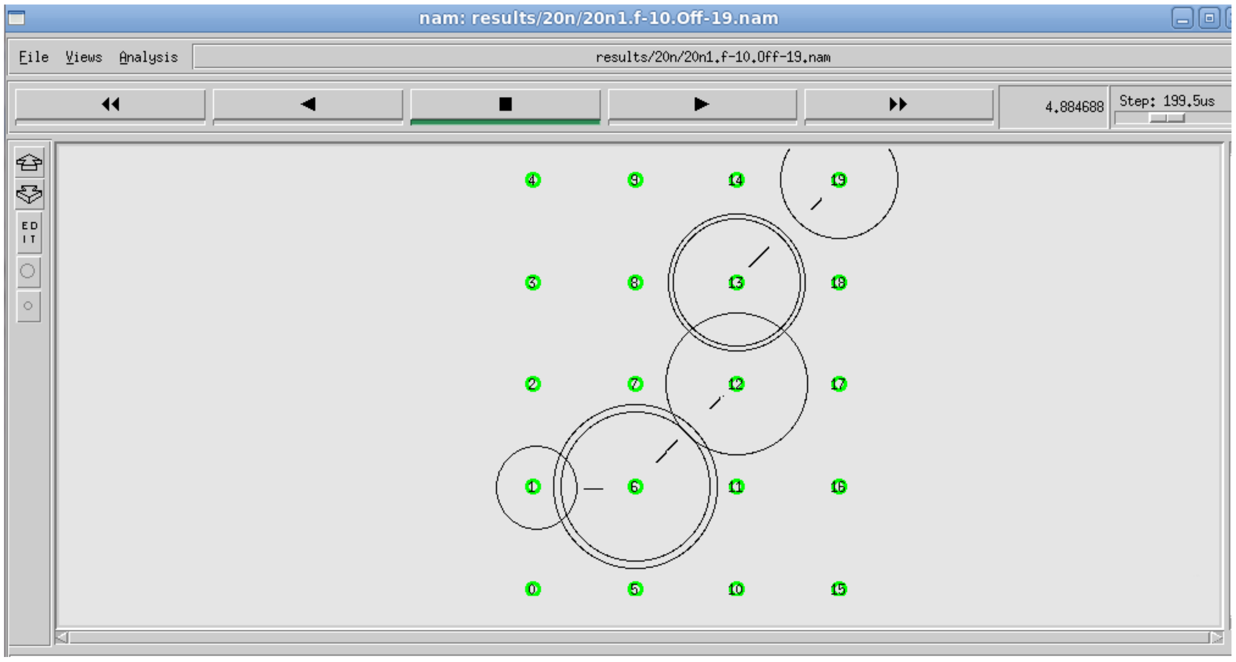
Figure 16- When 50% of nodes misbehaves in a WPAN, the throughput reaches 98% of its normal value at 4.7 s.

5.6 SNU

In order to find the network feedback time for a typical SNU, we simulated a network of 20 nodes, scattered in an area of 150*200 m^2. The test was repeated 30 times and the position of the misbehaving nodes were varied to form a valid test. A sample run is presented in Figure 17. The 90 runs were then analysed for their throughput and other network measures using gawk and Excel.



(a)



(b)

Figure 17- A simulation of a SNU with 20 nodes. (a) The initial flow between two nodes consists of nodes 1, 7, 13, 19 before the attack. (b) The rerouted path after node 7 is affected in the attack and is shut down by the intrusion response system.

The visualised examples of SNU simulations are illustrated in Figure 17. Figure 17-(a) shows initial flow(s) before any interruption or attack. Figure 17-(b) display rerouted paths after some nodes in the initial paths are affected in the attacks and are shut down by the intrusion response system.

Figure 18 illustrates the average throughput of recovered flows of the 30 runs in a SNU when some nodes are affected after the shutdown response. The response causes a rapid drop in throughput and as the network recovers it rises, we can safely say that it has recovered when the throughput reaches the 98% normal curve, which is 98% of the control curve. We found out that in a SNU when 10%, 20% and 50% of nodes are shut down, the throughputs reach 98% of its normal value at 4.3s (Figure 18), 4.9s (Figure 19) and 4.7s (Figure 20), respectively. The 98% mark was calculated from control experiments as the 98% of the average normal throughput curve.

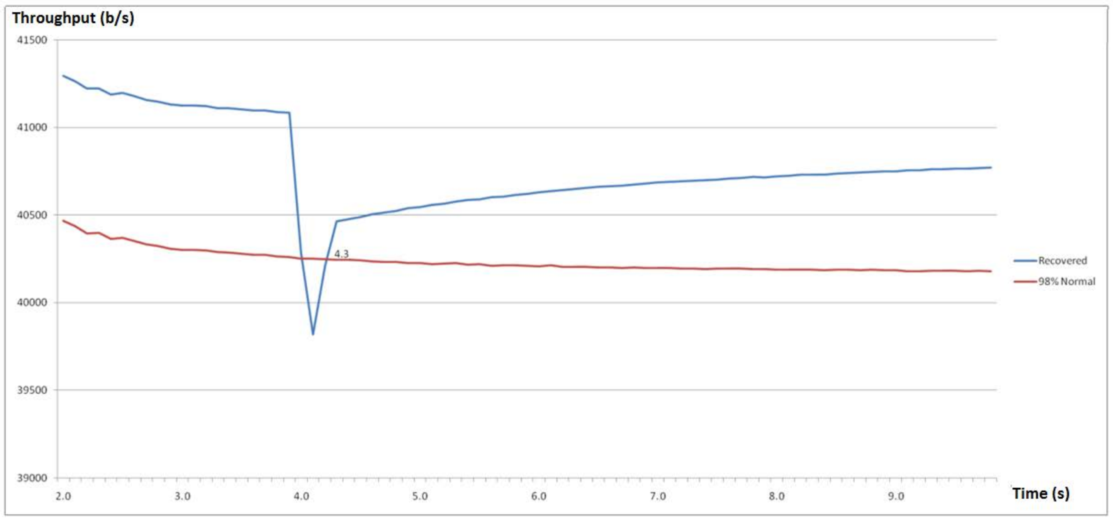


Figure 18-When 10% of nodes misbehaves in an SNU, the throughput reaches 98% of its normal value at 4.3s.

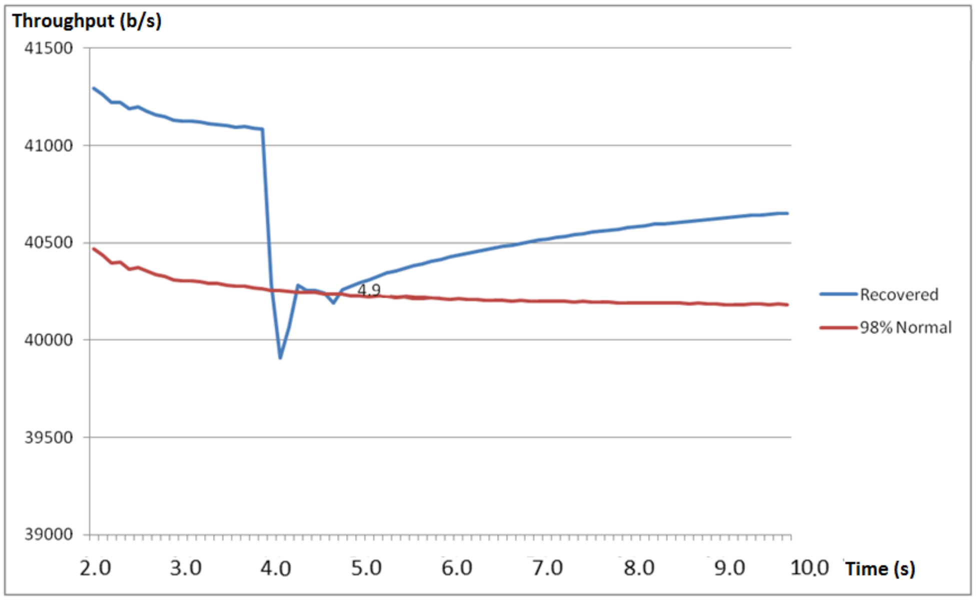


Figure 19-When 20% of nodes misbehaves in an SNU, the throughput reaches 98% of its normal value at 4.9s.

