A Layered Security Approach for Cooperation Enforcement in MANETs

Sohail Abbas

July 2011

Thesis submitted to the School of Computing & Mathematical Sciences in partial fulfilment of the requirements for the Degree of Doctor of Philosophy

Liverpool John Moores University

School of Computing and Mathematical Sciences

Acknowledgements

First and foremost I must thank ALLAH, The Most Beneficent and The Most Merciful, for giving me the ability, strength and determination to do a PhD at this prestigious institution.

I am profoundly grateful to the mentorship of my supervisor, Professor Madjid Merabti. In addition to an excellent supervisor, he is a man of principles and I think myself fortunate enough to be one of his students. His constant motivation and guidance enabled me to think and write more clearly and to face audaciously the hurdles during my research. I am also deeply obliged to Dr. David Llewellyn-Jones for his precious time that he dedicated to my research work and discussion on ideas that eventually helped solidify many of the new concepts presented in this thesis. I am also thankful to Dr. Kashif Kifayat who spent his valuable time in helping me in my testbed experiments.

I am very thankful to my parents and other family members for their prayers and support which enabled me to successfully complete my PhD.

I would like to thank all my friends for their support, especially Muhammad Zahid Khan, Haseeb-ur-Rahman, Waqar Ahmad, Arshad Ali, Shafiq-ur-Rehman, Ali Mohmand, Mohammad Haroon, Adnan Sultan, etc., because of their company I did not feel the hardships of being abroad.

Special thanks go to the staff and technicians of the School of Computing and Mathematical Sciences, Liverpool John Moores University for their assistance and providing me with a pleasant work environment along with excellent computing facilities.

General thanks to the University of Malakand and Higher Education Commission (HEC) Pakistan for supporting me financially during my PhD.

I would also like to thank Sonja Buchegger from Deutsche Telekom Laboratories, Berlin Germany, for contributing NS-2 source code in support of the simulations experiments carried out during my research.

Abstract

In fully self-organized MANETs, nodes are naturally reluctant to spend their precious resources forwarding other nodes' packets and are therefore liable to exhibit selfish or sometimes malicious behaviour. This selfishness could potentially lead to network partitioning and network performance degradation. Cooperation enforcement schemes, such as reputation and trust based schemes have been proposed to counteract the issue of selfishness. The sole purpose of these schemes is to ensure selfish nodes bear the consequences of their bad actions. However, malicious nodes can exploit mobility and free identities available to breach the security of these systems and escape punishment or detection. Firstly, in the case of mobility, a malicious node can gain benefit even after having been detected by a reputation-based system, by interacting directly with its source or destination nodes. Secondly, since the lack of infrastructure in MANETs does not suit centralized identity management or centralized Trusted Third Parties, nodes can create zero-cost identities without any restrictions. As a result, a selfish node can easily escape the consequences of whatever misbehaviour it has performed by simply changing identity to clear all its bad history, known as whitewashing. Hence, this makes it difficult to hold malicious nodes accountable for their actions. Finally, a malicious node can concurrently create and control more than one virtual identity to launch an attack, called a Sybil attack. In the context of reputation-based schemes, a Sybil attacker can disrupt the detection accuracy by defaming other good nodes, self-promoting itself or exchanging bogus positive recommendations about one of its quarantined identities.

This thesis explores two aspects of direct interactions (DIs), *i.e.* DIs as a selfish nodes' strategy and DIs produced by inappropriate simulation parameters. In the latter case DIs cause confusion in the results evaluation of reputation-based schemes. We propose a method that uses the service contribution and consumption information to discourage selfish nodes that try to increase their benefit through DIs. We also propose methods that categorize nodes' benefits in order to mitigate the confusion caused in the results evaluation. A novel layered security approach is proposed using proactive and reactive paradigms to counteract whitewashing and Sybil attacks. The proactive paradigm is aimed at removing the advantages that whitewashing can provide by enforcing a non-monetary entry fee per new identity, in the form of cooperation in the network. The results show that this method deters these attackers by reducing their benefits in the network. In the reactive case, we propose a lightweight approach to detect new identities of whitewashers and Sybil attackers on the MAC layer using the 802.11 protocol without using any extra hardware. The experiments show that a signal strength based threshold exists which can help us detect Sybil and whitewashers' identities. Through the help of extensive simulations and real-world testbed experimentations, we are able to demonstrate that our proposed solution detects Sybil or whitewashers' new identities with good accuracy and reduces the benefits of malicious activity even in the presence of mobility.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background

The trend of wired networking has changed recently with the emergence of sophisticated wireless communication technology. Technology further brought significant advancements to mobile computing and as a result, mobile devices have gained sufficient computation, communication and memory resources to become widely interconnected. These advancements in the field have enabled people to use their computational devices and gadgets while on the move. The rapidly advancing wireless technologies have allowed the creation of one of the important generations of wireless networks, *i.e.* multihop mobile ad hoc networks (MANETs).

A mobile ad hoc network is a collection of nodes forming a temporary or permanent network without relying on any centralized architecture or control. Nodes can join or leave the network at any time and they can freely roam across the network. As MANETs do not rely on any centralized architecture, such as access points, base stations or remote servers, all the necessary network functionalities are performed by the nodes forming the network. Each node acts as a host as well as a router, relaying data to extend the range by establishing connectivity between the source and destination nodes that do not fall in direct range of each other. Distributing network functionalities, such as packet forwarding, to all the nodes increases robustness and avoids a single point of attack or failure. The avoidance of a centralized architecture also augments the MANETs' ability to support a wide variety of applications on low cost hardware with less time required to setup an infrastructure. Due to these flexibilities, it is tempting to use MANETs in situations where there does not exist a pre-deployed infrastructure or where it is costly to deploy an infrastructure such as in disaster relief scenarios, search and rescue operations, vehicular networks [1], casual meetings, video conferencing, personal area networks (PANs), campus networks, robot networks, and so on.

1.2 Cooperation in Self-Organized MANETs

In the last few years, self-organized systems have attained widespread attention in terms of research and deployment. These systems are typically organized according to the principle of Peer-to-Peer (P2P) organization, in which participants of the system have equivalent capabilities and responsibilities, *i.e.* all nodes are peers. Internet based P2P and mobile ad hoc networks are the two major examples of P2P communication systems. These systems have no centralized control and hence their main problem is cooperation. The participants are primarily concerned with their own benefit, because there is an incentive for peers to only consume network resources and services without contributing them back to the network. As a result, fairness and cooperation cannot be guaranteed. In a P2P context this conservative nature of participants is called *free-riding* [2-5] whereas in MANETs it is termed as *misbehaviour* or *selfishness* [6-10]. For example, in Internet based P2P networks, peers may not be inclined to provide bandwidth whereas in MANETs the users may not want to spend their own limited battery to forward others' packets. A promising solution to overcome this problem is to use reputation-based systems in order to give users some sort of incentive. However, due to the different characteristics of P2P and MANET systems, reputation-based schemes developed for them are different. MANETs are organized according to the same principle as P2P, *i.e.* nodes are autonomous (independent of any infrastructure), decentralized and self-organized. However, they have some special issues that are not present in P2P networks, such as limited battery power, mobility, wireless links, the broadcast nature of transmissions and the provision of packet forwarding services [11]. Due to these different characteristics of MANETs, we are not going to discuss reputation-based systems that have been proposed for P2P systems. Throughout this thesis, we will only discuss reputation-based schemes that have been developed for MANETs.

Most of the routing protocols developed for MANETs, such as Dynamic Source Routing (DSR) [12], Ad hoc On-demand Distance Vector (AODV) [13] and Destination-Sequenced Distance-Vector (DSDV) [14] are based on the multihop assumption: each node must cooperate and forward other nodes' packets to extend the range and enhance network performance. The multihop assumption is valid for *closed* or *managed* MANETs where nodes belong to a single authority (such as the military); however, this assumption may not be properly followed in a fully self-organized *open* MANET where nodes have their own domains and objectives. Since data transmission is the most expensive function in a wireless environment as compared to other functions such as data processing, nodes are naturally reluctant to spend their precious resources forwarding other nodes' packets and can therefore exhibit selfish or sometimes malicious behaviour in open MANET environments. Nodes that utilize the services provided by other nodes but that do not contribute their services to the network with the intention to save their resources are called *selfish* nodes. This is also called *misbehaviour* because selfish nodes do not follow the predefined duties of the network, for instance, relaying other nodes' packets. This selfish behaviour of nodes could potentially lead to network partitioning[1] and network performance degradation. Some authors, such as Agrawal *et al.* [15], have shown that a small percentage of selfish nodes can disrupt the whole network and severely degrade network performance. One can object that regular users don't usually have the appropriate skills and knowledge to modify the software or hardware functionalities for their own goals or interests and act selfishly. Hubaux *et al.* [16] answers this by pointing out that "criminal organizations can have enough interest and resources to reverse engineer a node and sell tampered nodes with modified behaviour on a large scale."

To enforce cooperation and discourage misbehaviour of nodes in MANET, three[2] major schemes have been proposed: a) Reputation-based, b) Trust-based and c) Credit based models. By utilizing the past behaviour of end-users, reputation and trust based schemes enable a node to decide whether other nodes are trustworthy and cooperative. Eventually

[1]By partition here we mean logical partitioning, which is not the same as topological (physical) partitioning. These logical partitions cause service holes in the network.

[2]There is another category of schemes, called Game Theoretic models, but we will not discuss them in this Thesis.

nodes having high reputation or trust are given services and nodes having low reputation or trust are isolated from the network. In credit-based schemes nodes usually pay for services; payments are made in the form of virtual currency. Nodes are buyers and/or sellers of the packet forwarding services. Nodes require credit to forward their packets.

Credit-based models are not scalable because they need a centralized virtual bank to regulate the transactions. Apart from the central virtual bank each node is equipped with tamper proof hardware for enhanced security. Due to these reasons credit-based schemes are not suitable for MANETs. Reputation or trust-based models on the other hand do not need any centralized entity or any tamper proof hardware and hence can be implemented in a fully distributed manner to increase scalability, thereby making them much more suitable for use in MANETs. Since our focus is on reputation-based schemes, we will not discuss credit-based schemes further, for a detailed survey interested readers are referred to Agrawal *et al.* [17] and Mandalas *et al.* [18].

1.3 Issues in Reputation-based Schemes

In the last decade, substantial research has been undertaken on the important issue of how to enforce cooperation in mobile ad hoc networks. Reputation/trust-based and credit-based models are the major cooperation enforcement models that have been proposed in the literature for mobile ad hoc networks. Due to the distributed nature of reputation-based systems, they are deemed to be a promising solution to enforce cooperation in MANETs. The main purpose of these schemes is to ensure that selfish nodes bear the consequences of their misbehaviour. However, after being quarantined, these discredited nodes apply certain strategies in order to escape the consequences or increase their benefits thereby promoting lack of accountability in the network. These nodes can exploit certain weaknesses of the MANETs, such as free identities available to them, or weakness of the reputation system, such as the fact they detect identities but not distinct nodes, or take advantages of mobility in order to escape consequences. Due to these issues, cooperation enforcement schemes must be secured for best results. The following sections detail the major issues or strategies that selfish nodes can maliciously apply for their own benefit.

1.3.1 Direct Interactions

A selfish node after being detected by a reputation-based system as a misbehaving node can still obtain benefit (throughput and utility) by interacting *directly* with the source or destination nodes of their communication by exploiting mobility. This gaining of benefit can occur due to the fact that there is no *intermediate* node along the path to drop the selfish nodes' data packets. Packet dropping is the strategy used by reputation-based schemes to decrease selfish nodes' throughput and utility. The direct interactions (DIs) have the effect of significantly improving the overall throughput and utility of both the good nodes as well as selfish nodes. DIs therefore cause two main problems. First, the detected selfish nodes can strategically select locations where they can directly interact with their source or destination. As discussed above, the motivation here is not to let the intermediate nodes be involved in the communication because, in reputation-based systems selfish nodes are isolated from the network by the intermediate nodes. Second, since high DIs can cause increased overall throughput and utility of a network, the confusion might be caused during the reputation-based systems evaluation, if they are not taken into account. For example, DIs make it unclear whether the increase in good nodes' throughput and utility is due to the efficiency of the reputation-based scheme or it is due to the high DIs produced in the network.

1.3.2 Identity based Attacks

Like every detection scheme, reputation-based systems require identities to be persistent as well as distinct. The persistence characteristic of identity implies that identities will be for long term use, in other words the lifetime of an identity should be long enough to make it hard for a selfish node to use it for short-term benefits. Otherwise a selfish node can simply discard an identity to escape the punishments imposed due to a poor reputation. The distinctness of identity implies that there will be only one identity tied to a single physical device or, conversely, a node should not be able to create more than one identity on a single physical device. For example, communications in wireless networks are usually based on a unique identity that represents a network entity: a node. Identities are used as an address to communicate with a network entity. This forms a one-to-one mapping between an identity and an entity and that is usually assumed either implicitly or explicitly by many mechanisms; hence two identities implies two distinct nodes [19]. Unfortunately malicious nodes can illegitimately claim multiple identities and violate this one-to-one mapping of identity and entity philosophy. Douceur [20] termed this as a Sybil attack, in which an attacker manages to create and control more than one identity on a single physical device.

In order to impose identity persistence and distinctness a variety of methods have been proposed which is discussed in Section 3.6. One common technique is to use centralized Trusted Third Party (TTP), as proposed by Hoeper and Gong [21], which is responsible for the identity management in the network. However, due to the centralized nature and costly configuration, in terms of money and initial setup, these systems are not suitable for MANETs. Lack of identity persistence and distinctness can cause a variety of problems in reputation-based systems, which we consider below.

As mentioned above, one of the main purposes of reputation-based schemes is to let selfish nodes bear the consequences of their bad actions. However, the open nature of MANETs enables a malicious or selfish node to change its identity and start again with a fresh (new) identity; in this way a selfish node whitewashes its previous misbehaving history. This is called a whitewashing or identity changing attack. Non-persistent identities make it difficult to hold malicious or selfish nodes accountable for their actions.

In the context of reputation-based schemes, a Sybil attacker can disrupt the detection accuracy by defaming other good nodes, self-promoting itself or exchanging false positive recommendations about one of its quarantined identities. Perrig *et al.* [22] identified the following security protocols of ad hoc networks that can be affected by Sybil attacks.

- *Distributed Storage*: A Sybil attack can disrupt the architecture where data is usually replicated or fragmented on several nodes, one such example is the use of distributed hash tables, because in reality data will be stored on Sybil identities generated by the same malicious node.
- *Routing*: in those routing mechanisms where nodes are supposed to be disjoint, as is the case in multipath routing and location based routing, are affected by Sybil identities because one node will be on various paths and/or in different locations at the same time.

- *Data Aggregation*: In a sensor network due to the lack of resources, data is often aggregated to one node. A Sybil node can change the whole aggregation reading outcome by contributing many times as a different user.
- *Voting*: In ad hoc and Peer-to-Peer networks decisions can be made using voting; however in this case a Sybil node can control the result by rigging the polling process using multiple virtual identities.
- *Misbehaviour Detection*: In misbehaviour detection schemes a Sybil node can increase its reputation, credit, or trust value and can decrease the same value for the other legitimate nodes by exploiting its virtual identities. Eventually this can diminish the detection accuracy of intrusion detection systems.
- *Traffic Congestion in a Vehicular Ad hoc Network (VANET)*: A malicious attacker can create the illusion of traffic congestion by spreading false information in a VANET. A malicious attacker can create an arbitrary number of virtual non-existent vehicles and transmit false information in the network to give a phoney impression of traffic congestion and eventually un-necessarily divert traffic [23].

1.3.3 The Difference between Whitewashing and Sybil Attacks

The main difference between whitewashing and Sybil attacks is the notion of *simultaneity*. The whitewasher creates new identities while discarding its previous identity, hence only one identity is up at a time in the network. The Sybil attacker, on the other hand, creates and controls identities but simultaneously without discarding its previously active identities. Some of the authors, for example Tangpong *et al.* [24] consider them both as Sybil attacks with two flavours, *viz.* either the creation of identities one-by-one or simultaneously; regardless of their applications, which are different (as discussed above). Our proposed methods in this thesis for thwarting these attackers are to *deter* and *detect* every new identity created by these attackers, regardless the nature of their creation *i.e.* simultaneously or one-by-one. Due to this reason, the work presented in this thesis is mainly focussed on whitewashing and all of the simulation-based and testbed experiments that we present are for whitewashing attacks. However, our proposed schemes can equally be applied for the counteraction of Sybil attacks as well.

These attackers can get identities by two ways. First, they can fabricate identities (for example, creating an arbitrary identifier). Second, they can use stolen identities, *i.e.* spoof the identity of a legitimate node in the network. In this thesis, we assume the first case where nodes can create arbitrary identifiers because in MANETs, there are no restrictions on identity creation.

1.4 Aims and Objectives

Reputation-based schemes like other cooperation enforcement schemes have been proposed to counteract selfish nodes by detecting them and ensuring them bear the consequences of their bad actions. However, selfish nodes can escape the detection or still gain benefits after being detected. In essence, these schemes are not secured and malicious nodes can apply certain strategies to make their detection and punishment mechanism ineffective. The aim of this thesis is to provide security for reputation-based systems in order to make them detect and isolate malicious nodes effectively. So that malicious nodes may not bypass the detection mechanism by interacting directly with their sources or destinations, changing identities (whitewashing) or creating more than one identity on a single physical device (Sybil attack). Since identity-based attacks are more serious than the direct interactions, we will deeply focus on them by proposing a layered security approach that incorporates two line-of-defences, *i.e.* proactive and reactive security paradigms. The countermeasures proposed for these attacks should be distributed and should not involve any trusted third party for their operations, in order to make them suitable for open MANET environments. The following are objectives for achieving our aim.

- In order to tackle the issue of DIs, we will look at DIs from two different perspectives or facets: (i) reputation system security and (ii) bias in their experimental results. In the reputation system security perspective, we will look at how to thwart or discourage those selfish nodes that are trying to increase their benefit by strategically choosing locations so that they can interact directly with their source or destination. In the second one, we will investigate the effect of DIs on certain metrics in mobile environments. Furthermore, we will propose methods to reduce the effect of DIs and resolve the confusion that would otherwise be caused.

- Identity-based attacks, such as whitewashing and Sybil attacks, pose a serious threat to the fair and successful working of reputation-based schemes. For example, a whitewasher can bypass detection by simply switching over to a new identity. A Sybil attacker can disrupt the detection accuracy by defaming other good nodes, self-promoting itself or exchanging false positive recommendations about one of its quarantined identities. Counter measures need to be developed for whitewashers and Sybil attackers to safeguard reputation-based schemes. As a first line-of-defence, these attackers should be discouraged in a way that does not cause extra overhead for the system. However, if attackers still dare to attack the system, a second line-of-defence should be in place to detect and isolate such malicious nodes in the network. Regarding these attacks, we will look into the following questions in this thesis. Are these attacks a threat to the system? Can these attackers be discouraged or detected using efficient methods? What are the techniques in the literature proposed to counteract them and what are their flaws? Are the existing proposed methods suitable for MANETs? Can they be thwarted using a layered security approach?

1.5 Novel Contributions

- We propose a method that uses service contribution and consumption information of nodes in order to discourage those selfish nodes that try to increase their benefit using DIs. Whereas, for DIs that are not caused by selfish nodes but caused by mobility and inappropriate simulation parameters, we first investigate their effect on certain metrics that are usually used in reputation systems evaluation, such as throughput, utility, network overhead and delay. We propose proactive, pre-simulation measures in order to reduce them in simulations and also propose to categorize throughput and utility to mitigate the confusion caused by DIs during evaluation. We adopt a widely used reputation-based scheme, called CONFIDANT [25-26] as our case study for our extensive simulations of DIs. As far as we are aware, no previous work has considered this issue in such context. Rather, some authors, such as [27] demonstrated that mobility can increase network throughput in the presence of selfish nodes; however, they did not show the reason of direct interactions and its consequences.

The idea (along with a survey on reputation-based schemes) has been published and given in Appendix E (E.3 and E.5).

- We propose a non-monetary fee based scheme for MANETs that acts as a deterrent for whitewashing attacks, acting as a first line-of-defence of our layered security approach. Rather than trying to detect whitewashing attacks, we approach the problem in a novel way by removing the advantages that whitewashing can provide. In our proposed scheme, each node must pay an entry fee to consume network services. As monetary fees are not suitable for MANETs due to certain problems such as fee management complications, rather than a monetary fee we use a fee in the form of cooperation, *i.e.* packet forwarding. Fee imposition makes whitewashing costly (in terms of battery power) for an attacker, the same is the case with Sybil attackers; hence an attacker can perform fewer whitewashes or generate fewer identities within the limits of its battery capacity. Furthermore, this form of fee enforcement will also improve the overall system performance (*i.e.* network throughput and utility). The simulation results show that our scheme work better than the widely used reputation-based scheme, called CONFIDANT, in discouraging the attackers by reducing their throughputs and utilities. We published this idea as listed in Appendix E (E.1, E.4 and E.6).
- We propose a lightweight approach to detect the new identities of whitewashers and Sybil attackers, as a second line-of-defence of our layered security approach. The scheme works on the MAC layer using the 802.11 protocol without the need for any extra hardware. We investigate the received signal strength of a node to distinguish between a legitimate new node and a whitewasher's new identity. We conducted several experiments to show that a signal strength based threshold exists which can help us detect a whitewasher's identities. We demonstrate through the help of extensive simulations and real-world testbed of Sun Spot sensors that our proposed solutions improves the overall system performance thereby detecting such malicious nodes with good accuracy and reducing further their benefits even in the presence of mobility. Part of the idea has been published and given in Appendix E (E.2).

1.6 Thesis Organization

This thesis is organized as follows.

Introduction (Chapter 1): in this chapter we highlight the potential issue of selfishness in self-organized MANETs and the main schemes to counteract this issue. We then discuss the security issues of reputation-based schemes, such as direct interactions, whitewashing and Sybil attacks. Finally we outline the aims and novel contributions of the thesis and thesis organization.

Background (Chapter 2): this chapter encompasses preliminary information about general network security and includes some discussion about why traditional network security techniques are not appropriate for use in mobile ad hoc networks. Furthermore, we discuss routing modes of operation in mobile ad hoc networks, describe in detail the Dynamic Source Routing protocol (DSR) and discuss the threats posed to ad hoc routing protocols including the packet relay problem and selfishness. Finally, we discuss the main ad hoc network simulators used in the research community for protocol evaluation.

Cooperation Enforcement in MANETs (Chapter 3): this chapter covers cooperation enforcement schemes: reputation/trust and credit based schemes. Among these schemes we discuss which of these is more promising for use in MANETs. Finally, the limitations of the current schemes and the countermeasures for the Sybil and whitewashing attacks are discussed.

A General Reputation System (Chapter 4): in this chapter we discuss the basic building blocks of our proposed reputation-based scheme *i.e.* the monitor, reputation system and path manager. Finally, we evaluate the scheme in order to confirm its working because we use this system for our layered security approach.

The Effect of Direct Interactions (Chapter 5): in this chapter we demonstrate through simulation experiments the effect of direct interactions on reputation-based schemes. We adopt the widely used reputation-based scheme to evaluate this and identify the common misjudgements that may be caused by such direct interactions. We propose methods to mitigate its effect whether it is caused by inappropriate simulation parameters or a selfish node strategy.

Deterring Whitewashing in MANETs (Chapter 6): in this chapter we demonstrate where in the reputation range a whitewasher node can obtain the most benefits. We then propose a novel approach to discourage whitewashing by reducing these benefits. We show through simulation that our scheme works better than CONFIDANT in reducing the evil nodes' throughputs and utilities in the network.

Detecting Whitewashing in MANETs: A Reactive Approach (Chapter 7): we use a cross-layer approach to detect the new identities of whitewashers and Sybil attackers on the MAC layer using the 802.11 protocol without the need to use any extra hardware. Investigating the received signal strength of a node, we distinguish between a legitimate new node and a whitewasher's new identity. Through extensive simulations and real-world measurements, we show that our proposed solution improves the overall system performance thereby detecting such malicious nodes with good accuracy and reducing further their benefits even in the presence of mobility.

Conclusions and Future Work (Chapter 8): we conclude in this chapter by summarizing our research contributions and discuss future work.

Chapter 2 Background

In this chapter we will give a brief overview of network security and shed light on the concepts, components and protocols used throughout the rest of this thesis. In addition to the traditional network security, we will also review the security requirements of mobile ad hoc networks, their implications and the challenges in making them secure. Since routing is an important mechanism, especially for ad hoc networks where nodes act as a host as well as a router, for making a network operational; we will overview the modes of operation used by the routing techniques in mobile ad hoc networks with an elaboration of the Dynamic Source Routing (DSR) protocol that we use in this thesis. Furthermore, we will briefly describe the threats that endanger the routing process including the issue of selfishness and its consequences in terms of network performance degradation. Finally, we discuss various network simulators that are available to simulate MANETs.

2.1 Mobile Ad hoc Networks (MANETs)

A mobile ad hoc network (MANET) is a collection of nodes forming a temporary or permanent wireless network without relying on any centralized architecture or control. As shown in Figure 2-1, the nodes could be laptops, mobile phones, PDAs, web-enabled cell phones and so on. Nodes can join or leave the network at any time and they can freely roam across the network. As MANETs do not rely on any centralized architecture, such as access points, base stations or remote servers, all the necessary network functionalities are performed by the nodes forming the network. Each node acts as a host as well as a router, relaying data to extend the range by establishing connectivity between the source and destination nodes that do not fall in the direct range of one another. Distributing network functionalities such as packet forwarding to all the nodes increases robustness and avoids the creation of a single point of attack or failure. The exemption of a centralized architecture also augments the MANET's ability to support a wide variety of applications on low cost hardware with less time required to set up. Due to this flexibility, it is tempting to use MANETs in situations where there does not exist a pre-deployed infrastructure or where it is costly to deploy an infrastructure such as in disaster relief scenarios, search and rescue operations, vehicular networks, casual meetings, campus networks, robot networks, and so on.

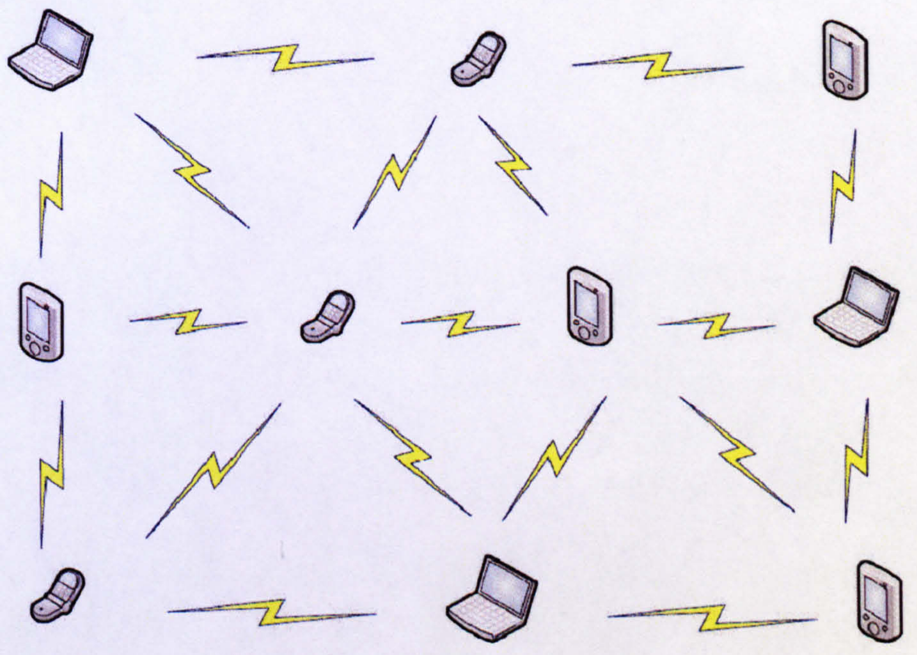

Figure 2-1: Mobile Ad hoc Network

The main characteristic of MANETs is that they are self-organized networks, since there is no central authority; nodes autonomously form the network quickly without any human intervention. The network autonomously determines its own configuration parameters: routing, addressing, *etc*. Another important characteristic of MANETs is the ability of a node to move freely while remaining connected to other nodes in the same network. However, nodes in these networks may be mobile, static or a mixture of both.

2.2 A Succinct Review of Network Security

In this section we skim through the basic security services that have been addressed in the research community over the years [10, 28-30].

Confidentiality maintains the secrecy of a message, even if the data travels via an insecure medium. It ensures the security of the content of a message communicated between authorised parties. The content of the message should not be disclosed to another party except to the sender and the receiver. Attacks on confidentiality include message interception, release of message contents to other parties, *etc.* as shown in Figure 2-2 (iii).

Confidentiality is difficult to maintain in mobile ad hoc networks due to the broadcast nature of the wireless links, which facilitates eavesdropping. As a result of this, various security mechanisms, including key distribution, are more difficult.

Authentication is an important service that verifies the identities of sending and receiving parties. The main attack on authentication is the masquerading attack- an attacker pretending to be a legitimate user. It is harder to detect such attackers in mobile ad hoc networks because there is no central authority controlling certificates and key distribution for identity authentication. A distributed authentication service is required in such networks.

Integrity of a message ensures that the message should not be modified in any sense including addition, deletion, introduction of unnecessary delays, *etc.* by unauthorized parties, as shown in the Figure 2-2 (iv).

Access Control ensures only legitimate users have access to resources. It restricts services, resources or data to users according to their access rights. It also enforces authorisation. To attack access control, an attacker can use masquerading, message interception and modification, or fabrication. The absence of infrastructure and random mobility makes it harder to detect corrupted nodes in a mobile ad hoc network. Some kind of distributed authentication management service is therefore needed to ensure access control.

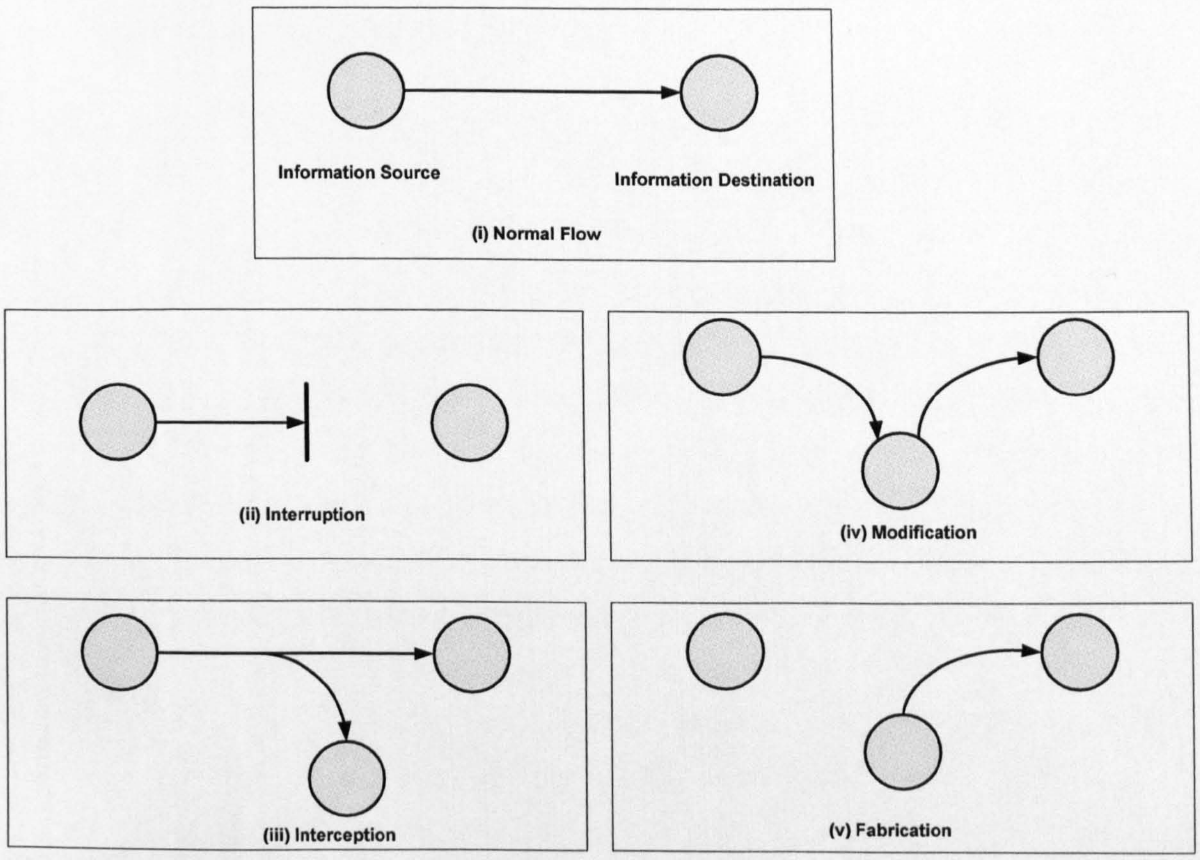

Figure 2-2: Security Threats

Availability ensures that services or devices are always available. An attack for disrupting availability is called a *denial of service* (DoS) attack which can be achieved by interruption of data, interruption of the network or overloading the server, as shown in Figure 2-2 (ii). One of the potential issues in mobile ad hoc networks is that nodes are usually low powered devices, so they can easily be attacked by sleep deprivation (keeping the CPU of a device engaged until the battery is exhausted) or incorrect packet forwarding that can cause a DoS attack.

Non-Repudiation is the ability for preventing a sender or a receiver of a message denying the sending or reception of a message. An attack on non-repudiation is masquerading.

2.3 Security Implications of MANETs

Wireless communications enables users and organizations to enjoy various functions, such as portability, flexibility and low installation costs, which they may not enjoy using traditional wired networks. Apart from these advantages, wireless communication also creates a new set of security implications. Some of these implications are the same as those for wired networks, some are caused by the wireless connectivity itself, and some are new, as (these implications are) pointed out by D. Djenouri *et al.* [10]. Srivatsa [31] categorises them as follows. First, the main risk in wireless networks is the ***broadcast nature*** of wireless connectivity itself which makes it easier for an adversary to eavesdrop on data or inject bogus data into the network. Second, the devices usually used in wireless networks are *resource constrained*, having limited bandwidth, memory, computational power, and battery power. They are a palatable target for a powerful adversary because they cannot use heavy cryptographic mechanisms for power saving reasons. Third, the absence of a *centralized Trusted Third Party (TTP)* or a certification authority in a wireless ad hoc network induces challenges for efficient trust and identity management. Fourth, in the case of ad hoc networks nodes are assumed to be *cooperative* and will relay others' packets; however a selfish or a compromised node may violate this assumption. Fifth, nodes in wireless networks are more vulnerable to *physical security* threats than those of wired networks, such as an enterprise server, because nodes in wireless networks are portable handy devices and may be easily stolen.

Security in MANETs is more challenging because apart from the lack of infrastructure, mobility of nodes causes ad hoc networks to have a dynamic, continuously changing topology. Furthermore freedom of movement offers flexibility for the potential attackers to move across the network to search for victims. Nodes may be small hand-held devices, such as PDAs, palmtops and smart phones, which can easily be compromised or stolen. The situation may become even worst when an adversary exploits a number of these compromised devices to launch an insider attack in a sensitive network, such as a military network. Due to the above mentioned constraints the security protocols used in existing wired networks may not be very effective in MANETs. MANET security protocols must be able to scale instantly and on the other hand they should not be affected by the dynamic topology. Protocols that need a lot of computational, memory or bandwidth resources are not suitable for MANETs because nodes usually have limited resources as compared to their wired counterparts.

MANETs were originally introduced as closed or managed networks that belonged to a single entity or organization called an offline authority, such as the military. In this case the end-users have a pre-established relationship and they work under this offline authority. The offline authority usually performs the initial set up procedures, for example to allot keys or certificates to nodes, and to allocate initial trust to the nodes, before network deployment. Since nodes belong to a single authority and all have a common objective, they are therefore motivated to cooperate. However, the proliferation of mobile communication devices such as laptops, PDAs, cell phones may produce a fully self-organized mobile ad hoc network, where nodes do not pertain to a single organization. End-users come together to form a network in a purely ad hoc manner forming an *open* or *pure* ad hoc network. Users are usually strangers without having any relationship and nodes have no pre-established security associations. They have different interests and objectives and they share their resources for global connectivity. The lack of a Trusted Third Party (TTP) and the existence of untrusted users in a fully self-organized MANET causes the following security problems, as highlighted by Mcdonald *et al.* [32].

- Fully self-organized MANETs are open in nature. Just like the Internet, nodes are liable to join and leave the network at random. Openness attracts selfish and malicious users.
- Each end-user will be its own authority domain, and therefore responsible for accomplishing distributed network functionalities, such as packet forwarding for other nodes, as well as generating and maintaining its own keying material.
- There will always be a threat of active insider attacks in the network.
- As Douceur [20] pointed out, in the absence of an offline TTP, a node can create and control more than one identity without any cost or difficulty, termed as a Sybil attack. As a result, a node can join the network every time under a different identity and hence it will be difficult to hold malicious nodes accountable for their actions.

In MANETs, routing is based on multihoping, *i.e.* in order to extend the range of communication each node is supposed to forward other nodes' packets. However, in open

MANETs nodes may not cooperate and may exhibit selfish behaviour by not providing packet forwarding services to other nodes. This selfishness or misbehaviour in packet forwarding degrades overall network performance. Cooperation enforcement schemes, such as reputation, trust and credit based solutions have been proposed to enforce cooperation in MANETs. Since routing is an important networking function these schemes (for example reputation-based schemes) improve routing by motivating nodes to cooperate in the packet forwarding process and restricting selfish nodes' services.

Our work is related to the improvement of the reputation-based system that is used ultimately to secure the routing process from selfish nodes. Furthermore reputation-based schemes have been proposed to work as an extension of routing protocols. Likewise, our reputation-based scheme, described in Chapter 4, is an extension of the DSR protocol. Broch *et al.* [33] have undertaken a performance comparison of four popular routing protocols and have established the result that DSR has the best throughput performance: more than 95% across all movement speeds and mobility rates. This motivates us to use DSR as the basic routing protocol for our proposed work.

2.4 Routing in MANETs

Routing in mobile ad hoc network is deemed to be the most critical part of network control. Since there is no infrastructure (routers etc.), routing protocols in ad hoc networks use mobile nodes to route packets from a source to a destination node using multihoping if the destination does not lie in the direct radio range of the source node. Node mobility can cause a highly dynamic network topology where wireless links are constantly established and pulled down due to the nodes' movements. As a result, conventional routing protocols designed for infrastructure or wired networks do not work for the network in ad hoc settings. Several routing protocols have been proposed in the literature for mobile ad hoc networks and performance-based comparative studies have been conducted by the research community to judge the properties of various solutions. However, in this section we will generally discuss routing protocols based on their mode of operations.

2.5 Mode of Routing Operation

Proactive *vs.* Reactive

Proactive and reactive modes are concerned with whether nodes in ad hoc networks should maintain routes to all possible destinations in the form of a table or should discover destinations on demand.

Proactive routing protocols store route information (before it is required) in a table hence they are sometimes referred to as table-driven routing protocols. Route updates are periodically broadcast by each node; consequently every node updates its routing table accordingly. These protocols have the advantage of experiencing minimal delays when communicating with arbitrary destinations in the network. However, their main disadvantage is the additional overhead of control traffic that is used to continually update stale route information. The situation gets worse when the control packet overhead increases in MANETs where links are often broken due to mobility. Furthermore, these protocols may not scale well for larger networks [34]. An example of proactive routing is Destination-Sequenced Distance-Vector (DSDV) [14].

Reactive protocols, on the other hand, generate routing information only on-demand without maintaining routes beforehand. This significantly reduces the control packet overhead. However, it causes delays when a node wants to communicate with another node for the first time. Route caches are used to store routes that may be used in the near future. Examples of reactive routing include DSR [12] and Ad hoc On-demand Distance Vector (AODV) [13].

Research has also been done on hybrid routing protocols that combine the advantages of both the proactive and reactive protocols. The hybrid protocols make use of the hierarchical network architectures. The reactive and proactive routing techniques are exploited in different hierarchical levels of the network. The example of the hybrid routing protocol is the Zone Routing Protocol (ZRP) [35]. However, these protocols may cause large communication overhead due to the topological changes in the network.