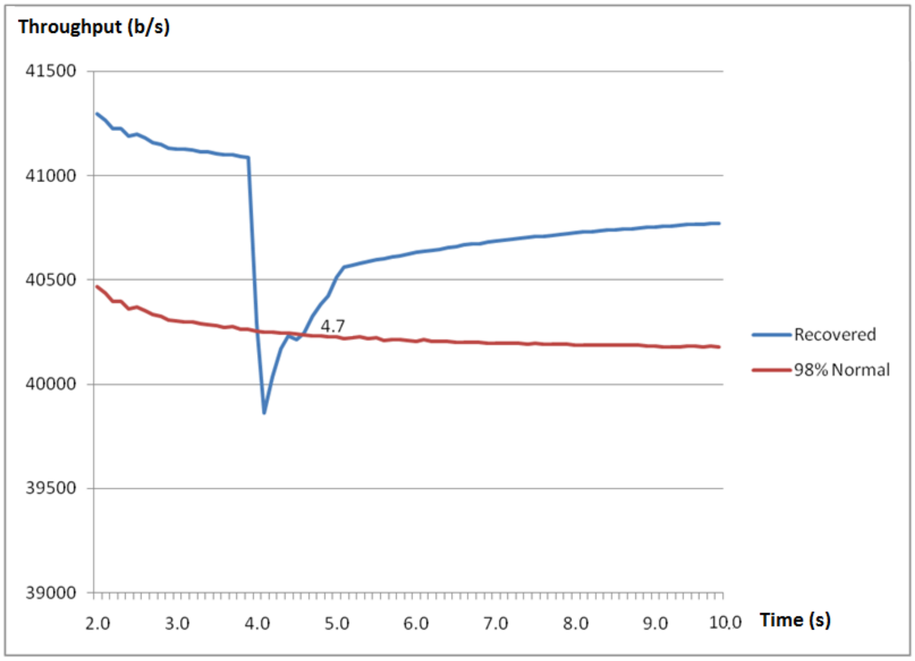
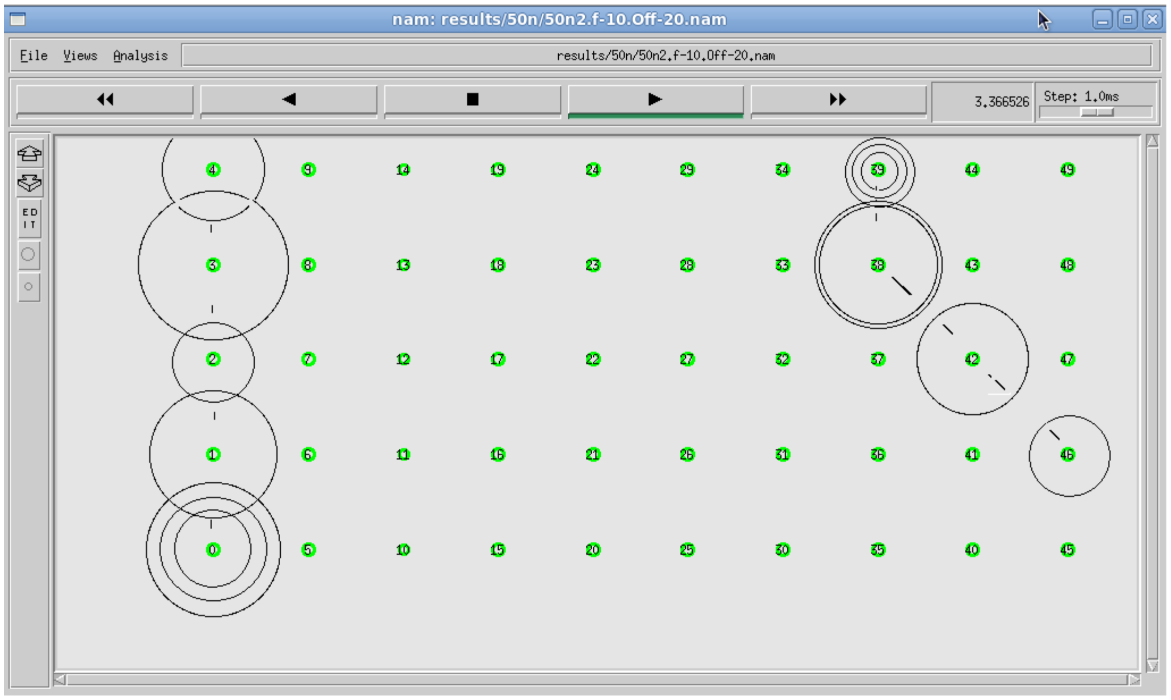


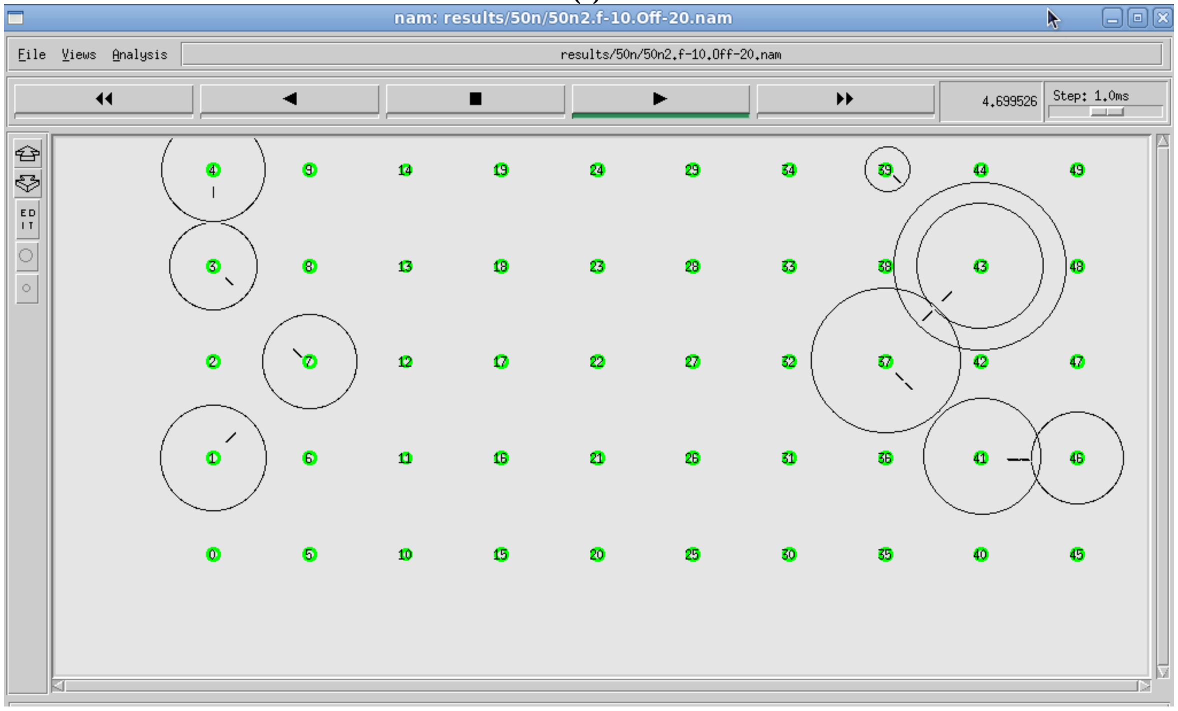
Figure 20-When 50% of nodes misbehaves in an SNU, the throughput reaches 98% of its normal value at 4.7s.