Source *vs.* Hop-by-hop Routing

These modes of operation are concerned with whether the source node of a packet decides the route for the packet to be forwarded to the destination node or the intermediate nodes decide the next hop until the packet arrives at the destination.

In source routing protocols, the source node is responsible for selecting the route and storing the route information in the packet header. All intermediate nodes follow the packet header to forward the packet *faithfully* along the path. These protocols have the advantage that intermediate nodes are not required to maintain the routing information. However, the packet size is increased due to the source routing information carried in each packet.

In hop-by-hop routing protocols source nodes are only required to know how to get to the next hop; subsequent intermediate nodes also find their next hops up to the destination node. These routing protocols reduce the packet header size but they require all the intermediate nodes to retain routing information.

As shown in Table 2-1, the four main routing protocols can be classified based on their mode of operation.

Table 2-1: Classification of Routing Protocols

Mode of Operation	Proactive	Reactive
Source Routing	ZRP	DSR
Hop-by-hop Routing	DSDV	AODV

2.6 The Dynamic Source Routing Protocol (DSR)

The DSR routing protocol is one of the main routing protocols developed for mobile ad hoc networks, proposed by Johnson *et al.* [12]. It is an on-demand routing protocol in which source nodes initiate a route discovery process to find the destination. The discovered route is then listed in the data packet's header, called the source route. Each node maintains a dynamic cache, called a route cache that is used to store routes to other destination nodes in the network. Routes can be determined through a route discovery process, forwarding packets along an active path and through overhearing using passive acknowledgments. The on-demand routing reduces the overhead caused by periodic route advertisement. As is evident from the title, DSR is a source routing protocol that allows source nodes to select and control multiple routes to a destination for routing packets. There are two main mechanisms in DSR: Route Discovery and Route Maintenance, which are discussed below.

Route Discovery

In DSR routes to destinations are determined through a route discovery process that is illustrated by Figure 2-3 and Figure 2-4. When a source node *S* wants to send a data packet (message) to some destination *D*, after saving the data packet in its send buffer it first searches its route cache to establish whether there is already a route to node *D*. If there is no route found in the route cache to *D*, then *S* will commence a Route Discovery process by broadcasting a Route Request (RREQ) control packet. *S* adds itself to the beginning of the route record before sending the message. As a result all nodes falling within the radio range of *S* will receive the RREQ packet, for example node *B* and *G* in Figure 2-3. The RREQ packet contains the source and destination nodes, a unique RREQ identifier and a list of all nodes along the path that have been traversed by this packet so far. Figure 2-4 shows the step by step processing of the RREQ and Route Reply (RREP) messages along the path, the red line in Figure 2-3.

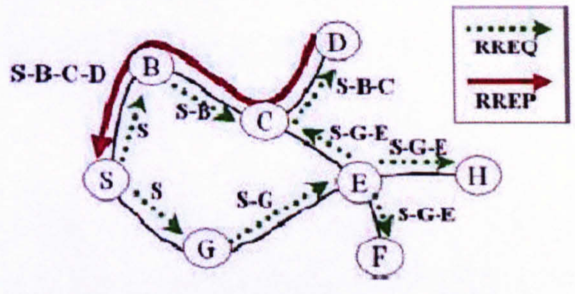

Figure 2-3: Route Discovery Example [36]

RREQ Broadcast	$S \rightarrow *$:	$< RREQ, S, D, S_{id}, () >$
	$B \rightarrow *$:	$< RREQ, S, D, S_{id}, (B) >$
	$C \rightarrow *$:	$< RREQ, S, D, S_{id}, (B, C) >$
RREP Unicast	$D \rightarrow C$:	$< RREP, D, S, S_{id}, (B, C, D) >$
	$C \rightarrow B$:	$< RREP, D, S, S_{id}, (B, C, D) >$
	$B \rightarrow S$:	$< RREP, D, S, S_{id}, (B, C, D) >$

Figure 2-4: Route Discovery Processing [36]

When a node receives a RREQ message, it compares the destination address contained in the message with its own address to establish whether the message is intended for itself. If the message is not destined for itself, it will append its address to the route record and broadcast the RREQ further on until the destination is reached or time to live (TTL) expires. In order to control the flooding and unlimited forwarding of RREQ messages in the network (*i.e.* to prevent loops), the DSR protocol specified that nodes should only forward the first copy of the same RREQ messages they receive.

If the destination node *D* receives a RREQ message, it will send a Route Reply (RREP) control packet to the source node along with the route record that has been accumulated during the route discovery process. If the destination node uses a MAC-based protocol (such as IEEE 802.11) that allows bidirectional links, it simply reverses the source route record in order to send the RREP back to the source node *S*. If the MAC protocol does not allow bidirectional links then the destination node should find a route back to the source node *S* by searching its route cache or by launching a route discovery process.

After receiving the RREP message, *S* saves the returned route into its route cache. In future, *S* will use this route to send all packets destined to node *D* until the route is broken.

Route Maintenance

Since the topology of a mobile ad hoc network changes frequently, the paths contained in the route caches of nodes are often broken. To prevent broken links, each node after forwarding a packet must confirm the reachability of the next hop. After a certain period of time if the node does not receive any confirmation from the next hop, it retransmits the packet. After a maximum number of retransmissions, if the node still does not receive any response from the next hop, the link to the next hop is thought to be broken and the node will send a Route Error (RERR) control packet back to the source node. The source node will start a new route discovery process.

In Appendix B we will show the DSR protocol structure in the NS-2 simulator.

There are three mechanisms that DSR proposes to confirm the data flow over the link from the node to the next hop:

- ***Network layer acknowledgements*** are used when a node sends an explicit acknowledgement request to probe its next hop node.
- ***Link layer acknowledgements*** are provided by the MAC layer protocol, such as in IEEE 802.11.
- ***Passive acknowledgements*** can be used to overhear (promiscuous listening) packet forwarded by the next hop node in order to confirm its reachability. Most reputation-based schemes monitor nodes for packet misbehaviour using passive acknowledgements; we will discuss these schemes in detail in Chapter 3.

2.7 Threats to Ad hoc Routing

Since routing in ad hoc networks is deemed as the most critical component of network control, substantial research has been undertaken in order to secure it. Security of routing becomes very challenging in a *pure* ad hoc network where nodes do not belong to a single authority and nodes do not have an *a priori* trust relationship. In Section 2.2 we discussed these challenges posed in mobile ad hoc networks in detail. Here in this section we will briefly discuss common attacks on ad hoc routing protocols, pointed by many authors such as [28-29, 37] that can be categorized into *Active* and *Passive* attacks.

Network attacks can be classified as active and passive attacks. In active attacks, an attacker carries out some actions, for example addition, deletion or delays. In passive attacks, on the other hand, an attacker acts as an observer to analyze data exchanged between communicating parties without any modification or intervention.

2.7.1 Active Attacks

- ***Black Hole Attack:*** in a black hole attack, a malicious node advertises the shortest path (zero routing metric) to all or some of the destinations. As a result, neighbours route their traffic towards the attacker node. The attacker then drops all these packets to create a denial of service attack in the network. In other situations an attacker advertises spoofed messages from the victim node to falsely show the shortest route to all destinations. Consequently, all nodes will direct their traffic towards the victim node to quickly deplete its battery power.
- ***Worm Hole Attack:*** the attacker receives data at one location and tunnels it to another location in the network causing invalidity of routes for the data.
- ***Resource Consumption Attack:*** in the network layer, a malicious node can generate or inject packets that can consume network resources uselessly. For example, an attacker can initiate a lot of route discoveries, route discoveries of bogus destinations, or the forwarding of outdated packets to nodes in the network.
- ***Route Cache Poisoning:*** this attack is used against on-demand routing protocols such as DSR in which route caches are poisoned to disrupt the whole routing of the network. A malicious node can inject spoofed packets with bogus route information into the network; every node overhearing this packet will save the route in its route cache. As a result, packets are routed on longer paths or through routing loops consuming network resources needlessly.
- ***Rushing Attack:*** a malicious node that receives a RREQ packet from a source node immediately responds to the source node before any other legitimate nodes may react. Any responses after it will be considered as duplicates and will be deleted.

Consequently, any route discovered by the source node will contain the malicious node information as one of the legitimate nodes.

2.7.2 Passive Attacks

- ***Eavesdropping:*** it is the act of secretly listening to the conversation or communication of other parties without their consent. Eavesdropping is one of the main issues in wireless networks; traffic can easily be eavesdropped due to the broadcast nature of wireless networks. In routing, mostly the routing information is communicated within the packets with the header in plain text that can be easily overheard and analysed to disclose the path to the source or destination or to compromise their location privacy. Cryptographic techniques are usually used to conceal data from eavesdropping.
- ***Selfishness:*** in multihop ad hoc networks, intermediate nodes act as a host as well as a router to forward packets for other nodes. As nodes have limited battery power, they are naturally inclined to save power by spending power only on their own network activities. Therefore it can cause selfishness in nodes and they may drop other nodes' packets. This is a passive attack [38] and also called passive Denial of Service (DoS) attack [39]; however, these packet drops can also be used as an active attacks, such as black hole attack. Since selfishness or passive DoS attack is one of the main themes of our work in this thesis, we will discuss this issue in more detail in the next section.

2.8 Packet Relay Problem

In self-organized *open* ad hoc networks, there is no strong authentication infrastructure nor any centralized authority to manage trust relationships among nodes. Consequently, the reliability of the basic network functions can be jeopardized by any node in the network. However, none of the classical security mechanisms is helpful in counteracting a misbehaving node. The proper network operation not only relies on the correct execution of critical network functions by the network nodes, but it also needs each node to cooperatively perform a "fair share" of the network functions: node cooperation, such as packet forwarding. Nodes are usually assumed to be cooperative in ad hoc networks.

In this thesis we consider malicious nodes as being distinct from selfish nodes. Malicious nodes intentionally damage others by breaking the proper network operation. This kind of misbehaviour can be tackled using traditional security services, such as authentication, integrity, *etc.* A malicious entity, for example, can launch a route disruption attack by injecting spoofed routing packets to partition the network or to create a routing loop. This kind of attack can be tackled using authentication and integrity to verify the routing control packets. Selfish nodes, on the other hand, do not intend to directly damage the functioning of the network, but are reluctant to spend their resources for others. This kind of misbehaviour is unique to self-organized networks and can be grouped under non-cooperation problems. Several techniques have been proposed to enforce cooperation and to counteract this kind of misbehaviour in ad hoc networks.

Cooperation enforcement schemes offer a reasonable solution to selfish nodes. In order to stimulate nodes to cooperate, these schemes offer incentives to nodes that cooperate in the form of reputation, trust or of virtual currency. We will discuss cooperation schemes in detail in Chapter 3, however here we will discuss the types of selfish nodes.

- ***Type 1: Selfish forwarding.*** This type of selfish node does not forward any data packet at all. When this type of behaviour is exhibited, all data packets that have a different source or destination address than the current node address are dropped. However, a selfish node following this model takes part in the Route Discovery and Route Maintenance phase of the DSR protocol. Since data packets are significantly larger in size than control packets, a selfish node dropping all data packets can save a significant fraction of its battery power while still contributing to the network maintenance.
- ***Type 2: Selfish routing.*** This type of selfish model focuses on selfish nodes that do not take part in the Route Discovery and Route Maintenance phase of the DSR protocol. A node not participating in the Route Discovery process will not be selected in any path and hence the packet forwarding services will automatically be disabled. A selfish node of this type will not be detected by any reputation-based scheme; however, work has been done on tackling them, such as in the work of Djenouri *et al.* [40-41].

- ***Type 3: Energy-driven selfish behaviour.*** This type of misbehaviour is more complex than the other two, and is discussed by Michiardi and Molva [38]. In this model, a selfish node changes its behaviour with respect to its battery power level. The node behaves well until an energy threshold is met then it starts behaving like the type 1 selfish node until the next lower energy threshold is reached, at which point it starts behaving like the type 2 selfish node.
- ***Type 4: Partial selfish behaviour.*** This type of selfish node partially drops packets; it can therefore be very tricky to detect in the network. Reputation-based schemes usually use a threshold to detect selfish nodes; selfish nodes of this kind may not cross the threshold.

We use only the type 1 selfish model throughout this thesis. This type of selfishness is much more serious and dangerous to routing protocols because in this model selfish nodes interrupt the data flow by dropping the data packets, hence forcing the routing protocol to restart the route discovery process. The route discovery process may not be initiated if an alternate route is available however; it may again contain some selfish nodes due to which it may also fail. This process continues until the source concludes that the data cannot be further transferred.

2.9 Network Simulators

There are mainly two ways to investigate MANETs (or its protocols): by using software-based simulators or conducting real network experimentation using a testbed. Testbeds are not usually favoured by most researchers due to their cost and lack of flexibility, especially when the scale of the experimental network increases. Software simulators are then considered as a viable alternative and are a widely adopted solution.

In order to study various simulators available for MANETs, it is important to specify the basic components required for MANETs, some of them are given below.

- *Nodes*: a set of mobile devices are required that act as a basic entity of a network and that should have the processing as well as wireless transmission capabilities.

- *TCP/IP protocol stack*: since nodes are communication devices, their internal architecture should be defined in terms of the TCP/IP protocol stack. In other words, nodes' communications are regulated by the functionalities of the TCP/IP protocol's layers, *i.e.* application, transport, network, MAC/data link and physical layers.
- *Wireless interface*: nodes should be equipped with radio devices that enable them to transmit and receive wirelessly.
- *Mobility*: nodes should have the ability to move freely across the network following random patterns or mobility models.
- *Data traffic*: nodes are required to establish connections with other nodes or group of nodes in the network. These connections will be in the form of data traffic flows with various transmission rates where nodes will act as sources and destinations of the data traffic.
- *Nodes as routers*: since, there are no dedicated routers in ad hoc network, such as MANETs, nodes are required to act as a host as well as a router in order to forward other nodes' packets.

Currently there are many simulators that can simulate a MANET. In this section we will briefly discuss the most commonly used network simulators. We will compare their merits and demerits and opt for one as a platform for our work in this thesis.

2.9.1 GloMoSim

GloMoSim was developed at the University of California, Los Angeles, USA. According to L. Hogie *et al.* [42] it is the second most popular network simulator but suffers from a lack of in-depth documentation. It provides a scalable simulation environment both for wired and wireless networks. It is built using a layered approach that mimics the OSI seven-layer network architecture.

GloMoSim is designed to simulate wireless communication protocols using a set of library modules that have been written using the parallel discrete-event simulation language, called PARSEC. New modules and protocols can be added to the library using the PARSEC language. The DSR protocol implementation is also available in GloMoSim.

GloMoSim is not entirely free, but its source and binary code can be obtained for research purposes by academic institutions only. The commercial version of GloMoSim is called QualNet and is based on the C++ language.

2.9.2 OPNet Modeler

OPNet (Optimized Network Engineering Tools) is a commercial discrete event network simulator used for network modelling and simulation. It was developed at MIT and introduced in 1987 as the first commercial network simulator. It is written in C++. The OPNet Modeler allows users to design and study networks, devices, protocols and applications in an interactive development environment. It provides a GUI-based environment simulating the network graphically in a way that mimics the structure and components of an actual network. It allows users to design the network visually without manually writing scripts for it. The development language is C/C++.

An object-oriented approach is used by the modeler: nodes and protocols are modelled as classes along with the inheritance and specialization facility.

2.9.3 NS-2

NS-2 [43], developed at ISI (Information Sciences Institute), California and supported by DARPA and NSF, is the *de facto* standard for network simulation and its behaviour is highly trusted within the networking research community [42]. It is also a discrete event simulator targeted at networking research organized according to the OSI model. It provides support for the simulation of Transmission Control Protocol (TCP), routing and multicast routing protocols over wired and wireless networks. NS-2 simulates a layered wired or wireless network from the physical layer all the way to high level applications. Wireless networking support has been introduced through a number of extensions, for example the Monarch CMU projects [44] developed protocols for wireless and mobile hosts, one of them being the implementation of IEEE 802.11 layers (WiFi). It includes all commonly used IP protocols along with a simulation visualisation tool called network animator (nam).

NS-2 is written using C++ and OTcl and hence is an object-oriented simulator. OTcl is an interpreted language used as a "front-end" while C++ is a compiled language and used as a back-end language. The reason for the use of two different programming languages is that OTcl is appropriate for programs and configurations needing rapid changes while C++ is appropriate for programs that need faster execution. In other words, C++ is fast to run but slow to change whereas OTcl is fast to change but slow to run. NS-2 contains a C++ class hierarchy (compiled hierarchy) with a similar class hierarchy in the OTcl interpreter. There is a one-to-one relationship between a class in the compiled hierarchy with the same class in the interpreted hierarchy.

NS-2 not only provides the most commonly used IP protocol, but also allows users to implement their own protocols; hence it is highly extensible. It also supports the four ad hoc routing protocols discussed above, including DSR. The downloaded source code of NS-2 can be compiled on multiple platforms, for instance UNIX, or on Windows using Cygwin[3] [45].

2.9.4 Comparison

All of the above mentioned simulators provide the components required for MANETs, discussed above. Now we have to consider other criteria to narrow the selection. There are many criteria based on which a suitable simulator can be chosen; we will start by considering accuracy as an important factor. However, there is no clear conclusion in the literature about which one is the most accurate. Cavin *et al.* [46] conducted experiments for comparing the accuracy of the simulators and the authors concluded that the results are barely comparable because the results are different quantitatively (not the same absolute value) as well as qualitatively (not the same general behaviour). Moreover, there is no standalone simulator that can fit all the necessities of the wireless application developers. It would be more realistic to think about a hybrid approach in which the lowest layers (MAC and physical) and the mobility model are simulated while the upper layers (transport to application layers) are executed on dedicated hosts, such as a cluster of machines.

[3] Cygwin is developed by the Open Source community in order to provide a Unix-like environment running in Microsoft Windows Systems.

Since there is no definite conclusion regarding the accuracy of the above mentioned network simulators, we have to opt for one as our simulation environment based on our own ease and attraction. In Table 2-2, we summarise the comparison of simulators based on a variety of metrics.

Table 2-2: Comparison of Simulators

Simulators	**Free**	**Open Source**	**Programming Language**	**Implementation of DSR**
OPNet	No	No	C	Yes
GloMoSim	Limited	Yes	PARSEC	Yes
NS-2	Yes	Yes	C++, OTcl	Yes

From the comparison summary, we opt to use NS-2 as a network simulator for our research work for the following reasons.

- NS-2 can be easily downloaded and installed for free.
- It is open source.
- NS-2 has active research community and a large archive of questions and answers can be found on the Internet which can provide solutions to many problems; hence makes the research progress smooth.
- Source code of other protocols, such as the CONFIDANT protocol, can be easily obtained, which makes the job of comparing our scheme with other benchmark schemes easier.

2.10 Summary

This chapter provides background for our work that includes network security, mobile ad hoc networks and their requirements, various routing techniques developed for MANETs – especially DSR protocol – and the threats that endanger the routing process. We also discussed the issue of selfishness or misbehaviour. Finally, we discussed various network simulators available in the research community that could be used to simulate MANETs, along with a discussion to justify our selection of NS-2.

Chapter 3 Cooperation Enforcement in MANETs

Cooperation enforcement schemes can be mainly categorized into three groups *i.e.* reputation-based, trust-based and credit-based schemes, as shown in Figure 3-1. Reputation and trust based schemes work more or less the same way, *i.e.* nodes are classified as trustworthy or untrustworthy based on their cooperation history in the network. Eventually, nodes having high reputation or trust (greater than some threshold) get services; whereas nodes having low reputation or trust are isolated from the network. The reputation and trust evolve locally in a distributed manner. Reputation-based schemes are further divided into two sub categories, *i.e.* active and passive acknowledgement schemes which will be discussed later in this chapter. In credit-based schemes, on the other hand, payments of usually a virtual currency are made for cooperation in the network; however for secure payments nodes are employed with a tamper proof hardware. A central virtual bank is used for payments management. We discuss these schemes in the following sections.

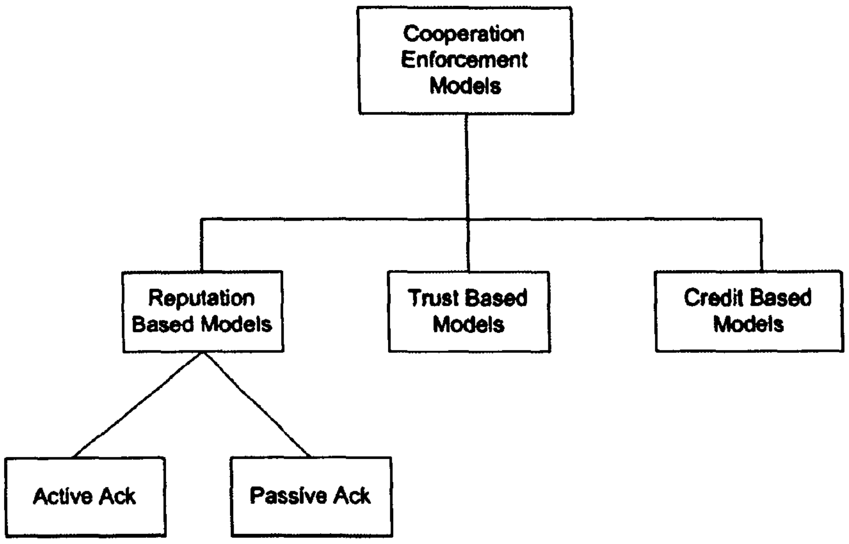

Figure 3-1: Cooperation Enforcement Models

3.1 Credit-based Schemes

In credit-based schemes the packet-forwarding function is treated as a service which can be weighted (valuated) and charged. Nodes pay for services like packet forwarding and they also act as a buyer and/or seller of services. These models incorporate a form of virtual currency to regulate the dealings between the nodes for services. They require the existence of tamper-resistant hardware and a virtual bank. The former is used by each node for secure payments (on client side) and the latter is used on network level that offers trusted-third party services to the nodes, such as managing payments, securing transactions, etc.

The concept of a commercial transaction in a MANET was first used by Buttyan *et al.* [47] to prevent selfishness in MANETs. They proposed two models: the Packet Purse Model (PPM) and the Packet Trade Model (PTM); both of these models make use of a tamper proof hardware security module. These models make use of a virtual currency called nuglets. In PPM, the source node adds nuglets to the packet and each intermediate node will take an amount for forwarding the packet until the packet reaches the destination node. The source node is responsible for loading a sufficient quantity of nuglets onto the packet; otherwise a packet having insufficient nuglets will be discarded on intermediate nodes. The problem with this model is that a node can take more nuglets than the amount they are supposed to take. The use of this model is limited to source routing protocols, such as DSR. It is due to the fact that in the PPM model source nodes are responsible for adding the correct amount of nuglets in the packet; and in source routing; only source nodes select the paths for communication and they know the number of nodes in the selected path. Another problem is in the Route Maintenance phase, when the path is considered broken in the middle, half node will take money for nothing useful. In the PTM, each intermediate node purchases packets from the previous node and sells it to the next node at a higher price. Eventually, the destination node will pay probably the highest price to the second last node along the path. However an attacker can easily launch a DoS attack in this scheme, because the source node does not pay for the service.

Wang, *et al.* developed SPM: A Security Policy Based on Micropayment in Ad hoc Networks [48], which combines the features of credit-based schemes with the subjective observation based monitoring of reputation-based schemes. Nodes are characterized in the network as ***consumers***, ***merchants***, and ***brokers***. Consumers are nodes that send their data/control packets to other nodes (merchants) to forward. Merchant nodes forward data/control packets from consumer nodes and take a virtual currency called ***scrip*** for providing services to them. Brokers on the other hand are the neighbours in a transaction that issues scrip to the consumers and merchants based on their reputation. Merchants sell their services to consumers and consumers acquire scrip from brokers (neighbours) to buy services from merchants.

3.2 Trust-based Schemes

Reputation and trust are sometimes used interchangeably; however some authors [49] consider them different entities. Reputation and trust are tightly related to each other: reputation enables trust. Reputation reflects a collective opinion that leads to trust or distrust, whereas trust typically exists at a personal level. That's why more weight is given to the personal or direct opinion than that of indirect or secondhand opinion. In other words, trust is one's subjective or personal experience of an entity about another entity. When these experiences are combined with other entities in the community, its collective reflection in the community is reputation. In this section we will overview a couple of trust-based schemes for MANETs.

Pirzada *et al.* [50] describe an approach to building distributed trust relationships between nodes, assuming DSR as the routing protocol used in an ad hoc network. It is assumed that nodes in the network passively monitor events; we will discuss these techniques in the next section. Events are the activities of nodes in the network, for instance forwarding, receiving, and overhearing of data or control packets. Independent trust agents are used that reside on each node in the network. Each agent firstly drives trust for other agents by monitoring other nodes in promiscuous mode for different events, such as data or control packet forwarding events. Information gathered from these events may be classified into one or more trust categories which indicate that a node is trusted for the particular category of service. For example, an agent may be trusted for data forwarding, but not for routing security. Secondly, each agent quantifies trust in a continuous range by collecting information from the neighbouring nodes, filtering it, and assigning weights to each event, before finally computing trust levels for each node.

Based on the subjective trust model developed by Xiaoqi *et al.* [51], Vasantha *et al.* [52] proposed a Subjective Trust Model, called Trust-based DSR (TBDSR). Here each node is capable of having an opinion about every other node in the neighbourhood. The opinion is represented by a triple value containing belief, disbelief and uncertainty (*b, d, u*), where all of these values are continuous ranging from 0 to 1. Based on the opinion of successful and failed communication experiences, each node maintains a list called a Trust Table. A node must forward others' packets to achieve a good opinion in the neighbourhood. Eventually nodes having a bad opinion in the community may be isolated from the network.

3.3 Reputation-based Schemes

Reputation-based models consider the past history of interactions and based on that history they enable nodes to identify cooperative (trusted) or uncooperative (untrusted) nodes in the network. Nodes build up these histories or subjective reputation from the direct interaction experiences that are made visible to the new interacting nodes in the form of secondhand reputation information. However, nodes can use both direct and indirect experience to better evaluate the interacting nodes. Visible past histories are of significant importance in building up reputation in the network. According to Friedman *et al.* [53], history is very beneficial in many aspects. First, a history may show the information about the ability of an entity. Second, history deters moral hazards in the present: each entity will perform to the best of its ability because current actions will become history in the future. Finally, since histories reveal information about a node's abilities, nodes with higher abilities are distinguished from the nodes having lower abilities. Reputation-based systems usually enable nodes to collect, maintain, and disseminate reputation information histories in the network. In a MANET environment, reputation information is gathered either through passive acknowledgments (promiscuous mode listening) or by active two-hop acknowledgments. Eventually nodes having high reputation can access services whereas nodes having low reputation are isolated from the network.

A core component of the reputation-based models is the monitoring or observation of one hop neighbours. Due to the fact that a node can trust nobody but itself, it gives more weight to direct observations, which are called firsthand information. The accuracy and efficiency of misbehaviour detection is strongly dependent on the monitoring component. However, due to the complex and unpredictable nature of a mobile ad hoc network, it is very difficult to have a perfect monitoring system at low cost (where cost may be measured as energy or some other metric). In this section we will discuss the two main sub categories of reputation-based schemes, *i.e.* active and passive based acknowledgments schemes and will also discuss their pros and cons along with other related issues that will need to be addressed in the future research.

3.3.1 Passive Acknowledgement Based Schemes

In these schemes, monitoring is done by making use of passive acknowledgements and the networking hardware will be put in promiscuous mode. The schemes may not work efficiently in the presence of ambiguous collisions, partial dropping, or unidirectional links [22] which will be discussed in this section.

Watchdog and Pathrater

Two techniques were proposed by Marti *et al.* [54]: Watchdog and Pathrater which run on the Dynamic Source Routing (DSR) protocol. Watchdog is a process running on each node that maintains a reputation table to record the reputation of one hop neighbours. Every time a source node *S* sends a packet to the destination node *D* via intermediate nodes, *S* holds a copy of the packet in memory until it overhears (in promiscuous mode) the same packet forwarded by the intermediate node before time *T* expires. Node *S* increases the reputation of the next node when it is confirmed that it has forwarded its packet, and decreases it otherwise after a time-out period. If the neighbouring node drops packets exceeding a given threshold the node is deemed to be misbehaving. Pathrater is used to evaluate paths in order to avoid the path having misbehaving nodes. Each node maintains a list of ratings for every neighbour node in the range from 0 to 1. Node ratings are initialized to a neutral value such as 0.5 and then incremented or decremented depending upon its behaviour. Using these ratings Pathrater evaluates paths and selects the one having the highest rating. This technique has no punishment for misbehaving nodes, a drawback which is addressed by the CONFIDANT scheme.

CONFIDANT

Buchegger and LeBoudec [25] proposed an extension to the DSR protocol called CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks), which is similar to Watchdog and Pathrater. It has four main components: the Monitor, the Trust Manager, the Reputation System, and the Path Manager. The Monitor applies similar techniques to those of the Watchdog process as it promiscuously listens to neighbours but also observes routing protocol behaviour. The Trust Manager is responsible for sending an ALARM to all its friend nodes when misbehaviour is detected, and other trust managers receiving ALARM messages determine the trustworthiness of the message by scrutinizing the trust level of the sender. Based on this information the Reputation System maintains a local rating list and a blacklist, and further exchanges these lists with friends. Finally the Path Manager, which applies similar techniques to those of Pathrater, evaluates paths according to the reputation of nodes along the path and discards paths that include misbehaving nodes. Since this protocol allows nodes in the network to send ALARM messages to each other, it could give more opportunities for attackers to send false alarm messages that a node is misbehaving while actually this might not be the case, a process referred to as rumour spreading [55]. The authors enhanced their scheme in later work [26] by using a Bayesian model that classifies and rules out liars. We discuss and use this latest version of CONFIDANT in order to analyse the effect of direct interactions on reputation-based scheme in MANETs, in Chapter 5. We will also compare our scheme with this in Chapter 6.

CORE

As with the previous schemes, CORE [38] also relies on the DSR routing protocol. It monitors neighbours via the watchdog mechanism as well. CORE combines three types of reputation information: subjective (direct) reputation, indirect reputation and functional reputation. Functional reputation depends on certain functions which are given a weight based on their importance, for example control packets are considered less important than data packets, and hence more weight is given to data forwarding functions. These reputations are combined for a node to have an aggregate reputation in the network. Nodes having reputation below a given threshold are isolated from the network. An isolated node can, however, rejoin the MANET if it successfully increases its reputation by cooperating well for a period of time. To protect nodes suffering temporarily from bad environmental conditions, more weight is given to past behaviour. However this also provides an opportunity for an attacker to misbehave after building up a good reputation. Unlike CONFIDANT, only positive recommendations are allowed to be communicated in the network.

3.3.2 Problems with Passive Acknowledgment

Most reputation-based schemes make use of promiscuous listening or passive acknowledgments to observe their neighbours for packet forwarding activities. Apart from its advantages, such as the fact that it needs no specialized hardware ensuring low cost, promiscuous listening has several disadvantages which are caused by the peculiarities of mobile ad hoc networks. Marti *et al.* [54] identify the following weaknesses of the Watchdog mechanism in the presence of which it might not detect a selfish or misbehaving node.

Ambiguous collision: as shown in the scenario demonstrated in Figure 3-2, an ambiguous collision occurs at *A* while it is listening for *B* to forward a packet on. Node *A* does not know whether the collision is caused by its neighbouring nodes or by node *B*.

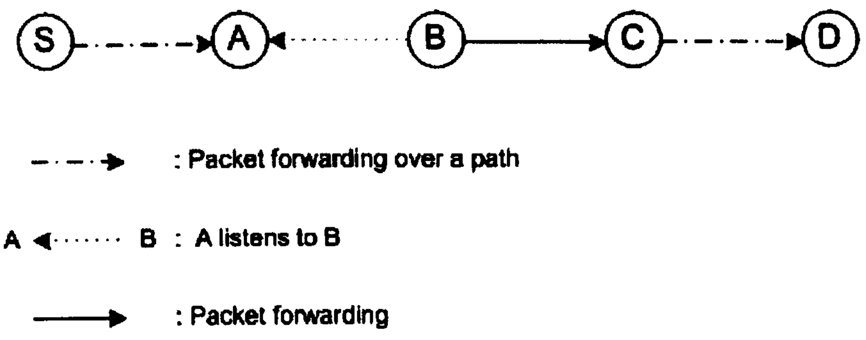

Figure 3-2: Passive Acknowledgment

Receiver collision: due to the receiver collision problem node *A* can tell whether *B* sends the packet to *C*, but it cannot tell whether *C* received it. Node *C* might not receive the packet because of a collision.

Limited transmission power: a node can limit its transmission power such that a signal is sufficiently strong to reach the previous node while weak enough not to reach the true recipient.

Collusion: more than one misbehaving node can collude to disrupt the Watchdog mechanism. For example, *B* forwards a packet to *C* but *B* does not report to *A* when *C* drops the packet.

Partial dropping: when a node drops packets at a rate lower than the configured misbehaving threshold.

3.3.3 Active Acknowledgment Based Schemes

These schemes do not suffer from the problems mentioned above, but instead of listening to passive acknowledgements, they explicitly send acknowledgements to inform the source node of packet receipt. For this, they incur extra overhead; however some schemes use Piggybacking to decrease the overhead incurred.

2ACK

Kejun *et al.* [56] propose a scheme that focuses on the detection of misbehaving links instead of misbehaving nodes and which may be used as an add-on to the existing source routing protocols, such as DSR. The scheme makes use of a special acknowledgment packet which has been assigned a fixed route of two hops or three nodes in the opposite direction of the actual data traffic flow. Three consecutive nodes (triplets) N_1, N_2, and N_3 are assumed to lie along the path from source to destination. When N_1 forwards a packet to N_3 via N_2, N_1 will not be sure whether N_3 received the packet due to the misbehaviour or ambiguous collisions in the path. In order to confirm the packet reception N_3 will send an ACK packet to N_1 via N_2, called a 2ACK. Among the triplet, N_1 is the observer (or ACK receiver) of the link N_2 => N_3. This triplet formation is carried out along the whole path.

Every outgoing packet N_1 will store the ID of the packet for time t in a list it maintains. When an ACK is received for a packet and is matched to an ID in the list before time t expires, the entry in the list is discarded and a special counter C_{pkt} is incremented and C_{mis} is incremented otherwise. After a time period T_{obs} the ratio of C_{mis} and C_{pkt} is compared with a preset threshold R_{mis}, if the ratio is greater than the threshold all nodes are reported regarding the misbehaving link N_2 => N_3 by sending a RERR message. Each node receiving or overhearing such a RERR message deems N_2 => N_3 to be a misbehaving link. These links are then avoided in the future. Nodes such as N_2 should not alter the 2ACK packet passing through them; a one-way hash chain mechanism is used as an authentication scheme to avoid such tampering.

MARS

Zhao *et al.* [57] propose MARS (MultipAth Routing Single path transmission), which provides protection against cooperative misbehaving nodes. The scheme combines multipath routing and single path data transmission with an end-to-end feedback mechanism. Before a source node starts communication with a destination node, it first selects two node-disjoint paths (paths having no node in common) from its path list; one is used for data transmission and the other is for transmission information exchange. Two new control packets INF and NTF are introduced in the scheme; however, it is not specified what these abbreviation stand for. When the first data packet is sent on one path then a special control packet INF is sent through the other path to inform the destination about the data rate, packet size, and other path related information. The destination node then detects misbehaviour by analyzing the actual data received and the data sent by the source node. If the data rate falls below a certain threshold or the packets are tampered with then it notifies the source node about the misbehaviour along the path using a special control packet called an NTF. Authentication is used on source and destination nodes only. Upon receiving an NTF packet the source node will remove the paths from the list and select two other paths; otherwise it will start a new route discovery process to discover new paths.

MARS reduces overhead compared to 2ACK; however, because it only detects misbehaving *paths* it can reduce the chance of utilizing good nodes that happen to fall along a path with another node that misbehaves. There is no punishment strategy, so if the number of selfish nodes increases then there would eventually be no path available for data transmission.

LeakDetector

Graffi *et al.* [58] propose a scheme which they refer to as LeakDetector. It works using a proactive routing protocol and uses both the Watchdog mechanism and active acknowledgments for detecting misbehaviour of a single node or multiple colluding nodes. Each source node maintains a traffic counter for each path indicating the amount of traffic transmitted to that path. The periodic proactive routing messages will contain two extra (new) fields *i.e.* T_{total} and T_i. T_{total} is the field that specifies the total traffic for this route and the T_i field is used by the visited node *i* to specify the fraction of traffic that has passed through the node in relation to the total traffic sent by the source node. Along the path, data will be passed through nodes. Let's say N_1 passes data to N_2. Then N_2 will first append its information to the visited node list field in the packet and then report the amount of data received from N_1 in the T_{N1} field in relation to the preset field T_{total}.

The destination node creates a virtual graph based on the information received (T_{total} and T_i) from the nodes constructing the path. Vertices represent nodes and the weighted edges represent the amount of traffic flowing between two nodes. The graph obviously provides a top level view of the whole traffic flow in the path. When the destination node determines a node with significant in/out traffic flow variation, it is reported to the source node via a route reply message on a disjoint path. The source node can then take further action by updating the blacklist table or using a reputation technique to punish the malicious node. Moreover traffic counters are refreshed periodically to detect a node that was previously cooperating and then switched over to malicious behaviour.

The authors evaluate the scheme using seven stationary nodes. Since it is proposed for use in MANETs, a larger number of nodes in a mobile environment would be needed for a more complete evaluation.

3.4 Discussion

Credit-based schemes have some issues that make them impractical for use in MANETs. Firstly, they are not scalable due to the central virtual bank. Secondly, these models need some form of tamper proof hardware on each node. Reputation or trust-based models on the other hand do not need any centralized entity, such as a virtual bank, or any tamper proof hardware. As a consequence they can be implemented in a distributed manner to increase scalability, making them much more suitable for use in MANETs.

Table 3-1 summarizes the important features and limitations of the schemes we discussed in Section 3.3 of this chapter. Generally speaking, passive acknowledgment techniques are more promising than active acknowledgment techniques because they do not cause any extra communication or memory overhead. Active acknowledgment provides reliability at the cost of extra memory and communication overhead but in environments where there is a high collision rate, the active acknowledgements are as prone to errors as promiscuous listening techniques. Schemes such as MARS [57] that make use of node-disjoint multipath routing maintain a large cache for storing these paths. Since in mobile environments path reconstruction rate can be high, this can cause substantial overhead and delay due to the frequent path discovery messages in the network. Furthermore, acknowledgments no matter if explicit, may still be lost due to high mobility or collisions resulting in further overhead in the form of retransmissions. On the other hand promiscuous listening techniques can cope with high mobility or collisions by only relaxing the misbehaviour threshold.

Table 3-1: Comparison of Reputation-based Schemes

Scheme	Observation	Detection	Punishment Strategy	Features	Limitations
Watchdog	Passive	Single Node	No	Path ranking based on selfish nodes in them; hence selfish nodes are bypassed.	Selfish nodes are not isolated from the network.
CONFIDANT	Passive	Single Node	Yes	Selfish nodes are isolated and trusted recommendations are taken into account.	Vulnerable to rumour spreading.
CORE	Passive	Single Node	Yes	Second chance is given for nodes in bad locations and no negative ratings are communicated.	More weight is given to past behaviour and hence recent misbehaviour will be ignored.
2ACK	Active	Single Node	Yes	Use active acknowledgments for two hops to remedy the problems in promiscuous listening.	Substantial memory and message overhead.
MARS	Active	Collusion	No	Acknowledgment from destination to source using node-disjoint multiple path.	Large memory overhead caused by storing multiple node-disjoint paths and sometimes it does not utilize good nodes because it deletes paths containing selfish nodes.
LeakDetector	Both	Collusion	Yes	Destination node creates a virtual graph of the traffic it receives from a source, and hence it helps to identify traffic leaks.	Large memory overhead and evaluation is based on limited nodes with no mobility.

Buchegger *et al.* [59] conducted a testbed to establish how efficient promiscuous listening is and they discovered some very interesting results. First, in high traffic loads they did not experience a single collision; hence there was a very low chance of ambiguous and receiver collisions; however these can still be compensated by adjusting the threshold. Second, they found out that it is not easy for a misbehaving node to drop a packet due to malicious power adjustment (limited transmission power) because it is difficult to achieve the power range adaptation using current off-the-shelf hardware. Third, in some situations passive acknowledgments perform even better than active acknowledgments.