5.7 WSN

In order to find the network feedback time for a typical WSN, we simulated a network of 50 nodes, scattered in an area of 300*400 m^2. The test was repeated 30 times and the position of the misbehaving nodes were varied to form a valid test. A sample run is presented in Figure 21. The 90 runs were then analysed for their throughput and other network measures using gawk and Excel.


nam: results/50n/50n2.f-10.Off-20.nam
File Views Analysis
results/50n/50n2.f-10.Off-20.nam
3.366526
Step: 1.0ms

(a)


nam: results/50n/50n2.f-10.Off-20.nam
File Views Analysis
results/50n/50n2.f-10.Off-20.nam
4.699526
Step: 1.0ms

(b)

Figure 21- A simulation of a WSN with 50 nodes. (a) The initial flows; nodes 0, 1, 2, 3, 4 and nodes 39, 38, 42, 46 before the attack. (b) The rerouted path after some nodes 2, 38 and 42 are affected in the attack.

The visualised examples of WSN simulations are illustrated in Figure 21. Figure 21-(a) shows initial flow(s) before any interruption or attack. Figure 21-(b) display rerouted paths after some nodes in the initial paths are affected in the attacks and are shut down by the intrusion response system.

Figure 22 illustrates the average throughput of recovered flows of the 30 runs in a WSN when some nodes are affected after the shutdown response. The response causes a rapid drop in throughput and as the network recovers it rises, we can safely say that it has recovered when the throughput reaches the 98% normal curve, which is 98% of the control curve. We found out that in a WSN when10%, 20% and 50% of nodes are shut down; the throughputs reach 98% of its normal value at 6.1 s (Figure 22), 6.0 s (Figure 23) and 4.8 s (Figure 24), respectively. The 98% mark was calculated from control experiments as the 98% of the average normal throughput curve.

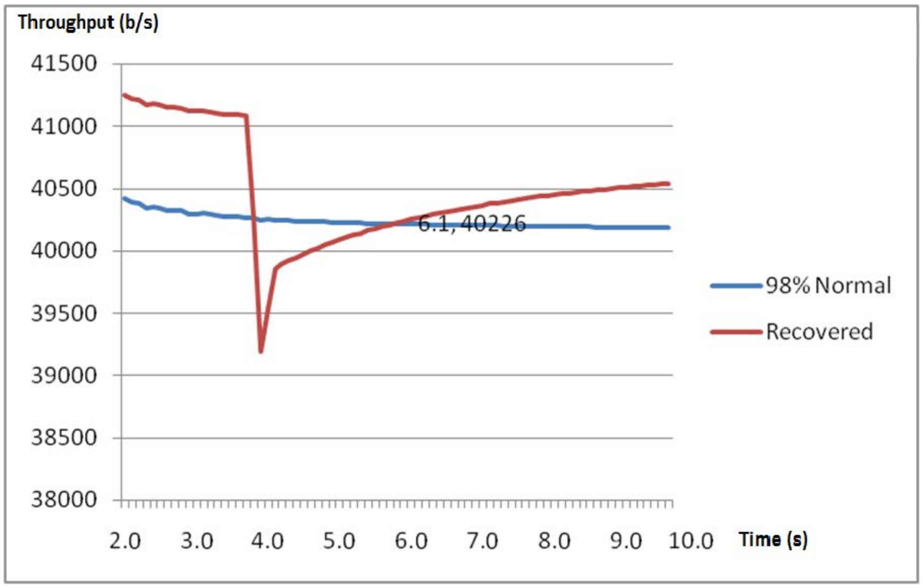


Figure 22-When 10% of nodes misbehaves in a WSN, the throughput reaches 98% of its normal value at 6.1s.

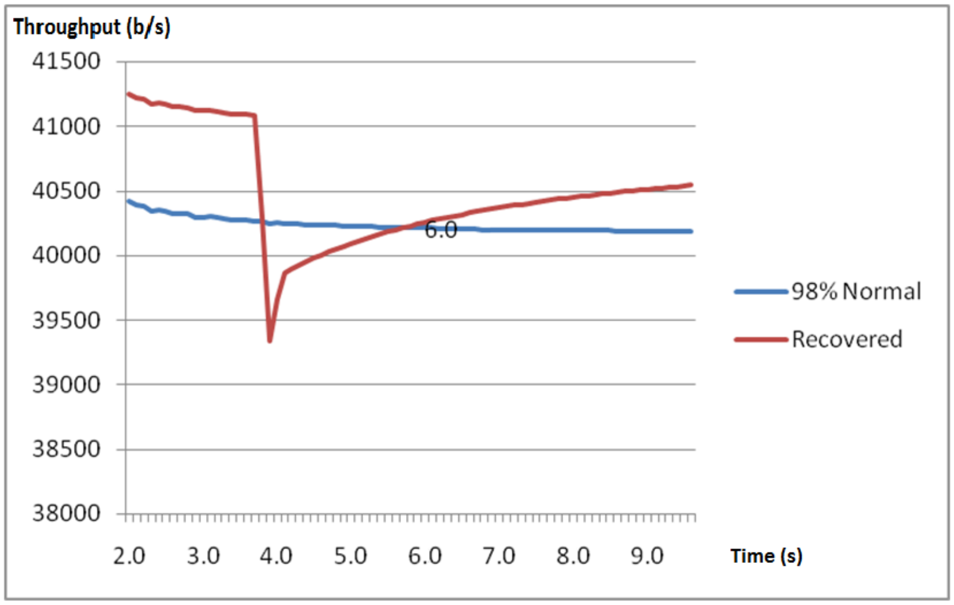


Figure 23- When 20% of nodes misbehaves in a WSN, the throughput reaches 98% of its normal value at 6.0s.

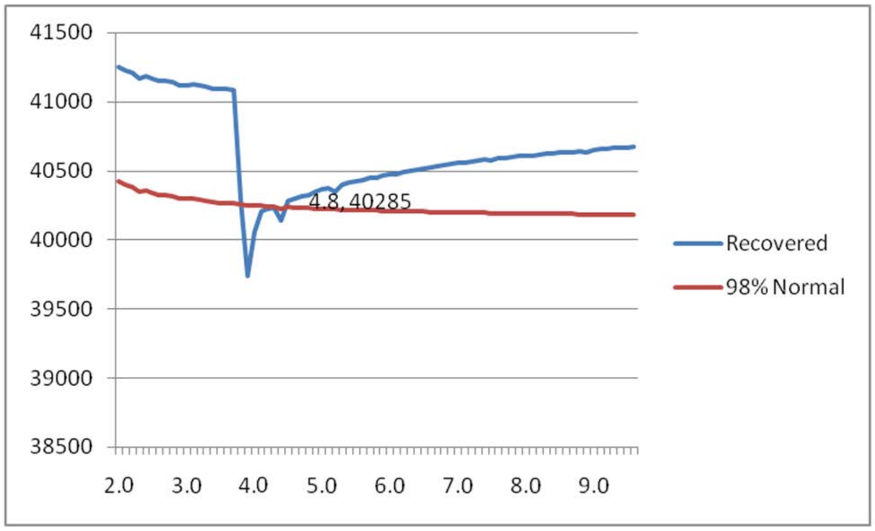


Figure 24- When 50% of nodes misbehaves in a WSN, the throughput reaches 98% of its normal value at 4.8s.

5.8 Analysis

The feedback time for each type of ad-hoc network is calculated in the different attack size: small, medium size and large (Table 5-3). In small attacks, in which 10% of ad-hoc nodes are affected and are shutdown as a response, the feedback time is 2.1 seconds for both WPAN and WSN and 0.3 seconds for SNU. When a medium size attack happens, which means 20% of nodes are affected, the average feedback time falls to 1.6 seconds for WPAN and 2.0 seconds for WSN, while rising to 0.9 seconds for SNU type of network. The feedback time further falls to an average of 0.7 seconds in WPAN and SNU and 0.8 seconds in WSN for large attacks. When the attack size is not known, the maximum calculated time for each type of network can be used as the feedback time. In comparison with the time used in the *antigen-receptor degeneracy* method (Schaust & Szczerbicka, 2011) which was 5 seconds, the maximum calculated time is 2.1 seconds. That is more than twice of what we have calculated which could save a lot of time and delay in the algorithm.

Table 5-3- The calculated feedback times (in seconds) for different ad-hoc networks

Network type / Attack size (% affected nodes)	WPAN	SNU	WSN
Small (10%)	2.1	0.3	2.1
Medium (20%)	1.6	0.9	2.0
Large (50%)	0.7	0.7	0.8

We see that in all three types of network, the feedback time becomes less and less as the number of shutdown nodes increases. An explanation for this could be because as the percentage of shutdown nodes increases, the availability of more options for re-routing falls greatly.

The recovery behaviour of WPAN and WSN are similar but SNU is different. The area in which the nodes are scattered could play a part in the results, as the area per node decreases the distance between nodes decreases, the number of neighbours of a node increases, leaving it with more options to re-route in case of a rout failure. The network density of WSN and WPAN are similar, but SNU's density is nearly half as much, which could explain the different behaviour of the SNU compared to the other two.

In all of the networks, after drawing the 30 runs of the simulation with the shutdown response call occurring to random nodes in the network, three sets of charts were visually distinguishable (Figure 13). A set of unaffected runs where the misbehaving nodes where not in the route of the flow and so the response had no effect on the flow. A set of non-recovering runs where the shutdown nodes were in a way that the flow was not able to reroute, and the recovered flows where the shutdown node was in the route but the flow had managed to find a new path and maintain the flow successfully. The percentage of these three sets changes in all three types of network, as the number of affected runs changes, this is shown in Table 5-4.

Table 5-4- The percentage of these three sets changes in all three types of network, as the number of affected runs changes

	WPAN			SNU			WSN		
⇩Behaviour	**10% off**	**20% off**	**50% off**	**10% off**	**20% off**	**50% off**	**10% off**	**20% off**	**50% off**
Recovered	7%	20%	23%	7%	20%	23%	15%	22%	27%
Not recovered	13%	27%	60%	17%	30%	54%	3%	10%	40%
Not affected	80%	53%	17%	77%	50%	23%	82%	68%	33%

The percentage of recovered data-flows increases as the attack size increases, but this increase is less than that of the total effected flows, or the non recovering ones. The percentage of not recovered flows is showing the same growth rate as the attack size. As expected the number of affected flows increases with the size of attack.

5.9 Conclusion

In this chapter we began by explaining our methodology and experiment design. We discussed that in order to have valid tests we needed to perform our experiments around 30 times, for each type of network while responding to different nodes each time. We began by performing a set of control experiments and worked out the point where the network stabilised. We then found the feedback time for each type of network using an average of the 30 different tests.

Interestingly we found that it is not the case that with an increase to the size of attack, the feedback time increases as expected, and in our experiments the overall trend was a decrease in feedback time. This could be because the number of test cases where the flow cannot recover increase greatly with the attack size. In calculating the feedback time we disregarded the cases where the data flow had not managed to reroute after the shutdown response because we needed to find out the average time it takes for the network to reroute if it was able to. If the attack size is known then its best to use feedback time as suggested by Table 5-3, otherwise the maximum time calculated for each type of network is preferred.

Chapter 6

Conclusions and Future Work

6.1 Overview

In order to enhance immune inspired intrusion response in ad-hoc networks we tried to calculate the minimum time needed to wait before a feedback was read from the network, called the feedback time.

In this thesis we began by studying ad-hoc networks and looking into their specific features and challenges for the security of ad-hoc networks. We concluded that any security solution for ad-hoc networks would need to be light-weight, adaptive and non-centralised. We then looked into bio-inspired methods for intrusion detection in ad-hoc networks, as they best fit the required properties for ad-hoc networks. We noticed that bio-inspired methods use feedback from the network in order to adapt to it, and are usually host based and light-weight. After reviewing the existing research on bio-inspired intrusion detection methods for ad-hoc networks, we focussed on the problem employing an adaptive response to a detected intrusion using bio-inspired methods.

The ARDA algorithm was then chosen as the prevalent bio-inspired intrusion response solution for ad-hoc networks, and as its most important part, the feedback procedure was concentrated on. We tried to overcome the problem of time to wait after a response was enacted, before taking feedback measures from the network. We categorised ad-hoc networks into three types: WPAN, SNU and WSN based on their size and application. After modifying ns2 to accommodate for the shutdown response in its implementation of the AODV routing protocol and recompiling it, we used it to simulate these three types of networks and calculate the effects of the shutdown response to a detected intrusion. We investigated response time in three attack sizes, where 10, 20 or 50 percent of nodes in the network were engaged in the attack and responded to using the shutdown response. We ran the simulation 30 times for each attack size and network type, and calculated an average response feedback time for each case by analysing the results using nam, gawk and Excel. Therefore we have achieved the objectives we set out to do.

6.2 Discussion

As our research questions we wanted to know if it was possible to predict the required time to wait before a feedback is taken; by dividing ad-hoc networks into three categories, we were able to test the reaction of each category to the response. In calculating the feedback time we disregarded the cases where the data flow had not managed to reroute after the shutdown response because we needed to find out the average time it takes for the network to reroute if it was able to. We were unable to see a clear trend in the feedback time that changed with the different types of networks.

We also varied the attack size to test its effect on the results. Interestingly we found that it is not the case that with an increase to the size of attack, the feedback time increases as expected, and in our experiments the overall trend was a decrease in feedback time. This could be because the number of test cases where the flow cannot recover increase greatly with the attack size. If the attack size is known, in the tested networks with the tested communications scenarios then its best to use feedback time as suggested by Table 5-3, otherwise the maximum time calculated for each type of network is preferred.

In comparison with the time used in the antigen-receptor degeneracy method (Schaust & Szczerbicka, 2011) which was 5 seconds, the maximum calculated time is 2.1 seconds. That is more than twice of what we have calculated which could save a lot of time and delay in the algorithm.

Briefly, this research has contributed in the following ways. This thesis calculates response feedback time for WSN, WPAN and SNU as the three main types of ad-hoc network. It also could be used to improve the bio-inspired *antigen-receptor degeneracy* method (ARDA algorithm) by suggesting the appropriate time lag to perform the feedback procedure. Moreover, this thesis upgrades the C++ implementation of the AODV routing protocol in ns version 2.35 by resolving an issue in ns2 that prevented node shutdown. This thesis then simulates the shutdown response to attacks in different types of ad-hoc networks.

This research can also benefit methods that require feedback after a change to the ad-hoc network either bio-inspired or non-bio-inspired. It can also help analysis of ad-hoc network resilience against a set of attacks, in which nodes are destroyed such as DoS attacks or do stability analysis of ad-hoc networks in case of node failure.