3.5 Limitations of Current Work

The schemes we have discussed in previous section have some limitations which need to be seriously considered in future research. Hoffman *et al.* [60] classified attacks against reputation systems that have been developed mostly for peer-to-peer systems; more or less the same attacks have been pointed out by various authors including Mandalas *et al.* [18], Mitchell *et al.* [61], Carrara *et al.* [49], and Agrawal *et al.* [17]. As we discussed in Chapter 1, due to the different characteristics of MANETs, reputation systems developed for MANET environments are different from those that have been developed for P2P environments. Our aim here is to classify the attacks against the reputation systems that have been proposed for mobile ad hoc networks only. Below, we discuss some of the attack mechanisms that exploit the component(s) of reputation systems in MANET environments.

3.5.1 Self-Promotion

In this kind of attack, an attacker tries to falsely increase its reputation in the network. The possibility for such attacks is usually found in reputation systems that consider positive feedback such as CORE and CONFIDANT.

Self-promotion attacks can be performed by a node in collusion with other nodes or through a single node using its own virtual identities, called Sybil identities. Sybil attack [20] occurs when a malicious node generates and controls a large number of logical identities on a single physical device. This gives an illusion to the network of each logical identity being a different legitimate node. The individual form of self-promotion attack can be found in the scenarios where a node broadcasts spoofed secondhand reputation information with fabricated augmented reputation about itself. Systems having strong authentication mechanisms can counteract this attack; however a node colluding with other nodes or with its own Sybil identities can still launch the attack. Collusion[4] among different distinct adversaries has been addressed by a number of researchers [58, 62-65]. Collusion in the form of Sybil attacks can be avoided or mitigated by detecting Sybil attacks in the network, these approaches will be discussed in the next section. We will discuss our proposed approach in Chapters 6 and 7.

3.5.2 Slandering

In this type of attack, which is more or less the reverse of self-promotion, one or more nodes or identity falsely generates negative feedback about other nodes or identities; this is also called badmouthing. An individual node can publish secondhand reputation information having falsely accused good nodes. Reputation systems that consider negative feedback mechanisms are vulnerable to this attack. For example, to cope with this attack CORE only allows positive feedback whereas CONFIDANT uses a trust mechanism and deviation tests to counteract the issue. However, in CONFIDANT there are no restrictions on the frequency of secondhand information exchange, a node can easily be *brainwashed* by frequently having false negative ratings sent to it while passing the deviation tests at neighbouring nodes. Slandering performed by an individual may have less destructive effects than that of an attack committed by a coalition of nodes or identities. However, these kinds of attacks can be mitigated using authentication mechanisms.

In case of a coalition of nodes or identities, a single node having other Sybil identities or a coalition of other distinct nodes badmouth about other nodes in the network. Some schemes have tried to mitigate the effects of it. For example LARS [66] and OCEAN [67] do not allow nodes to exchange their reputation information with neighbours whereas CONFIDANT assigns low weights to secondhand reputation ratings to reduce the effect of slandering.

[4] It is out of the scope of this thesis to discuss collusion attacks any futher.

3.5.3 Whitewashing

In a whitewashing attack, an attacker abuses the system for short-term benefits by allowing its reputation to degrade and then escapes the consequences of abusing the system by exploiting some system vulnerability to repair its reputation [60]. In MANET environments (as with other online systems), the easiest way for the attackers to repair their reputation is to re-enter the system with a new identity and get a fresh neutral reputation. The whitewasher further takes advantage of the availability of free pseudonyms to whitewash as many times as it likes. These zero cost identities make it harder to maintain reciprocity in the network [68]; in other words, harder to ensure that nodes face the consequences of their actions. It is due to the fact that accountability is based on identity which can be easily obtained, changed or discarded in mobile ad hoc networks given that there is no centralized identity management.

The main reason for whitewashing being beneficial for an attacker is the ***neutral reputation*** that is an initial reputation allocated to each newcomer. The amount of this initial reputation, no matter how big or small, can always be manipulated by an identity changer when there are no restrictions imposed on it. In other words, this neutral reputation gives the opportunity for whitewashers to utilize network services without contributing to the network. This will further encourage them to change their identities after they have been identified by a misbehaviour detection scheme. We will discuss in Chapter 6 about how we put restrictions on the neutral reputation to discourage whitewashers.

In order to make whitewashing attacks more effective and productive, an attacker can combine it with other types of attack. For example, in a reputation-based system that takes positive and negative feedback into consideration, a whitewasher can concurrently perform a self-promotion attack to lengthen its identity lifetime. Similarly along with other misbehaviour, a whitewasher can increase its identity lifetime by slandering those nodes that produce negative feedback about it, in order to make their negative feedback less reputable; hence their feedback will be considered less trustworthy in the network and have less impact.

3.6 Sybil Attack and Whitewashing Detection

The following are some of the existing solutions developed for attackers using more than one identity. These solutions can be applied for both whitewashing and Sybil attacks, because the only difference between whitewashing and Sybil attacks is that a Sybil attacker *simultaneously* controls more than one identity whereas a whitewasher uses more than one identity but one at a time: there is no notion of *simultaneity*.

Levine *et al.* [69] surveyed countermeasures against Sybil attacks and categorized these techniques as follows.

- *Trusted Certification*: Many authors suggest trusted certification as a solution to prevent Sybil attacks. Trusted certification employs a centralized authority which is responsible for establishing a Sybil-free domain of identities. Each entity in the network is bound to a single identity certificate. Douceur [20] has shown that trusted certification is the only approach which is fully capable of preventing Sybil attacks. However, there are problems in this approach. For instance, it suffers from costly initial setup, lack of scalability and a single point of failure or attack.
- *Resource Testing*: In this approach various tasks are distributed to all network nodes or all identities to test the resources of each node to determine whether each independent node has sufficient resources to accomplish the tasks. The tests are carried out to check the computational ability, storage ability and network bandwidth of a node. In the case of a Sybil attack, an attacker will not possess sufficient resources to perform the additional tests imposed on each Sybil node. The main drawback of this approach is that an attacker can acquire enough hardware resources, such as storage, memory, and network cards to accomplish these tasks.
- *Recurring Costs and Fees*: In this approach identities are periodically re-validated. Each participating identity is further periodically charged with a fee. For example, Margolin *et al.* [70] propose the use of a recurring fee per participating identity to deter Sybil attackers and they suggest that such a recurring fee is more of a deterrent than a onetime fee. They also establish that recurring fees can incur a cost to the Sybil attacker that increases linearly with the total number of participating identities, whereas a one-time fee incurs only a constant cost. The recurring fee may not be a

monetary payment mechanism, it can also be a non-monetary payment mechanism such as CAPTCHAs [71] or charged SMS messages. CAPTCHAs are automated puzzles which are hard for a computer to solve but very simple for a human. However, fee management is generally too costly to implement and manage in MANETs.

- *Trusted Devices*: This is a one-to-one mapping of a hardware device and a network entity. In other words, one hardware device, such as network card, is bound to one network entity. However, there is no way of preventing an entity from acquiring multiple hardware devices, such as where an attacker installs two network cards.

The above schemes were basically developed for P2P and web based applications which incur a fee or micropayment on a per identity basis. However, these are not suitable for MANETs for two reasons: first, monetary payments are not suitable for MANETs; one of the various reasons for this is that MANETs are often deployed in emergency or disaster scenarios where these payment schemes are not practical. Second, in the case of a onetime entry fee or recurring (monetary) fee per identity, the management of these payments is too costly to implement in MANET environments. We summarize the following schemes that are specifically developed for ad hoc networks.

Carrara *et al.* [49] suggest low reputation rating for newcomers in order to discourage whitewashers from changing identities. Nodes are then required to increase this low reputation further. This promotes identity persistence in circumstances where there exists the ability to change identities easily. Unfortunately, the use of this mechanism will discourage new legitimate users; moreover by changing an identity this low reputation can still be manipulated.

Hubaux *et al.* [72] exploit mobility to enhance security in MANETs. Off-line certification authorities are used for the authorization of mobile nodes when joining the network. For fully self-organized security where there is no central authority, nodes establish security associations purely by mutual agreement. Users can activate a point-to-point Secure Side Channel (SSC) using infrared or wired media between their personal devices to authenticate each other and set up shared keys when they are in close proximity to each other. The author attempts to solve the problem of impersonation and Sybil attacks by binding a user's face and identity using these SSCs. However SSCs are based on the assumption that nodes are connected through wired or infrared connections. Infrared and wired connections are often not practical in a MANET environment, because of the short range and line-of-sight nature of the infrared links and the static nature of wired media (movement constraints).

3.6.1 Cryptographic based Detection

Hashmi *et al.* [73] categorized cryptographic based authentication mechanisms for MANETs into three broad categories, *i.e.* central Certification Authority (CA), distributed CA and self CA systems.

Central CA systems: these are the simplest but most expensive solutions (in terms of required infrastructure cost); they are also called trusted certification which is discussed in Section 3.6. Before network formation, each node obtains its identity credentials from a central CA, which is also called Trusted Third Party. This TTP is responsible for thoroughly checking the credentials of the applicant before issuing any identity or certificate to it. It also maintains a database for storing all issued identities and uses this database for revoking or renewing any identity. These systems are very effective for Sybil attack detection; however there are certain limitations of this approach that make it unsuitable for MANETs. First, the reachability of TTP to all nodes in MANETs may not always be guaranteed. Second, TTP is the single point of failure or attack. Third, it is costly in terms of initial setup, etc.

Distributed CA systems: in this approach, n nodes collectively perform the tasks of a CA in a MANET. Each of these n nodes uses the threshold cryptography in order to generate a share of the master private key of the CA. Eventually any k out of these n nodes will provide CA services in the network. Different authors suggest different values for the parameters k and n. Zhang *et al.* [74] suggest all nodes to take part in CA formation and then for a new node any k neighbours can provide CA services. A new node will be issued a certificate after negotiating with each of the k CA nodes. The issuance or renewal of a certificate for nodes is a resource intensive job that requires considerable communication between the new node and k CA nodes. Zhou *et al.* [75] suggest the n number of nodes that establish CA to be static. However, this approach limits the availability of CA services, *i.e.* the required k out of n nodes might not be accessible to any new nodes at all times. Generally, in these schemes, a malicious node can easily get *k* multiple identities and *k* shares of the private key of distributed CA by continually making network joining requests and can eventually build the master private key of the distributed CA. It is due to the fact that these schemes do not specify how the new nodes' credentials are checked [73].

Self CA systems: these systems are not costly on node level; however they are not a good resistant for the Sybil attacks. In these systems, nodes can generate themselves as many identities as required. The Web of Trust model [76] is one such example of these systems. In this model, each node becomes an individual CA and issues its own certificate (identity). Nodes can create trusted certificate chains by signing and exchanging certificates with other trusted nodes. The trustworthiness can only be established by direct physical contact. Each node maintains a repository of certificates in order to store certificates issued by it and received from trusted ones. When a new node cast a request for authentication, the authenticator node checks its locally maintained repository for a certificate chain to leads to the requesting node. The authenticator signs the certificate if a certificate is found. Furthermore, both nodes exchange their certificate repositories (if trusted). The drawback of this model is that nodes are free to generate and sign certificates; similarly, they can create fake trust chains involving Sybil identities and hence subverting the whole authentication process.

A thorough discussion on this topic is out of the scope of this thesis, interested readers are referred to Hashmi and Brooke [73].

3.6.2 Signal Strength based Detection

The motivation for RSSI (Received Signal Strength Indicator) to be used for Sybil node detection in wireless networks is that it is lightweight compared to public key cryptography which requires heavy computation. However, RSSI is unreliable due to its time varying nature, its dependence on transmission power and, apart from this, the irregularity of the transmission medium. Most RSSI schemes are based on the radio model that says: the power received approximately decays with the distance to the power *α*, *i.e.*

$$P_r \propto P_t/d^{\alpha} \qquad \text{Eq. 3-1}$$

where P_r is the received power at the receiver node, P_t is the transmit power at the transmitter node, and d is the distance between the transmitter and the receiver. The value of α for outdoor propagation is 2, *i.e.* free space model and for indoor is 4, *i.e.* two-ray ground reflection model [77]. If the transmitted power is known, the receiver node can deduce the distance between them and thereby use simple geometric triangulation to locate the transmitter. Here we summarize some well-known RSSI based Sybil attack detection schemes.

Zhong *et al.* [78] show that no sensor node can hide its location in an environment where it is monitored by four or more nodes. Using the ratio of RSSIs from these multiple receivers, no node can hide its location from the authority that controls these monitoring nodes. The proof given by the authors can be found in Appendix D.

Demirbas *et al.* [79] implemented Zhong‘s algorithm of localization by conducting an indoor experiment of static MICA 2 motes. The author argues that since the locations of the nodes stay the same, it's cumbersome to calculate locations. In other words, they avoid calculations of fading through distance. They prefer to record and compare the ratio of RSSI for the received messages instead. They demonstrated through experimentations that RSSI values fluctuate a lot even for fixed nodes. Nonetheless using the ratio of RSSI, the time variance can be overcome and even two collaborating detector nodes are sufficient to detect a Sybil node. However, node cooperation is very important for the scheme to be viable; nevertheless, nodes may not be trusted or may collude in a hostile environment.

Jiangtao *et al.* [80] proposed a Sybil node detection scheme for static clustered wireless sensor networks using RSSI and status information aggregated in the head nodes. They also used Zhong's algorithm [6] for localization. Furthermore to emulate a real network space situation, Jake's Channel Model [81] was established between network nodes. William jakes developed a model for Rayleigh fading (it is a fading model developed specially for urban areas) based on summing a series of sinusoid signals. Two methods were proposed to enhance detection accuracy: judging member nodes and head nodes. To judge member nodes, head nodes accumulate status information from the member nodes and use RSSI to verify the results. An alarm message is broadcast to other heads if a Sybil member node is detected. To judge the head nodes, member nodes cooperate and share information to verify the location of each head. Members raise an alarm and announce re-clustering if all of the group members detect a Sybil head. The performance degrades with network sparseness due to errors in distance computation. The scheme suffers from the same problems as Demirbas *et al.*'s [79].

Suen *et al.* [82] propose a scheme for peer/node identification and authentication by associating transmitters' location information. Each node must be equipped with GPS to establish its own position. Using the magnitude of signal strength information, which is further confirmed by at least two collaborating nodes, the distance between receiver and transmitter is determined using lateration [83]. In order to precisely locate the transmitter, directional antennas are used with the collaboration of at least one trusted node (using angulation [83]) to capture the signal direction. The authors indicate that the factors affecting variance in location are speed and density. Location accuracy is decreased with an increase of speed and network sparseness. This scheme is lightweight but it incorporates directional antennas to determine the signal direction and the process needs trusted node collaboration.

Xiao *et al.* [84] propose a localized and distributed scheme to detect Sybil attacks in VANETs. The scheme takes advantage of the VANET traffic pattern and roadside base stations. The detection process uses statistical analysis of signal strength distribution. Signal strength distribution will be observed over a period of time for a suspect vehicle. Vehicles are categorized as claimer, witness, and verifier. Each vehicle will play all these roles, but on different occasions for different purposes. Nodes will periodically broadcast and receive beacon messages to/from their neighbours. Claimer nodes periodically broadcast beacon messages claiming their identities and positions, such as GPS position. In order to verify the position of the claimer, the verifier will collect the claimer's position plus its RSSI from the witnesses (neighbours) and compute locally the estimated position of the claimer. To get optimal estimated position an MMSE (Minimum Mean Square Error) calculation is performed on the collected signal strength information. Null hypothesis is used in order to confirm the detection. If the difference between the estimated and claimed positions of a claimer is greater than the significance level, then the claimer vehicle will be deemed as a Sybil node or vehicle. Once a Sybil node is detected the Sybil Classification algorithm is performed to check for other Sybil IDs generated by the same attacker.

In order to reduce the chance for Sybil nodes to take part in the witness group, witnesses are selected from the opposite traffic flow, which can be achieved by taking advantage of the traffic pattern and roadside base stations. Secondly, as the scheme uses signal strength distribution it is difficult for a malicious node to change its signal strength distribution. The scheme accuracy increases as the number of witnesses and observation period increases. Although results show the system to work effectively, it is nonetheless limited to VANET environments by design and would be difficult to adapt for more general MANET environments.

3.7 Summary

We have presented a detailed overview of the major schemes developed for enforcing cooperation in MANETs these are reputation-based, trust-based and credit-based schemes. Since reputation-based systems do not require any tamper proof hardware for nodes or a centralized management entity, such as virtual bank as used in credit-based schemes, they are deemed suitable for MANETs. Therefore our main focus has been on reputation-based schemes. We highlighted the potential limitations of these schemes, for example, a node can bypass the detection by changing identity (whitewashing attack), or a node can disrupt the detection accuracy by self-promoting itself or slandering about others by using Sybil attacks. We also surveyed the solutions for whitewashing and Sybil attacks from the literature, such as cryptography based and signal strength based detection; however cryptography based mechanisms usually need a trusted third party and signal strength based detection usually needs directional antennae or GPS systems for localizations. In chapters 6 and 7 we will present a layered security approach for the efficient tackling of whitewashing and Sybil attacks without using any trusted third party or any extra hardware, such as directional antennae or GPS systems. In the next chapter we will highlight another issue of detection bypassing which has not been addressed by the research community before, *i.e.* when a node interacts directly with its sources or destinations.

Chapter 4 A General Reputation System

4.1 Background

Currently, reputation systems are being widely used in many different fields of IT, such as on websites *e.g.* eBay and Amazon. In ad hoc networks, reputation mechanisms have been proposed to address and tackle the issue of selfishness in the network. The main difference between these two types of reputation-based systems is that the reputation systems used for websites are operated manually (*i.e.* involving human input); whereas in MANETs, the reputation systems are operated autonomously. One of the main advantages of reputation systems is that it works in a distributed manner which is why they are well adapted to an ad hoc network environment. In ad hoc networks, these mechanisms discourage and punish nodes that do not cooperate in the packet forwarding process. Ultimately, nodes having poor reputation are isolated from the network; whereas legitimate nodes (that cooperate) save battery power by not serving detected selfish nodes and enjoy higher throughput in the network as well.

Resnick and Zeckhauser [85] proposed the following three main goals to be accomplished by a reputation system.

- Provide information that helps distinguish between trustworthy and untrustworthy nodes or principals.
- Encourage nodes to operate in a trustworthy manner.

- Discourage untrustworthy nodes, preventing them from services to which they do not contribute.

Our aim in this chapter is to develop a reputation-based system for MANETs that captures the essence of the existing systems. We will evaluate that scheme through simulation in order to demonstrate that our scheme fulfils the above mentioned three goals. We will use that reputation-based scheme as a background for our work in the subsequent chapters (*i.e.* Chapter 5, 6 and 7). In Chapter 5, we use CONFIDANT for our investigation of direct interactions which is closely related to our reputation-based scheme. In the text that follows we will discuss the design of our general reputation-based system for MANETs, its various components and how they interact with each other in order to detect and isolate selfish nodes in the network. Finally, we show through simulation that the designed reputation-based system reduces the selfish nodes' throughput and the selfish packet drop rate as well while maintaining relatively high throughput for good nodes.

Throughout the operation of a network, a node can play two possible roles: the requestor and the provider role. The requestor refers to a node asking for the execution of a function f whereas the provider refers to a node supposed to contribute or to provide the execution of function f. By function f we generally mean a packet forwarding service which the provider offers to the requestor. The main objective of the requestor is to identify the trustworthy or cooperative nodes; whereas the objective of the provider is to discourage untrustworthy or selfish nodes by dropping their packets. In the context of (the initialisation of) a reputation system, we describe how reputation builds up with the help of the requestor and the provider model as follows.

The requestor: the requestor issues a request for function f (packet forwarding) and monitors its execution by using the promiscuous listening facility of the network interface card. The requestor confirms the result of the execution of function f and updates the reputation rating of the provider according to the acknowledged behaviour and eventually will periodically exchange this information with its neighbours.

The provider: upon receipt of a request for the execution of function f, the provider accepts or rejects the request based on the reputation rating associated with the requestor. The provider simply drops packets originated by a selfish node without explicitly broadcasting any message. This denial of execution of function *f* or dropping of packets by the provider is related to the node isolation mechanism. In the following sections we will discuss in detail the individual components, depicted in Figure 4-1, and their interaction among one another in the requester-provider context.