6.3 Future Work

We propose using a non-incremental method for calculating the throughput as future work for our project. This might benefit feedback time calculation in giving more exact times for the throughput. Testing the network for a feedback time based on different types of response could also improve the accuracy and better benefit an adaptive algorithm. We recommend running the ARDA algorithm with the feedback time calculated here for the shutdown response, and comparing results. We suggest devising a unified software that simulates and analyses the results with different parameters regarding the type of network and attack and number of runs, in order to produce an all purpose feedback timer that shows network recovery time for different purposes.

References

Aghaei, R. G., Rahman, M. A., Gueaieb, W., & Saddik, A. E. (2007). Ant Colony-Based Reinforcement Learning Algorithm for Routing in Wireless Sensor Networks. *Instrumentation and Measurement Technology Conference.*

Ahmed, E., Samad, K., & Mahmood, W. (2006). Cluster-based intrusion detection (cbid) architecture for mobile ad hoc networks. *AusCERT2006 Gold Coast, 5th Conference.* Australia.

Al Islam, A. A., Hyder, C. S., Kabir, H., & Naznin, M. (2010). Stable Sensor Network (SSN): A Dynamic Clustering Technique for Maximizing Stability in Wireless Sensor Networks. *Wireless Sensor Network* , 538-554.

Ananthapadmanabha, R., Manoj, B., & Murthy, C. (2001). Multi-hop cellular networks: the architecture and routing protocols. *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2001* .

Barr, R. (2012). *Introduction to JiST*. Retrieved from Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator : http://jist.ece.cornell.edu/

Brutch, P., & Ko, C. (2003). Challenges in intrusion detection for wireless ad-hoc networks. *IEEE Applications and the Internet Workshops*, (pp. 368-373).

Byers, P. K., & Bourgoin, S. M. (2005-2006). *Encyclopedia of World Biography.* Thomson Gale.

Chandola, D., Banerjee, A., & Kumar, V. (2009). Anomaly Detection : A Survey. *ACM Computing Surveys , 41* (3).

Chehri, A., Fortier, P., & Tardif, P.-M. (2007). Security Monitoring Using Wireless Sensor Networks. *Fifth Annual Conference on Communication Networks and Services Research, CNSR'07.* (pp. 13-17). IEEE.

Chen, G., Guo, T., Yang, W., & Zhao, T. (2006). An improved ant-based routing protocol in Wireless Sensor Networks. *Collaborative Computing: International Conference on Networking, Applications and Worksharing.*

Chow, C., & Yeung, D. (2002). Parzen-window network intrusion detectors. *The 16th International Conference on Pattern Recognition.* Washington, DC, USA: IEEE Computer Society.

Cordeiro, C., & Agrawal, D. (2006). *Ad Hoc & Sensor Networks: Theory and Applications.* World Scientific Publishing.

De Castro, L., & Timmis, J. (2002). Artificial Immune Systems: A New Computational Intelligence Approach. London: Springer-Verlag.

Djenouri, D., Khelladi, L., & Badache, N. (2005). A Survey of security Issues in Mobile Ad Hoc and Sensor Networks. *IEEE communications surveys , 7* (4).

Dressler, F. (2006). *Self-organization in ad hoc networks Overview and classification.* University of Erlangen, Dept. of Computer Science.

Dressler, F., & Akan O, B. (2010). A Survey on Bio-inspired Networking. *Computer Networks Journal , 54* (6), 881-900.

Drozda, M., Schildt, S., Schaust, S., & Szczerbicka, H. (2010). An Immuno-Inspired Approach to Misbehavior detection in Ad Hoc Networks. *arXiv preprint arXiv:1001.3113.*

Forooq, M. (2009). Bee-Inspired Routing Protocols for Mobile Ad Hoc and Sensor Networks. *Bee-Inspired Protocol Engineerin*, (pp. 235-270).

Forrest, S., & Hofmeyr, S. A. (2001). Immunology as Information Processing. In S. L.A., & I. Cohen, *In Design Principles for the Immune System and Other Distributed Autonomous Systems.*

Free Software Foundation, I. (2008). *GAWK*. Retrieved from http://www.gnu.org/software/gawk/

Greensmith, J., & Aickelin, U. (2008). The Deterministic Dendritic Cell Algorithm. *International Conference on Artificial Immune Systems (ICARIS 2008)*, (pp. 291-302).

Gu, Y., Lo, A., & Niemegeers, I. (2009). A Survey of Indoor Positioning Systems for Wireless Personal Networks. *Communications Surveys and Tutorials, IEEE* , 13-32.

Gupta, A., Sadawarti, H., & Verma, A. (2010). Performance analysis of AODV, DSR & TORA routing protocols. *IACSIT international journal of Engineering and Technology , 2* (2), 226-231.

Hofmeyr, S. (2000). An interpretive introduction to the immune system. In L. Segel, & I. Cohen, *Design Principles for the Immune System and Other Distributed Autonomous Systems* (pp. 3-26). Oxford University Press.

Hofmeyr, S. (2000). An interpretive introduction to the immune system. In L. Segel, & I. Cohen, *Design Principles for the Immune System and Other Distributed Autonomous Systems* (pp. 3-26). New York: Oxford University Press.

Hu, Y. C., Perrig, A., & Johnson, D. B. (2003). Rushing Attacks and Defense in Wireless Ad Hoc Networks. *2nd ACM workshop on Wireless security* , (pp. 30-40).

Hubaux, J., Buttya´n, L., & Capkun, S. (2001). The quest for security in mobile ad hoc networks. *2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing.* Long Beach, CA, USA.

IETF Working Group Charter. (1997). Retrieved from Mobile ad hoc networks(MANET): http: //www.ietf.org/html.charters/manet-charter.html

Issariyakul, T., & Hossain, E. (2012). *An introduction network simulator NS2.* Springer.

Jubin, J., & Tornow, J. (1987). The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE , 75* (1), 21-32.

Komninos, N., Vergados, D., & Douligeris, C. (2007). Detecting unauthorized and compromised nodes in mobile ad hoc networks. *Ad Hoc Networks* , 289-298.

Kong, J., Luo, H., Xu, K., Gu, D. L., Gerla, M., & Lu, S. (2002). Adaptive security for multilevel ad hoc networks. *Wireless Communications and Mobile Computing , 2* (5), 533-547.

Kuosmanen, P. (2002). *Classification of Ad Hoc routing protocols.* Finnish Defence Forces, Naval Academy, Finland, petteri. kuosmanen@ mil. fi .

Le Boudec, J., & Sarafijanovic, S. (2004). An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *ICARIS-3rd International conference on Artificial Immune Systems*, (pp. 342-356). Catania.

Li, W., & Joshi, A. (2006). *Security issues in mobile ad hoc networks: a survey.* Dept of Computer Science and Electrical Engineering, University of Maryland.

Luo, H., Ye, F., Lu, S., & Zhang, L. (2004). Security in mobile ad hoc networks: challenges and solutions. *IEEE Wireless Communications , 11* (1), 38-48.

MATLAB. (n.d.). Retrieved from http://www.mathworks.co.uk/products/matlab/

Morais, C., & Agrawal, D. (2006). *Ad Hoc & Sensor Networks: Theory and Applications.*

Murthy, S. R., & Manoj, B. (2004). *Ad Hoc Wireless Networks: Architectures and protocols.* Prentice Hall.

Murthy, S. R., & Manoj, B. S. (2004). *Ad Hoc Wireless Networks: Architectures and protocols.*

Sahu, S., & Shandilya, S. K. (2010). A comprehensive survey on intrusion detection in MANET. *International Journal of Information Technology and Knowledge Management* , 305-310.

Saleem, K. (2011). Ant based self-organized routing protocol for wireless sensor networks. *International Journal of Communication Networks and information security (IJCNIS) 1.2* .

Sarafijanovic, S., & Le Boudec, J. (2004). An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. *Bio-ADIT*, (pp. 96-111). Lausanne.

Sarafijanovic, S., & Le Boudec, J. (2005). An Artificial Immune System Approach with Secondary Response for Misbehavior Detection in Mobile Ad-Hoc Network. *IEEE Transactions on Neural Networks, Special Issue on Adaptive Learning Systems in Communi*, (pp. 1076-1087).

Sarafijanovic, S., & Le Boudec, J. (2005). An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *International Journal of Unconventional Computing*, (pp. 221-254).

Schaust, S., & Szczerbicka, H. (2011). Applying Antigen-Receptor Degeneracy Behavior for Misbehavior Response Selection in Wireless Sensor Networks. In *Artificial Immune Systems* (pp. 212-225). Springer.

Singh, S., & Yadav, R. S. (2007). A Review Paper on Ad Hoc Network Security. *International Journal of Computer Science and Security (IJCSS) , 1* (1).

Stakhanova, N., Basu, S., & Wong, J. (2007). A taxonomy of intrusion response systems. *International Journal of Information and Computer Security* , 169-184.

Stallings, W. (2009). *Cryptography and network security: principles and practice* (Fourth Edition ed.). Prentice Hall.

Timmis, J., Hone, A., Stibor, T., & Clark, E. (2008). Theoretical advances in artificial immune systems. *Theory of Computer Science* , 11-32.

Timmis, J., Lemos, R., Ayara, M., & Duncan, R. (2002). Towards Immune Inspired Fault Tolerance in Embedded Systems. *9th International Conference on Neural Information Processing* (pp. 1459-1463). IEEE.

Wu, H., Qiao, C., De, S., & Tonguz, O. (2001). Integrated cellular and ad hoc relaying systems: iCAR. *IEEE Selected Areas in Communications* , 2105-2115.

Zeng, X., Bargodia, R., & Gerla, M. (1998). GloMoSim: a library for parallel simulation of large-scale wireless networks. *Parallel and Distributed Simulation* (pp. 154-161). IEEE.

Zou, X., Ramamurthy, B., & Magliveras, S. (2002). Routing Techniques in Wireless Ad Hoc Networks Classification and Comparison. *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics, SCI 2002.*

Appendix

The Code describing a 50 node network to ns2 in tcl. Random nodes are shutdown as a response to misbehaviour. The number of data flows and misbehaving nodes are given at run time as arguments. Other sized networks are simulated using similar code, and the settings are changed accordingly.

```
###############################################################
# Input arguments
# argv 0 = rand seed,  argv 1= node off no.,   argv 2 = flow no.,     argv 3 =
nam file,      argv 4=result1 file
#
#################################################################

# 50 NODES with FIXED LOCATION in a 5*10 matrix, AODV routing protocol
# node shutdown now works in ns2 version 2.35.1, with a modification of the
function reset-state in the energymodel module, and a recompilation of ns2.

# Define options
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                         ;# max packet in ifq
set opt(row)        10;
set opt(col)        5;
set opt(nn)             [expr $opt(row)*$opt(col)] ;# number of mobilenodes
set opt(dist)       170;                    #distance between nodes
set opt(flow)       [lindex $argv 2]       ;# number of flows
set opt(sd)         [lindex $argv 1]       ;# number of shutdowns
set opt(offtime)    4.0                    ;# shutdown time
set opt(rp)             AODV                       ;# routing protocol
set opt(MaxX)           300                        ;# X dimension of topography
set opt(MaxY)           400                        ;# Y dimension of topography
set opt(stop)           10.0                       ;# time of simulation end
set opt(energy)         EnergyModel;
set opt(initialenergy)  11000; #10000;
set opt(txPower)        0.660;
set opt(rxPower)        0.395;
set opt(idlePower)      0.035;

# Source & Dest nodes for Flows
set opt(source01) 39;
set opt(dest01) 46;

set opt(source02) 0;
set opt(dest02) 4

set opt(source03) 20
set opt(dest03) 25


set ns          [new Simulator]
set tracefd       [open [lindex $argv 4] w]
set windowVsTime2 [open win6.tr w]
set namtrace      [open [lindex $argv 3] w]
$ns trace-all $tracefd
$ns use-newtrace
$ns namtrace-all-wireless $namtrace $opt(MaxX) $opt(MaxY)
```

```
# set up topography object
set topo       [new Topography]

$topo load_flatgrid $opt(MaxX) $opt(MaxY)


set god [create-god $opt(nn)]
set chan_1 [new $opt(chan)]


# configure the nodes
$ns node-config -adhocRouting $opt(rp) \
       -llType $opt(ll) \
       -macType $opt(mac) \
       -ifqType $opt(ifq) \
       -ifqLen $opt(ifqlen) \
       -antType $opt(ant) \
       -propType $opt(prop) \
       -phyType $opt(netif) \
       -channelType $opt(chan) \
       -topoInstance $topo \
       -agentTrace ON \
       -routerTrace ON \
       -macTrace ON \
       -movementTrace ON \
       -energyModel $opt(energy) \
       -initialEnergy $opt(initialenergy) \
       -txPower  $opt(txPower) \
       -rxPower  $opt(rxPower) \
       -idlePower  $opt(idlePower)


                 # seed the random generator function
expr srand([lindex $argv 0])

# Define node initial position in nam
for {set i 0} {$i < $opt(row) } { incr i } {
       for {set j 0} {$j < $opt(col) } { incr j } {
              set nodenum [expr ($i*$opt(col))+($j)]
              set node_($nodenum) [$ns node $nodenum]

              set xpos [expr ($i+1)*$opt(dist)]
              set ypos [expr ($j+1)*$opt(dist)]

       $node_($nodenum) set X_ $xpos
       $node_($nodenum) set Y_ $ypos
              $node_($nodenum) set Z_ 0.0

       }
}

# Defines the node size for nam
for {set i 0} {$i < $opt(nn)} { incr i } {
       $ns initial_node_pos $node_($i) 20
}

       set flow_start_time 1

# setup flow 1
set $i 0
       set udp_($i) [new Agent/UDP]
       set sink_($i) [new Agent/LossMonitor]
       set cbr_($i) [new Application/Traffic/CBR]
       $udp_($i) set packetSize_ 1500   # 1500 - 20 byte IP header = 1480 bytes
       $cbr_($i) set packetSize_ 1480  # This size INCLUDES the UDP header
       $cbr_($i) set interval_ 0.05000
       $cbr_($i) attach-agent $udp_($i)

       $ns attach-agent $node_($opt(source01)) $udp_($i)
       $ns attach-agent $node_($opt(dest01)) $sink_($i)
       $ns connect $udp_($i) $sink_($i)

       $ns at 1 "$cbr_($i) start"

# setup flow 2
set $i 1
       set udp_($i) [new Agent/UDP]
       set sink_($i) [new Agent/LossMonitor]
       set cbr_($i) [new Application/Traffic/CBR]
       $udp_($i) set packetSize_ 1500   # 1500 - 20 byte IP header = 1480 bytes
       $cbr_($i) set packetSize_ 1480  # This size INCLUDES the UDP header
       $cbr_($i) set interval_ 0.05000
       $cbr_($i) attach-agent $udp_($i)

       $ns attach-agent $node_($opt(source02)) $udp_($i)
       $ns attach-agent $node_($opt(dest02)) $sink_($i)
       $ns connect $udp_($i) $sink_($i)

       $ns at 1.2  "$cbr_($i) start"


set i 0
while { $i < $opt(sd) } {
       set j [expr round(rand()*($opt(nn)-1))]
        if { $j != $opt(source01)  &&  $j != $opt(source02)  &&   $j !=
$opt(dest01) && $j != $opt(dest01) && $j != $opt(dest02) } {
              $ns at $opt(offtime) "$node_($j) off"
              $ns at $opt(offtime) "puts  $j"
              incr i;
       }
}


# Telling nodes when the simulation ends
for {set i 0} {$i < $opt(nn) } { incr i } {
    $ns at $opt(stop) "$node_($i) reset";
}


# ending nam and the simulation
$ns at $opt(stop) "$ns nam-end-wireless $opt(stop)"
$ns at $opt(stop) "stop"
$ns at $opt(stop) "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}



$ns run
```