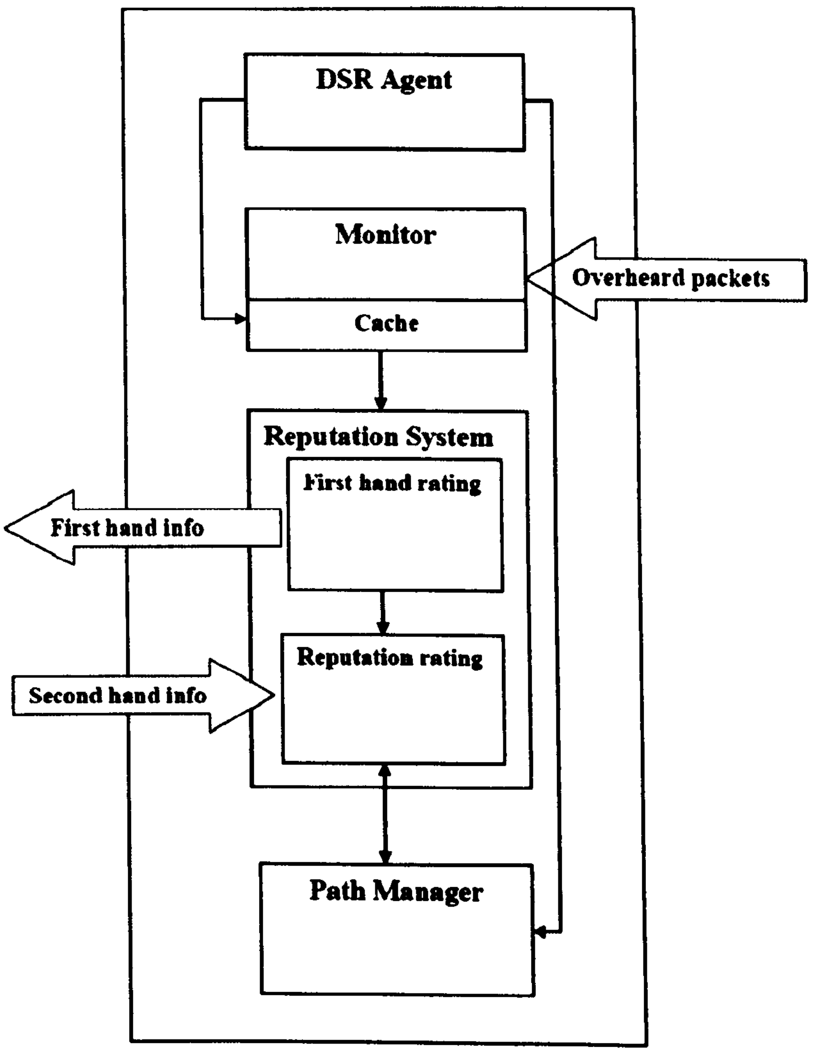

Figure 4-1: Major components of reputation-based system

4.2 The DSR Agent

DSR is the underlying routing protocol, called Dynamic Source Routing protocol, discussed in Section 2.5 that we use for our reputation system. The intention of this work is actually to fortify the DSR routing protocol against selfish nodes. The main job of the DSR protocol is to discover routes, and to send and receive packets on the routing level. It also taps (or overhears) packets in promiscuous mode. The DSR agent is the main component in Network Simulator 2 (NS-2.30) responsible for most of the routing related activities in the network; hence we refer to it as the DSR Agent instead of DSR. In appendix B, we will briefly demonstrate the DSR protocol structure in NS-2.

4.3 The Monitor

The Monitor is the main component of the reputation-based system which is used to detect selfish behaviour of nodes by confirming the execution of function *f*. In Section 3.2 we discussed the popular monitoring mechanism called watchdog by Marti *et al.* [54] and we also discussed its weaknesses. In spite of its drawbacks, it is still a promising solution for use as a monitoring tool; we elaborated our arguments about this in Section 3.3. Watchdog relies on the promiscuous mode operation of wireless NICs (network interface cards) based on the 802.11 standards. Promiscuous mode means that if a node *X* is within the range of node *Y*, it can overhear all communications from node *Y* even if node *X* is not directly involved in those communications.

Every node maintains a cache used to temporarily store copies of sent or forwarded packets. Whenever a node requires sending a data packet, no matter if as a source of data traffic or as a relay node forwarding a data packet for another node, it registers or saves a copy of the packet in its cache. An overheard packet from the next hop represents a proof of execution of the packet forwarding function *f*, if it is confirmed that the packet overheard is the one that is buffered in the cache. In order to confirm this, we compare the UID (unique identification) field of the packet in order to uniquely identify it. If the UID field of the packet is matched, the behaviour (which is *forwarding* in this case) is transferred to the reputation system and the packet entry is freed from the cache. On the other hand, if the watchdog timer expires and the corresponding *expected* packet has not been overheard, it is deemed that the next node has dropped the packet. The behaviour of the next hop node is passed to the reputation system (which is *not-forwarding* in this case) and the packet entry is freed from the cache, this whole process is shown in Figure 4-2. Based on the observation the behaviour can be written as the following.

$$Behaviour = \begin{cases} a, & \text{if the overheard packet} \neq \text{observed packet or } W_{DT} \text{ expires,} \\ b, & \text{if the overheard packet} = \text{expected packet.} \end{cases}$$

Here W_{DT} is the watchdog timer, Behaviour denotes the event observed through direct experience, with $a = 1$ if misbehaviour has been observed and $b = 1$ otherwise.

One important point to note here is that in the watchdog mechanism, the last hop before the destination node along the path will not perform all of the above formalities (for example activating the watchdog process) because the destination node is not expected to forward the packet any further.

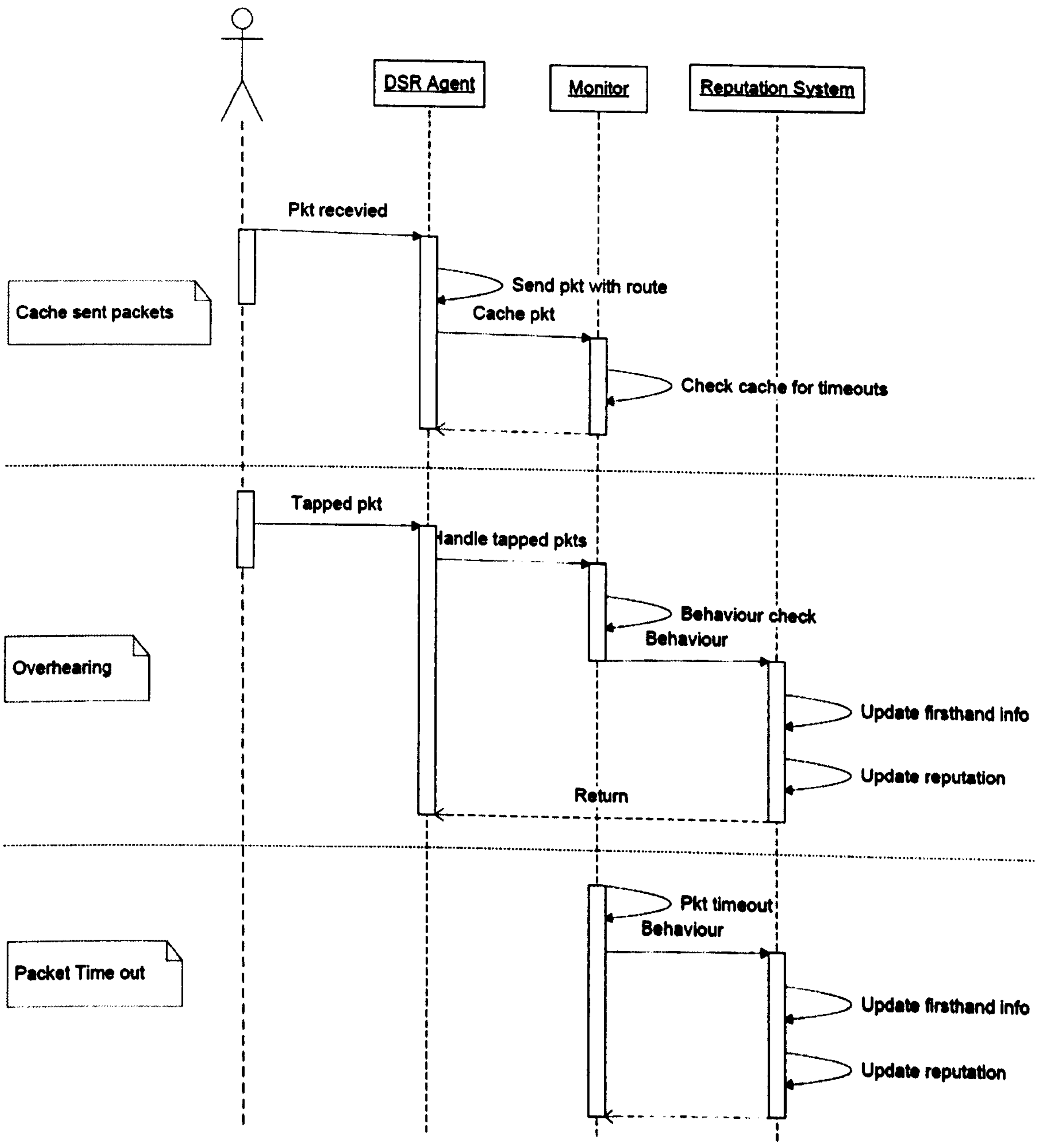

Figure 4-2: Monitoring nodes' behaviour

4.4 The Reputation System

The reputation system module also plays an important role: it is responsible for the detection of misbehaving nodes. It interacts with the monitor, the path manager and with the one-hop neighbours, as shown in Figure 4-1. Each node maintains two tables: a firsthand information table and a reputation rating table. Any direct interaction experiences regarding other nodes are captured through the monitor module and are stored in the firsthand information table. The firsthand information table contains the address of a node and its firsthand reputation rating. In order to keep the reputation secret, each node will share only its firsthand reputation information with its neighbours. The reputation table maintains the overall reputation of nodes in which reputation is computed from both firsthand and secondhand ratings. We assume neighbours to be trustworthy[5] for exchanging their direct experiences in the form of secondhand information. We adopt Buchegger *et al.* [55] model for reputation formulation and detection, which is as follows.

Handling firsthand information: We define two variables α and β, the former represents selfish (packet dropping) behaviour and the latter represents good (packet forwarding) behaviour these variables are increased or decreased based on the behaviour which is the result of the monitoring process. The purpose of the Eq. 4-1 is to provide a reputation system that incorporates firsthand, secondhand and fading updates. Based on the observed behaviour the variables α and β are updated accordingly as follows.

$$\alpha = \zeta\alpha + \omega a$$

$$\beta = \zeta\beta + \omega b \qquad \text{Eq. 4-1}$$

Here ζ is the fading factor that fades reputations after a fading timeout in order to assign higher weight to the recent activities, its value falls in [0, 1], and ω is the weight assigned to the secondhand information, its value should be less than 1 which falls in [0, 1]. Reputations *i.e.* α and β in Eq. 4-1 will be updated by three processes as depicted in Table 4-1; whereas

[5] In the thesis, we will not examine issues related to rumour spreading, *etc.*

the initial reputation rating of a node is $\alpha = \beta = 1$. Each process will provide different values for a, b, ω and ζ. These updates are explained below.

Firsthand update: in this process a and b represent a single direct event observed (1 indicates the confirmation of the observed behaviour, *i.e.* forward or drop of a packet); these values are simply added to the overall reputation rating, as shown in Eq. 4-1.

Secondhand update: each node will share its direct experiences, *i.e.* α and β, of nodes in the network. For example, after direct experience with node j, node i will share α and β of node j with other nodes, let's say node k. After receiving this information shared by node i about node j, node k will treat α and β as a and b in Eq. 4-1 and will apply the secondhand weight as well, as shown in Table 4-1.

Fading update: reputations are continually faded in order to motivate nodes for cooperation or to reduce a chance for a node that uses its high reputation for malicious activities. As shown in Table 4-1, the secondhand information will be ignored when reputations are faded.

Table 4-1: Reputation update processes

S. No.	Process Description	a	b	ζ	ω
1	Firsthand Update	0/1	0/1	1	1
2	Secondhand Update	α	β	1	Secondhand Weight
3	Fading Update	0	0	Fading Weight	1

Handling secondhand information: each node periodically broadcasts its firsthand information table after *PT* (publishing timeout), in order to inform its one-hop neighbours about its direct experiences. This can be performed by setting Time-To-Live (TTL) field to 1. After receiving firsthand information from a node, which will become the secondhand information for the receiving nodes, the reputation system conducts a deviation test on each individual rating. In other words, other nodes' ratings are checked against the node's own reputation rating, if other nodes' experiences deviate too much from the node's own experience, the secondhand rating will not be accepted; otherwise the reputation rating will be updated.

Detection: in order to setup a criterion for the detection of misbehaving nodes, we have to setup a threshold to detect selfish nodes because the distinction between good and selfish nodes is very important. We use the following formula for the misbehaviour threshold.

$$M_T = \frac{\alpha}{\alpha+\beta} \qquad \text{Eq. 4-2}$$

After calculating the reputation of a node, the reputation is checked against the misbehaviour threshold, as shown in the Eq. 4-2 above. Reputations below the misbehaviour threshold indicate well-behaved nodes and these nodes are therefore provided with packet forwarding services, whereas reputations above the misbehaviour threshold are deemed to be misbehaving and the identity of these nodes is communicated to the path manager component in order to clean the routes from misbehaving nodes. This whole process is shown in Figure 4-3.

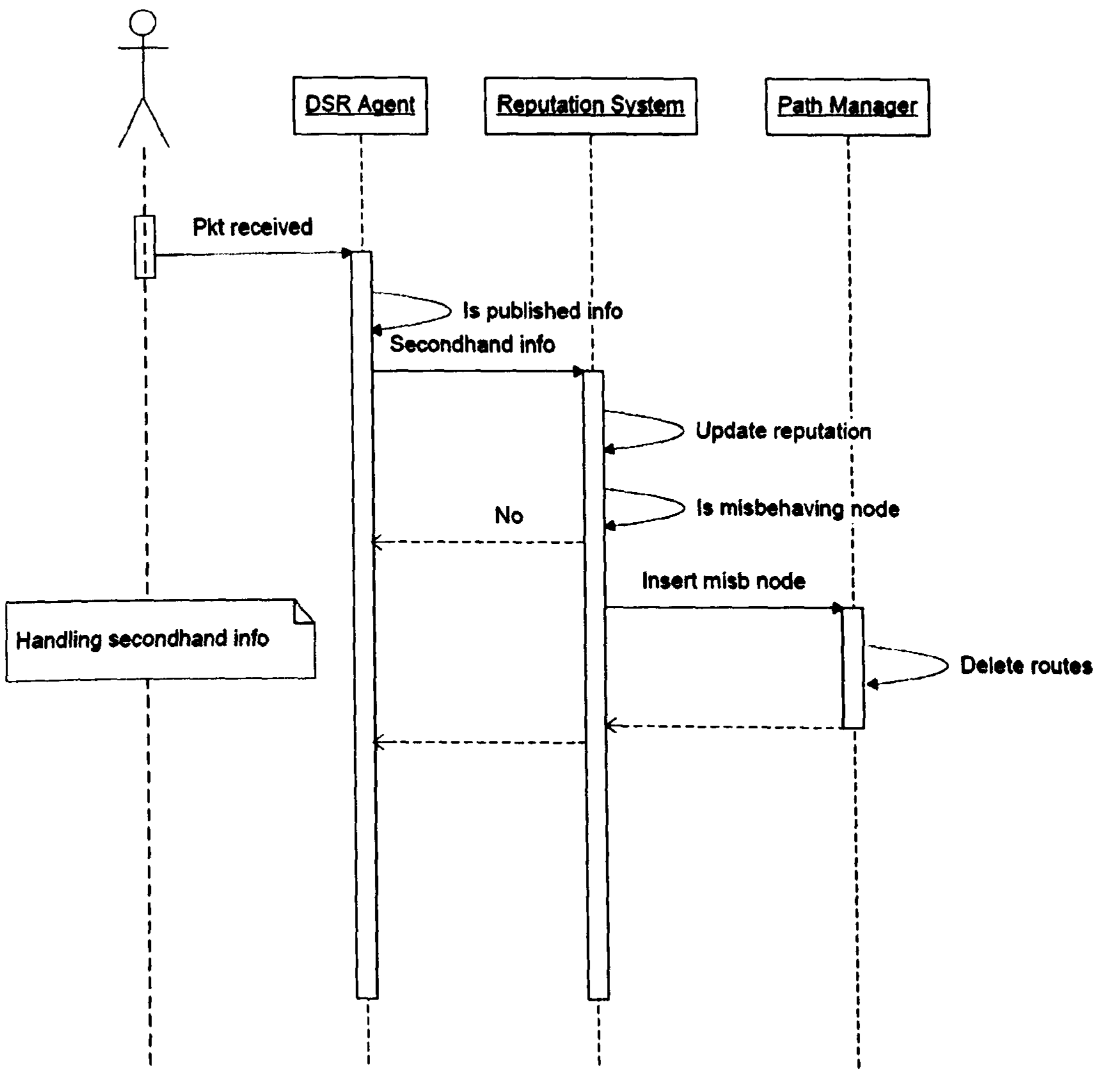

Figure 4-3: Secondhand information handling

4.5 The Path Manager

The path manager is originally responsible for maintaining the route cache of the DSR protocol. The reputation system interacts with the path manager to inform it whenever misbehaving nodes are detected. The path manager maintains a list of misbehaving nodes and provides the facility for insertion, deletion and searching a node address in the list. These types of query can be sent by the DSR agent or the reputation system. When a misbehaving node is detected and the path manager is informed of its identity, it should find out and delete all routes containing the misbehaving node. As a result, when a node searches for a route in the DSR route cache, the path manager only opts for safe routes that contains only good nodes.

4.6 Node Isolation

As discussed above that path manager will store the identity of misbehaving nodes and delete all routes containing these misbehaving nodes. This is considered as a first step of node isolation. The second step is that upon receipt of data or a route request packet, each node should check the source of the packet and then the DSR agent queries the path manager to establish whether the source of the packet is registered as misbehaving, as shown in Figure 4-4. If the data or the route request packet originated from a misbehaving node, the packet will silently be dropped. Eventually, misbehaving nodes will be gradually isolated from the network thereby reducing the overall selfish nodes' throughput in the network, as will be shown in the next section.

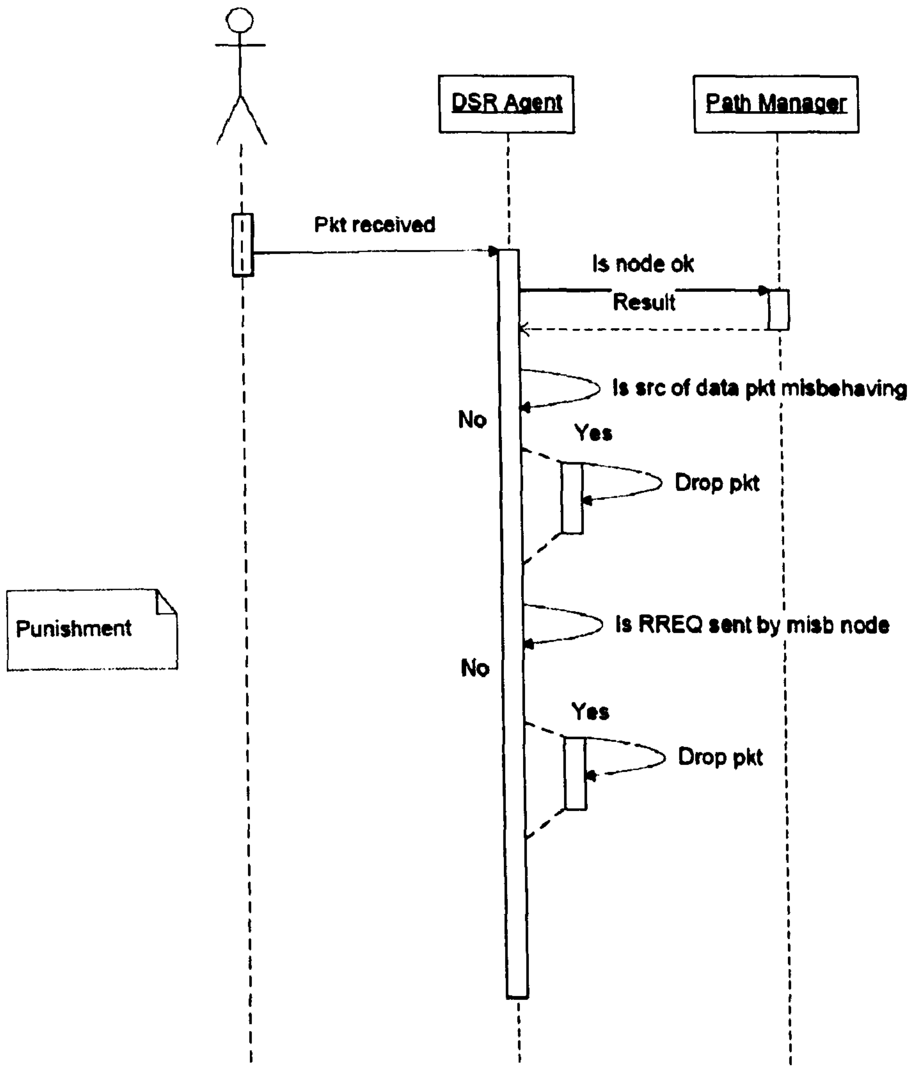

Figure 4-4: Node Isolation

4.7 Evaluation

In order to make sure that our designed reputation system achieves its primary goal, we evaluate it through simulation. As we mentioned in Section 4.1, the objective of a reputation-based system is to have a threshold that distinguish between good and selfish nodes, to encourage good nodes by providing them high throughput and to discourage selfish nodes by reducing their benefits (throughput). Once we find out that our designed reputation-based scheme achieves its goal, we will implement our proposed schemes (from Chapter 6 and 7) on top of it. Please note that the utility of a node can also be taken as a benefit of a node, as we did in the coming chapters, however, we will consider only throughput here.

4.7.1 Simulation Setup

We use Network Simulator NS-2.30 to implement the scheme and evaluate it using the parameters listed in Table 4-2. Additional parameters include: $M_T = 0.8$, $W_{DT} = 0.5$ seconds, $\omega = 0.2$, $\zeta = 0.9$, fading timeout = 60 seconds and $PT = 10$ seconds. Each node runs DSR as its routing protocol. Selfish nodes are selected randomly from all of the population. Each selfish node participates in the routing process by forwarding routing related *control* packets, such as route requests, route replies and route error messages. Selfish nodes drop all data packets received from other nodes for forwarding.

Table 4-2: Simulation Parameters

Parameter	**Level**
Area	1000m × 1000m
Maximum Speed	10 m/s
Pause Time	60 seconds
Radio Range	250m
Carrier Sense Range	550m
Number of Nodes	30
Number of Connections	20
MAC	802.11
Application	CBR
Packet Size	64 B
Simulation Time	900s
Movement	Random Waypoint Model
Placement	Uniform
Selfish Population	0% to 100%

The simulation results obtained are the averages from running 20 random scenarios for each point on the graph using random seeds. A comparison has been made between the DSR protocol and reputation incorporated DSR protocol. We evaluate the scheme using the following metrics.

4.7.2 Metrics

1. *Good Throughput.* The throughput available to good nodes. We calculate throughput as the ratio between the total number of data packets successfully received by destination nodes to the total number of data packets sent by the source nodes (at the application layer).

$$Good\ Thrput = \frac{Total\ Received\ Pkts}{Total\ Sent\ Pkts\ by\ Good\ Nodes}$$

2. *Evil Throughput.* The throughput available to selfish nodes.

$$Evil\ Thrput = \frac{Total\ Received\ Pkts}{Total\ Sent\ Pkts\ by\ Evil\ Nodes}$$

3. *Selfish Drop Rate (SDR).* The ratio of data packets dropped by selfish nodes to all dropped packets.

$$SDR = \frac{Total\ Selfishly\ Dropped\ Pkts}{Total\ All\ Types\ Dropped\ Pkts}$$

4.7.3 Simulation Results

The throughput available to selfish nodes in the defenceless DSR is comparatively greater than that of the evil throughput of the reputation enabled DSR, as shown by Figure 4-5. However, the difference tends to lower when the percentage of selfish nodes increases in the network. This is due to the fact that even selfish nodes' packets are dropped by the other selfish nodes and hence do not reach their corresponding destinations.

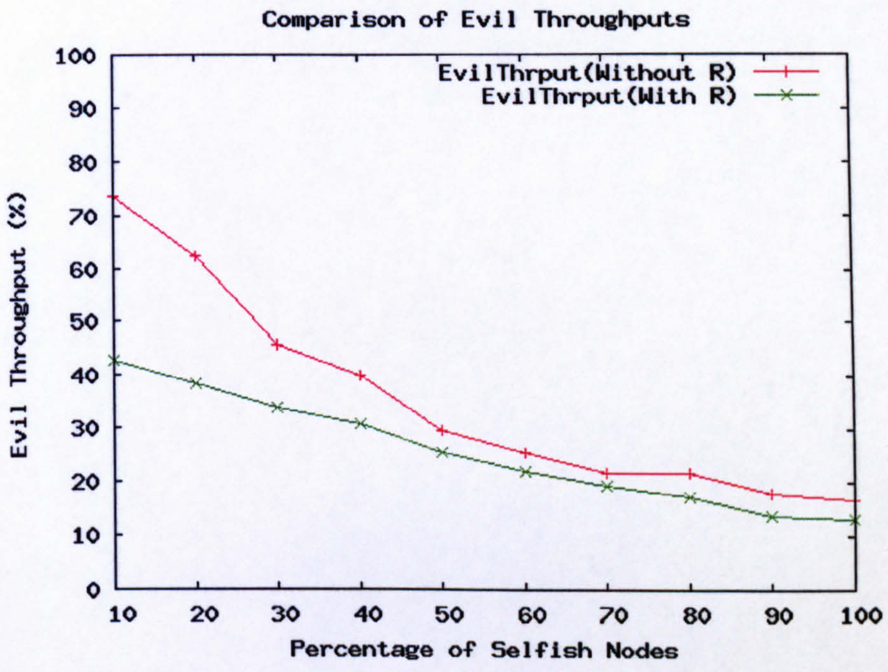

Figure 4-5: Comparison of Evil Throughput

Likewise, the throughput available to good nodes in reputation enabled DSR is also relatively stable as compared to that of the good throughput using the defenceless DSR protocol, as depicted in Figure 4-6. However, the good throughput in reputation enabled DSR decreases when the percentage of selfish population increases. The good throughput vanishes when the selfish population increases to 100 percent. This is due to the fact that all nodes becomes selfish: there are no good nodes left in the network. The reputation enabled DSR maintains a relatively stable and higher good throughput than the defenseless DSR, especially when mobility is moderate.

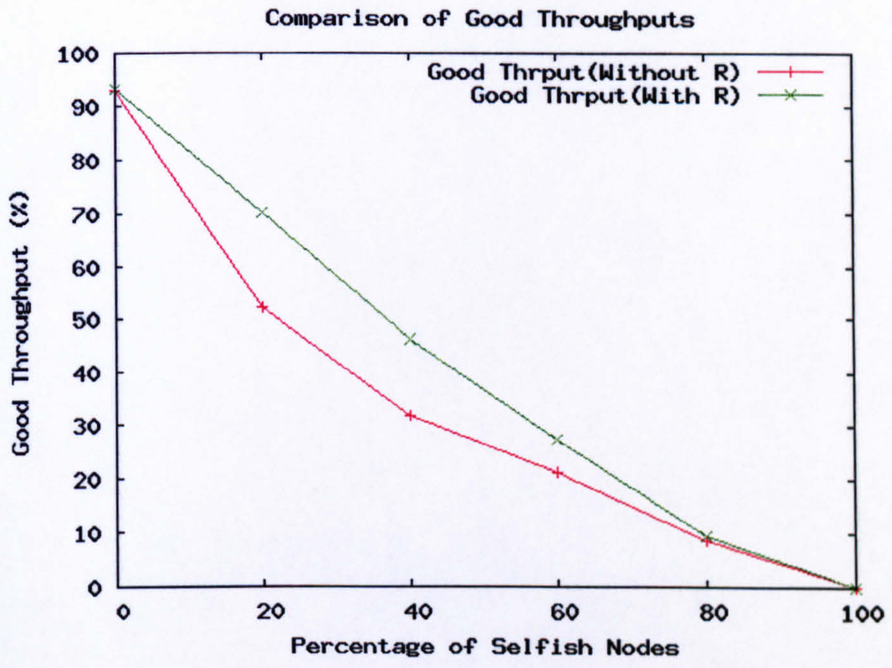

Figure 4-6: Comparison of Good Throughput

As shown in Figure 4-7, the reputation mechanism incorporated into the DSR protocol significantly reduces the selfish drop rate in the network. As the percentage of selfish population increases the selfish drop rate in the reputation enabled DSR decreases slightly. This is due to the fact that other drops in the network increase; for example when selfish nodes are detected, every other node will drop their data and route request packets in order to isolate them from the network. The good throughput ceases when there is no packet forwarder left in the network.

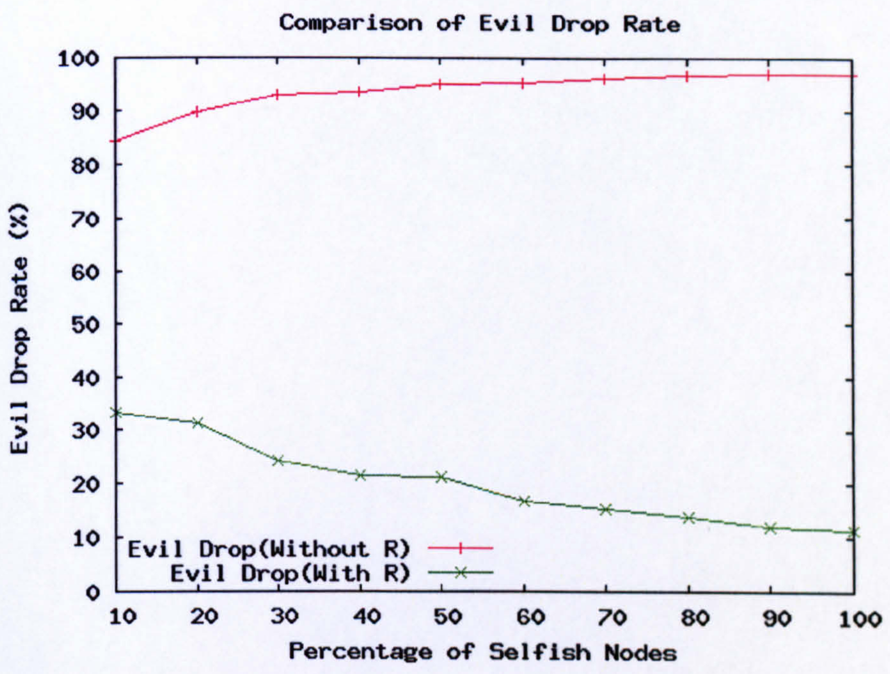

Figure 4-7: Comparison of Evil Drop Rates

4.8 Summary

In this Chapter, we presented our general reputation-based scheme for MANETs. We demonstrated its various components with the help of diagrams and their interaction among each other in order to detect and isolate selfish nodes in the network. We have shown through simulation that the designed reputation-based system successfully satisfied its main objectives. Firstly, it differentiates between good and evil nodes by detecting misbehaving nodes in the network. Secondly, selfish nodes are discouraged by reducing their benefits. Thirdly, good nodes gain higher benefits than the selfish nodes; hence there is advantage in cooperation. We will use our proposed reputation-based scheme in order to show in the coming chapters that selfish nodes can apply certain strategies to bypass the detection and gain higher benefits in the network, for example by changing identities or interacting directly with the source or destination.

Chapter 5 The Effect of Direct Interactions

5.1 Introduction

Selfish nodes are always the *intermediate* nodes along a path. They do not forward packets for others because there is no benefit for them in doing so. This selfish behaviour of nodes potentially effects throughput of the network. This issue has been considered by a number of authors by showing that a small percentage of selfish nodes can significantly reduce the overall throughput of the network [15, 25, 38, 54].

As discussed in Section 4.1, the main objectives of reputation-based schemes is to detect selfish nodes through a misbehaviour threshold, maintain network throughput by enabling nodes to construct paths that only include good nodes and discourage selfish nodes by restricting their services, *i.e.* selfish nodes are not provided with packet forwarding services anymore by the good nodes acting as *intermediate* nodes. However, selfish nodes can still gain benefit in the form of throughput and utility from the network by interacting *directly* with the destination nodes by exploiting mobility.

In the case of direct interactions, there is no intermediate node along the path to drop the selfish nodes' data packets as it is the strategy of reputation-based systems to decrease selfish nodes' throughput and utility. This has the effect of significantly improving the overall throughput and utility of both the good nodes as well as the selfish nodes. Figure 5-1 elaborates on this by depicting all possibilities for direct interactions between nodes. Direct interactions can cause problems which will be discussed in this chapter from two different facets or perspectives, *i.e.* reputation system security and the biasness in their experimental results. In the first one, we will look at direct interactions as a selfish node's strategy for increasing its benefits after being detected by a reputation-based system. In this case selfish nodes can strategically select their locations where they can interact directly with their source or destination. We propose methods to counteract and mitigate it. In the second one, we will look at the confusion caused due to the direct interactions when reputation-based schemes are evaluated under certain metrics, such as throughput, utility, network delay and overhead. We propose throughput and utility categorization technique in order to mitigate the confusion caused by direct interactions during evaluation. We will investigate and demonstrate this using CONFIDANT [25-26] as a benchmark scheme for our case study.

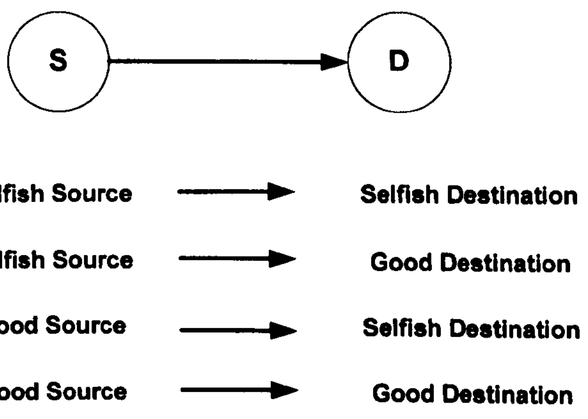

Figure 5-1: Possible ways for direct interactions to occur among nodes

This chapter is organized as follows. In Section 5.2 we will cover relevant background material and related work that has been proposed in the research community. In Section 5.3 we overview the benchmark scheme called CONFIDANT which we use for our analysis. In Section 5.4 we will highlight the potential problems caused by direct interactions that should be addressed when considering mobility in simulation-based evaluations of reputation-based schemes in MANETs. In Section 5.5, we give possible remedies for reducing direct interactions. Section 5.6 presents our analysis and results. We conclude the chapter in Section 5.7.

5.2 Background

In a MANET environment, reputation information is locally evolved through monitoring packet forwarding activities using passive acknowledgments (*e.g.* promiscuous mode listening). Schemes such as CONFIDENT, CORE and others [25, 38, 54] further share this information with neighbours to collaboratively detect and isolate selfish nodes. On the other hand, some schemes such as LARS [66], and OCEAN [67] rely only on local information to detect and isolate selfish nodes without considering secondhand reputation information. Due to ambiguous collision and receiver collision problems in passive acknowledgment based monitoring, two-hop (explicit) acknowledgment was proposed by Kejun *et al.* [56] to overcome these problems at the cost of increased communication overhead. In order to reduce overhead, Zhao and Delgado-Frias [57] propose a scheme that combines the multipath routing and single path data transmission with an end-to-end feedback mechanism; however it only detects misbehaving *paths* with no punishment strategy for individual misbehaving nodes.

Li and Wu [86-87] identify that mobility can increase the scope of interactions and recommendation dissemination thereby speeding up the overall trust convergence. Pirzada *et al.* [27] compare the performance of trust-based reactive routing protocols in which it is further shown that mobility increases throughput in the network (in the presence of selfish nodes) due to the increased interactions among nodes; however they did not explore this fact that the increase in throughput is actually caused by the DIs of nodes. Their work differs from ours in that they show the effect of mobility on throughput; however, they do not consider direct interactions among nodes while evaluating their schemes.

5.3 Overview of the CONFIDANT Scheme

Buchegger and Boudec proposed an extension to the Dynamic Source Routing (DSR) protocol called CONFIDANT [25] (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc

NeTworks). It has four main components: the Monitor, the Trust Manager, the Reputation System and the Path Manager. The Monitor promiscuously listens to neighbours to observe their behaviour, for example, packet forwarding behaviour *etc.* Every time a source node sends a packet to the destination node via intermediate nodes, the source node holds a copy of the packet in memory until it overhears the same packet forwarded by the intermediate node before time W_{DT} (watchdog timer) expires. The source node increases the reputation of the next node when it is confirmed that it has forwarded its packet, and decreases the reputation otherwise. The Trust Manager is responsible for sending an ALARM to all its friend nodes when misbehaviour is detected, and other trust managers receiving ALARM messages determine the trustworthiness of the message by scrutinizing the trust level of the sender. Based on this information the Reputation System maintains a local rating list and a blacklist, and further exchanges these lists with friends. Finally the Path Manager evaluates paths according to the reputation of nodes along the path and discards paths that include misbehaving nodes. Since this protocol allows nodes in the network to send ALARM messages to each other, it could increase the opportunity for attackers to send false alarm messages that a node is misbehaving while this isn't actually the case, a process referred to as rumour spreading [55].

The authors enhance their scheme in Buchegger and Boudec [26] by using a Bayesian model that classifies and rules out liars. In order to cope with rumour spreading, a trust rating mechanism is introduced that represents the trustworthiness of a node. Each node maintains a trust rating, a firsthand rating and a reputation rating for every other node. Only firsthand information is exchanged with neighbours. Secondhand information and trust ratings are never exchanged. Nodes build their reputation tables based on their own firsthand information, as well as the secondhand information received from neighbours. To reduce the effect of liars who publish false reputation ratings, a secondhand rating is accepted by the receiver only if it is compatible with the current reputation rating. For compatible reputation information, the trust rating of the publisher is increased and in case of incompatible information the trust rating is decreased. We implemented this version of CONFIDANT for our simulation study since it is robust and effective.

5.4 Problems Caused by Direct Interactions

5.4.1 A Selfish Node Strategy

It is believed that mobility increases the capacity of wireless ad hoc networks, as pointed out by Grossglauser and Tse [88] where selfishness in the network was not considered. However, selfish nodes can exploit mobility to increase their throughput and utility, even in the presence of a cooperation enforcement scheme. Crowcroft *et al.* [89] demonstrate that the geographic location of a node has a heavy effect on cooperation enforcement schemes. They observed in their study that nodes located at the middle of the network can have better services than those at the edges. Figueiredo *et al.* [90] argue that mobility decreases the capacity of wireless ad hoc networks by evaluating the strategies of selfish nodes for increasing their benefits. Using game theoretic models, they study the incentive mechanisms (reputation and credit) and show that if users strategically choose their positions in the system, network performance degrades significantly. They pointed out that when users apply their best strategies, the outcome would be the topologies where nodes are either tightly clustered together (in the case of credit-based scheme) or spread into a chain (in the case of reputation-based scheme).

In a similar way, a selfish node can exploit mobility and interact directly with a destination node to increase its benefits without being punished by the reputation-based system. In Section 5.5.1, we propose a general solution in order to discourage such types of selfish node activity.

5.4.2 Confusion in Evaluation

Reputation-based schemes are usually evaluated based on measurements of the utility and throughput [25-26, 38] of nodes. Other metrics have also been considered [56, 91] such as delay or the routing overhead of the network. These metrics are affected when the schemes are evaluated in environments where there are high levels of direct interactions between nodes in the network. We will show in Section 5.6 that the commonly adopted simulation parameters used by the research community for the evaluation of reputation-based schemes usually produce high levels of direct interactions that can cause confusing results. The following are the metrics affected by direct interactions.