Gawk code run on the trace file produced by the simulation to determine network parameters such as throughput, delay or packet drop is shown below.

```
# - number of flows (senders)
# - time of simulation run
# - number of packets sent (at the Application)
# - number of packets received (at the Application)
# - number of packets dropped (at the Application)
# - number of collisions (802.11)
# - average delay
# - average throughput
# - average traffic rate (measured)
#-------------------------------------------------------------------------------
function average (array) {
    sum = 0;
    items = 0;
    for (i in array) {
       sum += array[i];
       items++;
    }
    if (sum == 0 || items == 0)
       return 0;
    else
       return sum / items;
}

#-------------------------------------------------------------------------------
function max( array ) {
    begin=1;
    for (i in array) {
       if (begin || array[i] > largest){
           largest = array[i];
           begin = 0;
       }
    }
    return largest;
}
#-------------------------------------------------------------------------------
function min(array) {
    begin=1;
    for (i in array){
       if (begin || array[i] < smallest) {
           smallest = array[i];
           begin = 0;
       }
    }
    return smallest;
}
#-------------------------------------------------------------------------------
function std_deviation(array, avg) {
    total = 0;
    items = 0;
    for (i in array) {
       delta = array[i] - avg;
       #print("i="i" array[i]="array[i]" delta="delta);
       total += delta*delta;
       items++;
    }
    if (total == 0 || items == 0)
       return 0;
    else
       return sqrt(total/(items-1));
}
#-------------------------------------------------------------------------------


#==============================================================================
#
BEGIN {
    if (!NEWTRACE) NEWTRACE="true";
    if (!PKTSIZE) PKTSIZE = 256;
    if (!NODE_INITIAL_ENERGY) NODE_INITIAL_ENERGY = 100.0;

    total_packets_sent = 0;
    total_packets_received = 0;
    total_packets_dropped = 0;
    first_packet_sent = 0;
    last_packet_sent = 0;
    last_packet_received = 0;

}
#
# ===============================================================================
#
{

    if (NEWTRACE == "true") {
#Example of new trace format
#1 2  3           4   5 6   7  8  9  10  11      12   13      14  15  16  17
18  19  20 21
#s -t 1.142046604 -Hs 0 -Hd -2 -Ni 0 -Nx 1186.47 -Ny 1093.76 -Nz 0.00 -Ne
100.000000 -Nl AGT -Nw ---
#
#22 23 24 25 26 27 28 29 30 31  32  33   34  35   36  37  38  39 40 41 42 43 44
45   46 47 48 49 50 51
#Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 40.0 -It cbr -Il 512 -If 0 -Ii 3 -Iv 32 -Pn
cbr -Pi 1 -Pf 0 -Po 16777215
       event = $1;
       time = $3;
       node = $5;
       type = $19;
       reason = $21;
       packetid = $41;
       packettype = $35;
       src = $31;
       dst = $33;
       cbr_packetid = $47; #NEW: para monitorar o delay pkt a pkt
       numhops = $49;
       opt_numhops = $51;
       energy  = $17;

# strip  trailing .0 or :0 from src and dst
       sub(/\.0$/, "", src);
       sub(/\.0$/, "", dst);

    } else {
#Example of old format
#1 2           3   4    5   6 7   8   9   10 11 12  13      14  15   16 17 18 19
20
#r 2.112017031 _5_ AGT  --- 7 cbr 532 [13a 5 4 800] ------- [1:0 5:0 30 5] [1] 2
2
       event = $1;
       time = $2;
       node = $3;
       type = $4;
       reason = $5;
       packetid = $6;
       packettype = $7;
       src = $14;
       dst = $15;
       cbr_packetid = $6; #ESTA ERRADO! :-(
       numhops = $19;
       opt_numhops = $20;

# strip leading and trailing _ from node
       sub(/^_*/, "", node);
       sub(/_*$/, "", node);
       sub(/^\[/, "", src);
       sub(/\:0$/, "", src);
       sub(/\:0$/, "", dst);

    }

if (time > 0 && time < duration/10.0)
{
#  print(event" -t "time" -Hs "node" -Hd "type" "reason" "packetid" "packettype"
"src" "dst" "energy" -Pf "numhops" -Po "opt_numhops);


    if ( time < simulation_start || simulation_start == 0 )
       simulation_start = time;

    if ( time > simulation_end )
       simulation_end = time;


    #-------------------
    #---   Application   ---
    #-------------------
    if ( type == "AGT" ) {
       nodes[node] = node; # to count number of nodes

       flow = src"-"dst;
       flows[flow] = flow; # to count number of flows

       cbr_flow_packetid = flow" "cbr_packetid; #used to calc end-to-end delay

       if ( time < node_start_time[node] || node_start_time[node] == 0)
           node_start_time[node] = time;

       if ( time > node_end_time[node] )
           node_end_time[node] = time;

       if ( time < flow_start_time[flow] || flow_start_time[flow] == 0) {
           flow_start_time[flow] = time;
           #print("flow_start_time["flow"]="time"\n");
       }

       if ( time > flow_end_time[flow] ) {
           flow_end_time[flow] = time;
           #print("flow_end_time["flow"]="time"\n");
       }

       #---
       #--- SEND
       #---
       if ( event == "s" ) {

           if ( time < first_packet_sent || first_packet_sent == 0 )
              first_packet_sent = time;
           if ( time > last_packet_sent )
              last_packet_sent = time;
# rate
           packets_sent[node]++;
           total_packets_sent++;
           packets_sent_flow[flow]++;
```

```
# delay
          pkt_start_time[cbr_flow_packetid] = time;
          ####TESTE @INRIA pkt_start_node[packetid] = node;

      }

      #---
      #--- RECEIVE
      #---
      else if ( event == "r" ) {
          if ( time > last_packet_received )
             last_packet_received = time;
# throughput
          packets_received[node]++;
          total_packets_received++;
          packets_received_flow[flow]++;
#    printf("r  %d %d at %f %f\n",node,packets_received[node]++, time, 
last_packet_received);

          pkt_received_per_time[ int(time/10) ]++;
# end-to-end delay
          pkt_end_time[cbr_flow_packetid] = time;
          ###TESTE @INRIA pkt_end_node[cbr_packetid" "flow] = node;

# num hops
          if (opt_numhops < 16777215) {
             packet_hops[packetid]     = numhops;
             opt_packet_hops[packetid] = opt_numhops;
          } else {
             num_opt_packets_unrech++;
          }

      }

      #new trace format
      node_final_energy[node] = energy;

    }

    #-------------
    #--- DROP
    #-------------
    if ( event == "d" || event == "D") {

       if(packettype == "cbr" || packettype == "tcp") {

          if (type == "IFQ" && reason == "ARP") {
             packets_dropped["ARP"]++;
          }
          else if (type != "MAC" || (type == "MAC" && reason == "RET")) {
             packets_dropped[type]++;
          }

          packets_dropped2[type" "reason" "packettype]++;

          if ( reason == "COL" )
             total_collisions++;
          else if ( type != "MAC" || (type == "MAC" && reason == "RET") ) {
             total_packets_dropped++;
             pkt_dropped_per_time[ int(time/10) ]++;
          }


       }else {
          packets_dropped2[type" "reason" "packettype]++;
```

```
            if ( reason == "COL" )
               total_collisions++;
         }
      }


      #-------------
      #--- ENERGY
      #------------
      #EXAMPLE: N -t 600.000100 -n 49 -e 8.719321
      if (NEWTRACE != "true" &&  event == "N" ) { # ENERGY stuff
            #    printf("%s %s %f %s %d %s %f\n",$1,$2,$3,$4,$5,$6,$7);
            node_final_energy[$5] = $7;
      }

   }
   }
   ##=================================================================

   END {

      number_flows = 0;

   # ===== TOTAL RUNTIME =====
      total_runtime = last_packet_sent - first_packet_sent;

   # ===== OFFERED LOAD ==== (TODO: revise here)
      if ( total_runtime > 0 && total_packets_sent > 0)
         load = ((total_packets_sent * PKTSIZE)/total_runtime) / 2000000; # no
   overhead

   # ===== FLOWS, RATE and THROUGHPUT  =====
      for (f in flows) {
         number_flows++;
         flowend = flow_end_time[f];
         flowstart = flow_start_time[f];
         flowtime = flowend - flowstart;
         if ( flowtime > 1 ) { #jm 27/10/2005, changed from > 0. Due to big
   values...
            throughput[f] = packets_received_flow[f]*PKTSIZE*8 / flowtime;
            rate[f] = (packets_sent_flow[f]*PKTSIZE*8) / flowtime;
            printf("%s %f %f %f %f\n", f, flowstart, flowend, rate[f],
   throughput[f]) >> filename; #"stats-throughput.dat";
   #printf("f: %d sent: %d rcvd: %d flowtime: %f \n", f, packets_sent_flow[f],
   packets_received_flow[f], flowtime);
         }
      }

   # ==== DROPPED ====
      for ( typereason in packets_dropped2) {
         printf("D %s = %d\n",typereason, packets_dropped2[typereason]) >> "stats-
   dropped.dat";
      }

   # ===== END-TO-END DELAY ====
      for ( flow_pkt in pkt_end_time) { #TODO: trocar para start para mudar estilo
   do end-to-end delay
         start = pkt_start_time[flow_pkt];
         end = pkt_end_time[flow_pkt];
          ##DISABLED @INRIA: node_start = pkt_start_node[flow_pkt];
         ##DISABLED @INRIA: node_end = pkt_end_node[flow_pkt];

         delta = end - start;
         if ( delta > 0 ) {
            delay[flow_pkt] = delta;
```

```
           printf("%f %s %f %f\n", start, flow_pkt, end, delta) >> "stats-
delay.dat";
       }
    }

# ===== ENERGY STUFF =====
    printf("#node initial final delta\n")>> "stats-energy.dat";
    for ( node in node_final_energy) {
       initial = NODE_INITIAL_ENERGY;  #node_initial_enrgy[node];   #GAMBI:
arrumar pra ler do ns2.tr
       final = node_final_energy[node];

       delta = initial - final;
       delta_energy[node] = delta;
       printf("%d %f %f %f\n", node, initial, final, delta) >> "stats-energy.dat";

    }
    min_node_usedenergy  = min(delta_energy);
    max_node_usedenergy  = max(delta_energy);
    avg_node_usedenergy  = average(delta_energy);
    printf("# usedenergy: avg=%8.4f,  min=%8.4f,
max=%8.4f\n",avg_node_usedenergy,min_node_usedenergy,max_node_usedenergy) >>
"stats-energy.dat";

# ===== NUMBER OF HOPS ====
    for ( pkt in packet_hops) {
       numhops = packet_hops[pkt];
       opt_numhops = opt_packet_hops[pkt];
       diff = numhops - opt_numhops;
       printf("%d %f %f %s\n", pkt,  numhops, opt_numhops,diff) >> "stats-
numhops.dat";
    }
    min_packet_hops  = min(packet_hops);
    max_packet_hops  = max(packet_hops);
    avg_packet_hops  = average(packet_hops);
    avg_opt_packet_hops  = average(opt_packet_hops);
    printf("#    avg=%8.4f min=%8.4f max=%8.4f\n",
          avg_packet_hops, min_packet_hops, max_packet_hops) >> "stats-
numhops.dat";
    printf("#opt avg=%8.4f min=%8.4f max=%8.4f\n",
          avg_opt_packet_hops, min(opt_packet_hops), max(opt_packet_hops)) >>
"stats-numhops.dat";
    printf("#Num opt packets unreacheble=%8.4f/%8.4f\n",
          num_opt_packets_unrech, total_packets_received) >> "stats-numhops.dat";

# ==== RCVD and DROPPED HISTOGRAM ====
    num_blocks = int(total_runtime/10);
    for ( b = 0; b <= num_blocks; b++) {
       sum_rcvd += pkt_received_per_time[b];
       # pkt received per time block
       printf("%d %d %d\n", b*10, pkt_received_per_time[b], sum_rcvd) >> "stats-
pkt_rcvd_per_time.dat";
       sum_drop += pkt_received_per_time[b];
       # pkt dropped per time block
       printf("%d %d %d\n", b*10, pkt_dropped_per_time[b], sum_drop) >> "stats-
pkt_dropped_per_time.dat";
    }


# ==== experimental stuff.... :-) =====
    if (avg_opt_packet_hops > 0) {
       rel_packet_hops = avg_packet_hops/avg_opt_packet_hops;
    }
```

```
# ===============
#      OUTPUT
# ===============
#    printf("\
##      RUN    OFFRD   #PKTS  PKTS  PKTS |PKTS  PKTS  PKTS  PKTS |         AVG
MAX      MIN      AVERAGE    AVERAGE     AVERAGE    .MAX   MIN   PATH      AVG
MAX        MIN      \
##FLOWS TIME   LOAD    #SENT  RCVD  DROP |D.IFQ D.ARP D.MAC D.RTR|COLL
DELAY(s)   DELAY(s) DELAY(s) TPUT (b/s) RATE(b/s)   HOPS       .HOPS  HOPS
OPTMLITY  ENERGY    ENERGY     ENERGY   \
##----- ------ --------#----- ----- -----|----- ----- ----- -----|-------- ------
---- -------- -------- ---------- ----------- ----------.----- ----- --------- -
-------- ---------- ---------\n");

    printf("%5d %6.1f %7.6f %5d %5d %5d %5d %5d %5d %5d %5d  %8.6f %8.6f  %8.6f
%8.6f  %8.6f %8.6f  %5d %5d %8.7f  %8.4f %8.6f %8.6f\n",
          number_flows,
          total_runtime,
          load,
          total_packets_sent,
          total_packets_received,
          total_packets_dropped,
          packets_dropped["IFQ"],
          packets_dropped["ARP"],
          packets_dropped["MAC"],
          packets_dropped["RTR"],
          total_collisions,
          average(delay),
          max(delay),
          min(delay),
          average(throughput),
          average(rate),
          average(packet_hops),
          max_packet_hops,
          min_packet_hops,
          rel_packet_hops,
          avg_node_usedenergy,
          max_node_usedenergy,
          min_node_usedenergy) >> "stats-summary.dat";

  # system("cat stats-summary.dat");
}
```

Here is the batch codes used to run the above code in order to get a throughput value for every 0.1s.

```
gawk -f empty.awk
i=1
nodeOff=2
flow=1
for((j=1; j<31; j++))
do
       for (( i= 3; i<100; i++))
       do
              nsfileName="20n/20n$flow.f-$nodeOff.Off-$j.tr"
       throughputFileName="20n/20n$flow.f-$nodeOff.Off-$j.dat"

              gawk -v duration=$i -v filename="$throughputFileName" -f parse4.awk
$nsfileName
       done
done
```