Utility

In fact the direct benefit or utility relates to the source and destination of a packet only, and consequently intermediate nodes are naturally inclined to act selfishly and save power for their own communications rather than spending it on forwarding packets for others. Altman *et al.* [92] and Buchegger *et al.* [25] define the benefit or utility of a node as follows.

$$u_i = b_r \sum P_{\text{received}} + b_s \sum P_{\text{sent}} - c_f \sum P_{\text{forwarded}} \qquad \text{Eq. 5-1}$$

Where u_i is the utility of a node i, b_r is the benefit when i receives a packet as the destination node, b_s is the benefit acquired by i when its packet successfully reaches its destination and c_f is the cost (in terms of memory, bandwidth and CPU usage) incurred by node i of forwarding a packet for others. The exact values of benefit and cost variables (b_r, b_s and c_f) depend on energy models, CPU and memory usage per packet, but the value of each one is in the range [0, 1]. One of the main focuses of reputation-based schemes is on decreasing the utility of selfish nodes and increasing the utility of good nodes in the network. Utility is likely to increase in networks with more direct interactions between sources and destinations. We will show in Section 5.6 that the commonly adopted simulation parameters in mobile environments produce a substantial amount of direct interactions. This increases the nodes' utilities which can cause a misjudgement about whether the utilities are increased or decreased due to the direct interactions or due to the designed scheme.

Throughput

The main objective of reputation-based schemes is to maintain reciprocity in the network. This is performed as each node provides services to good nodes only. As a result, good throughput is increased and evil throughput is decreased. In ad hoc networks packet forwarding services are provided by the intermediate nodes. Despite the punishment and restrictions imposed on selfish nodes, they can still increase their throughput by interacting directly with a source or a destination (this, by the way, can also happen with the good nodes). If a selfish node takes a position as shown in Figure 5-1 acting as a source and/or destination for some time during its simulation lifetime, its originated or received packet count will be considerably increased; hence increasing its overall throughput. If this situation prevails in the network, it can cause confusing results as we will highlight in Section 5.6. That is, it is not clear whether the good and evil throughputs are being affected primarily by the direct interactions or by the reputation-based scheme.

Delay and Routing Overhead

The direct throughput gained also affects the delay and routing overhead of the network. When a source node directly interacts with a destination node there is no intermediate node involved in the traffic forwarding, which reduces the overall number of forwarding events in the network and hence reduces the overall routing overhead. It also affects the delay of the network because there are only source and destination nodes in the routing path and hence low delay results. In other words, increased direct throughput produces reduced delay in the network. Reputation-based schemes considering delay and routing overhead as metrics, can lead to confusion in understanding the results. In particular, it becomes harder to establish whether the reduced delay and routing overhead is caused by the direct interactions or by the designed reputation-based scheme.

5.5 Possible Remedies

It is important to reduce the direct interactions or at least consider them while evaluating reputation-based schemes. Alternatively, new strategies should be devised for when there is no intermediate node between a source and a destination. The following are possible remedies which may help reducing direct interactions or at least allow nodes to be treated fairly when interacting directly. The first method can be used to mitigate the effect when selfish nodes exploit direct interactions for their own benefit, while the latter two methods can be used to reduce the confusion caused by direct interactions.

5.5.1 Contribution to Consumption Ratio (C2CR)

In P2P networks, Contribution to Consumption Ratio (C2CR) plays an important role when network resources are allocated to nodes, for example allocating download limit, download speed *etc.* to nodes. The same concept may be used here to judge every node based on this C2CR value. By contribution we mean the data forwarding service provided by a node and by consumption we mean the use of other nodes' data forwarding services. The C2CR is different from reputation because reputation is the measurement of cooperation of a node, it does not say anything about the overall service consumption of that node. The C2CR on the other hand may reveal the overall behavioural history of a node in the network, *i.e.* how much data forwarding service is provided to the network and how much the same service is used by the node itself. Using reputation-based schemes, a node can enjoy low contribution and high consumption of the network resources, such as packet forwarding service by intermediate nodes, without being detected by the reputation system. For example, a node can aim at low contribution by performing selective dropping, dropping route request packets, interacting directly with its source or destination or providing services to one node and refuse them to another. C2CR can help to identify such dishonest nodes in the network. In MANETs, the C2CR of a node *n* can be calculated by overhearing all the packets forwarded by *n* and all the packets originating from *n*, formulated as follows.

$$C2CR(n) = \frac{Pkt\_Forwarded(n)}{Pkt\_Originated(n)} \qquad \text{Eq. 5-2}$$

Every node logs this information and shares it with its neighbours. Every source and destination interacting directly should provide quality of service (QoS) based on the C2CR value. The higher the C2CR value of a node the better the QoS it will get and the lower the delay it will experience. For example, if a source node asks for a file to download from a destination in the network, the destination node will intentionally delay the file based on the C2CR value of the source node. The request from a node will be ignored if its C2CR value is less than a certain threshold. In this way a source directly interacting with a destination can be assessed and can be fairly treated in terms of services. As a result a selfish node cannot increase its throughput or utility by interacting directly with the destination node.

5.5.2 Tweaking Simulation Parameters

The most commonly used simulation parameters for the evaluation of reputation-based schemes can cause substantial direct interactions among nodes in the network, as shown in Section 5.6, including the simulation parameters used for CONFIDANT. The radio transmission range and simulation area are usually taken as 250 meters and 1000 meters squared respectively by researchers for evaluation and testing; some authors such as Zakhary and Radenkovic [93] use 750 meters squared with 250 meters radio range. Some simulators, such as NS-2, use a 250m radio range as a default value. This 250m radio range in a 1 km squared area potentially increases the possibility of a source interacting directly with its destination in a mobile environment, as shown in Figure 5-2. This is due to the fact that one node having a 250m radio range can cover a diameter of up to 500m, while only three nodes (2-hops) can easily cover a distance of more than 1 km. In a mobile environment such nodes can easily interact directly as a source and destination. It is suggested that reducing the radio transmission range may help reduce direct interactions. However, establishing an optimal radio transmission range that produces high throughput and low collision rate in all situations is a difficult task. In order to avoid this problem, we discuss a promising method in the next section.

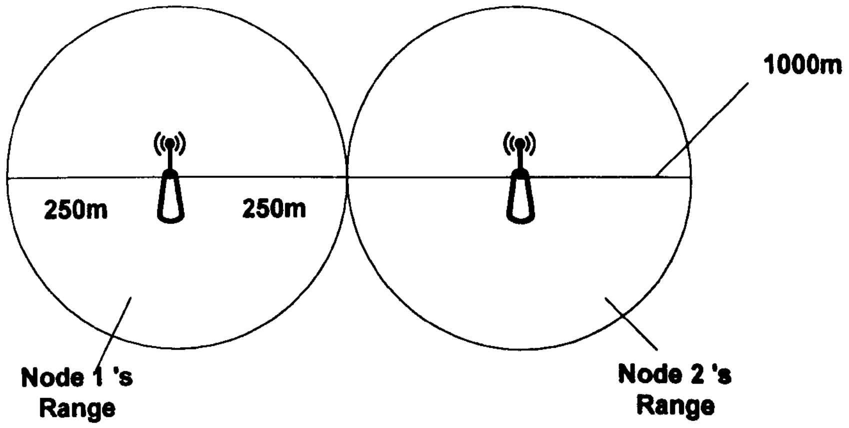

Figure 5-2: 1Km Radio Range Occupied by Two Nodes

5.5.3 Utility and Throughput Categorization

Throughput and utility categorization can be used to reduce the ambiguity and confusion caused by direct interactions. This kind of solution is not appropriate for the counteraction of selfish nodes that use direct interactions as a strategy to bypass the detection process. However, we will show in the next section that this method is suitable for reducing the prevalence of confusing results. Since high levels of direct interaction produce high evil and good throughput in a network, categorizing utility and throughput into direct and indirect[6] forms will help clarify the evaluation of a scheme. In other words, when evaluating the throughput of a reputation-based scheme, it may be better to isolate the regular (indirect) throughput from the direct throughput (based on direct interactions) to establish whether the throughput is increased due to the designed scheme or as an effect of the direct interactions.

In the next section, we implement a widely used reputation-based scheme, called CONFIDANT, to confirm how much throughput is the result of these direct interactions. Our simulation studies show that in the presence of 20% selfish nodes and in a continuous mobile environment, 59% of the evil throughput and 29% of the good throughput is gained through direct interactions among nodes. These direct interactions accordingly affect the delay and routing overhead in the network.

5.6 Simulation Based Analysis

5.6.1 Setup

We use Network Simulator NS-2.30 to implement the CONFIDANT scheme and evaluate it using the parameters listed in Table 5-1. These parameters are commonly used in the research community. In this simulation study our aim is to find out how often a source node interacts directly with a destination node and to what extent it affects the throughput, utility, delay and

[6] We will refer to the evil and good throughput as throughput available to selfish nodes and good nodes respectively. By direct and indirect throughput we mean that throughput gained through direct interactions and without direct interactions of a source and a destination respectively.

routing overhead. We categorize the throughput and utility of the network to improve the clarity of the results obtained.

Table 5-1: Simulation Parameters

Parameter	Level
Area	1000m × 1000m
Maximum Speed	10 m/s
Pause Time	0 to 1000 seconds
Carrier Sense Range	550m
Number of Nodes	50
MAC	802.11
Application	CBR
Packet Size	64 B
Simulation Time	900s
Movement	Random Waypoint Model
Placement	Uniform
Selfish Population	20%

5.6.2 Mobility Model

We use a random way-point movement model [94-95] for our mobile scenarios used in our simulations. In this model a node first waits for the pause time duration and then moves on to a randomly chosen position with a random velocity. The process is repeated until the simulation ends. A zero second pause time represents continuous mobility; we increase this up to a 1000 second pause indicating a static network (since each simulation run lasts for 900 seconds).

5.6.3 Metrics

We evaluated the CONFIDANT scheme using the following metrics.

1. *Throughput.* The ratio between the total number of data packets successfully received by destination nodes to the total number of data packets sent by source nodes at the application layer.

2. *Good Throughput.* The throughput available to good nodes.

3. *Evil Throughput.* The throughput available to selfish nodes.

4. *Direct Good Throughput.* The throughput achieved by good nodes through direct interactions, with no intermediate nodes involved in the path.

5. *Direct Evil Throughput.* The throughput gained by selfish nodes through direct interactions.

6. *Real or Indirect Throughput.* The throughput achieved without direct interactions.

7. *Utility.* The benefit a node can get from the network, the formula is shown in Eq. 5-1. We used the value of b_r, b_s and c_f to be 1 in order to measure the utility in packets only. A negative utility value means that a node incurs an overall loss.

8. *Evil Utility.* The utility gained by evil nodes.

9. *Good Utility.* The utility achieved by good nodes.

10. *Direct Evil Utility.* The sum of P_{received} and P_{sent} where the source, destination or both are selfish and happen to be interacting directly. Please note that in our scenario selfish nodes do not forward any data packets, hence $P_{\text{forwarded}}$ doesn't apply here.

11. *Direct Good Utility.* The sum of P_{received} and P_{sent} where the source, destination or both are good nodes that happen to be interacting directly. Please note that our aim here is to show that a considerable amount of sent and received packets are gained from direct interactions, hence we do not plot $P_{\text{forwarded}}$.

12. *Delay.* The total time (in seconds) taken by data packets to reach their destinations divided by the total number of packets.

13. *Routing overhead.* The total number of sent and forwarded events on the routing layer.

5.6.4 Analysis

Figure 5-3 shows the effect of selfish nodes on network throughput, with 60 seconds pause time. If all nodes are selfish, throughput still exists in the network due to the direct interactions; this throughput will be increased when the pause time reduces. As shown in Figure 5-4, both the *normal* evil and good throughputs are increased due to mobility where the increase in evil throughput is more than that of the good throughput. By normal we mean the throughput that is normally shown by reputation-based schemes in the literature, which is a mixture of both direct and indirect throughput. The evil throughput peaks at a 600 second pause time. One of the reasons for this is that the evil throughput is also affected by the actual act of the node pausing: during a pause when a selfish node is detected by its neighbours, it then moves to a new neighbourhood after the pause where it is likely to be treated as a new node, and hence starts receiving services again. However, more than one pause throughout the simulation lifetime can increase the chance of repeating interactions among nodes and hence faster detection. For example, a selfish node detected in one neighbourhood (after having paused there) may interact with the same nodes again without having any services a second time. The 600 seconds pause may be the optimal time for a selfish node to stay static along with other nodes, after which it then starts moving for the remaining period of 300 seconds until the end of the simulation.

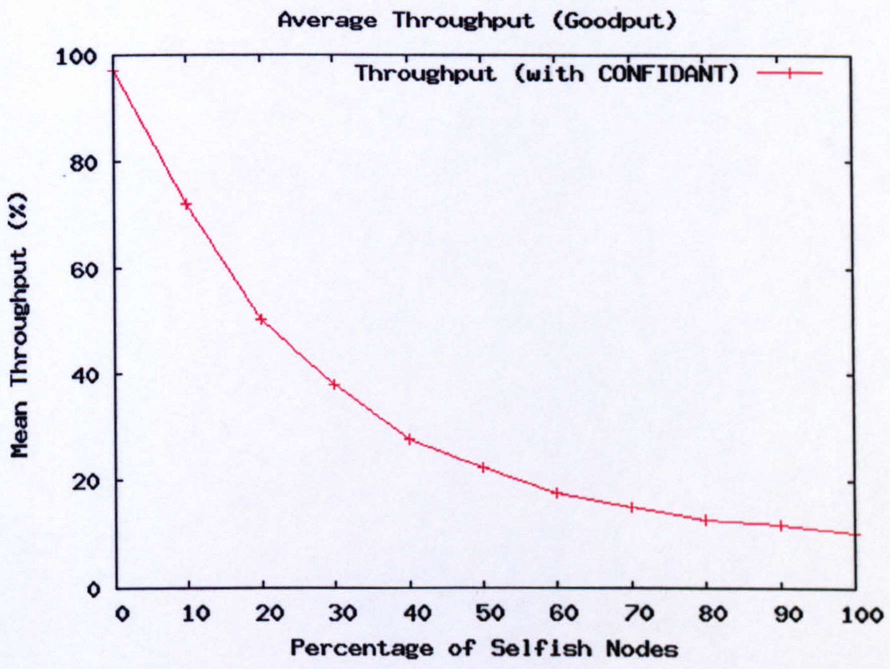

Figure 5-3: Average Throughput with respect to % of Selfish nodes

In Figure 5-4, the direct evil and direct good throughputs also increase with the increase in mobility and here again the increase in direct evil throughput is affected more than the direct good throughput. This is because the number of good nodes is greater than the number of selfish nodes. With a zero second pause time, 59% of the evil throughput is gained through direct interactions and 29% of the good throughput is achieved through direct interactions. In order to clarify this fact further, we found out that in 41.75% evil throughput, 24.75% is achieved through direct interactions while in 55.96% good throughput, 16.24% is gained through direct interactions.

So the real benefit to a selfish node when a cooperation enforcement scheme is in force is considerably lower than the results suggest, after the direct interactions are filtered out, as shown in Figure 5-5. Likewise, the good throughput is also increased due to the direct interactions of nodes. This difference suggests that in work related to the effectiveness of reputation schemes, there is a need to make clear whether the increased good throughput is due to the direct interactions or the cooperation enforcement scheme itself. In other words, in order to evaluate reputation-based schemes, it's important to filter out the direct throughput from the indirect throughput, in order to be clear about how much good throughput is increased due the scheme being assessed. Likewise, this also helps to establish how much evil throughput is reduced due to the reputation-based scheme. In short, categorizing throughput and utility clarifies that (i) the actual evil throughput is much lower and (ii) the good throughput is also lower than that claimed.

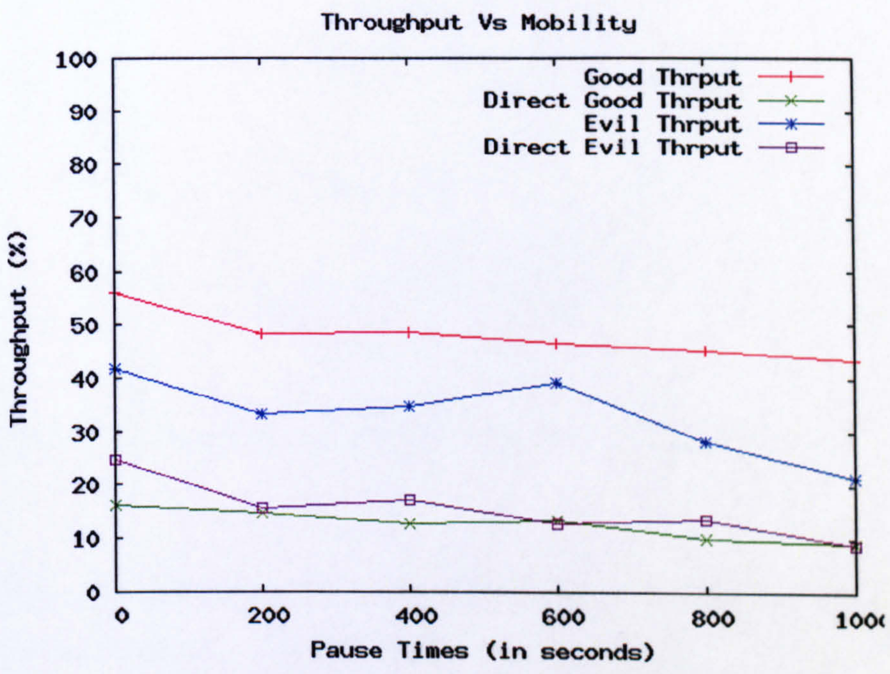

Figure 5-4: Direct and Indirect Throughput *vs.* Mobility

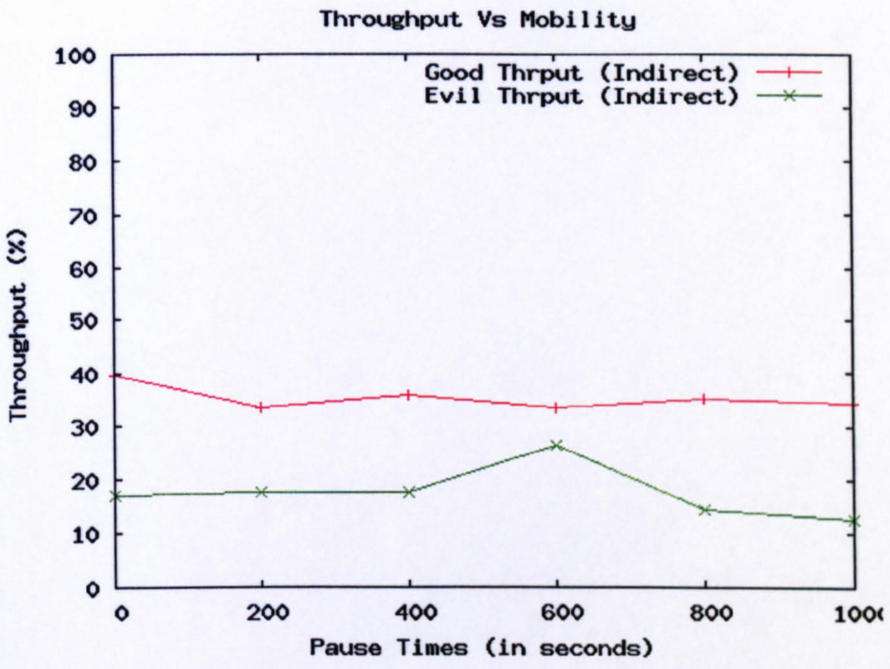

Figure 5-5: The Indirect Good and Evil Throughput vs. Mobility

As shown in Figure 5-6 and Figure 5-7, there is considerable amount of evil and good utility gained due to direct interactions which can cause confusion for the developers as well as for the readers. In the scheme evaluation a developer may be frustrated (as I was) about the increase in evil utility (which is against the aim of the reputation-based systems); however, in reality this might not be the case. Because the evil utility contains direct evil utility which a reputation-based system do not have the mechanisms to prevent or even mitigate it. In the case of good utility, the actual indirect good utility might not be so much to claim the scheme to be efficient. The reason for the greater received packets than sent packets in Figure 5-7 is that the sent packets are recorded for good nodes only, *i.e.* for 80% of the population (because it is good utility) while the received packets are recorded for the good as well as evil nodes, *i.e.* all population.

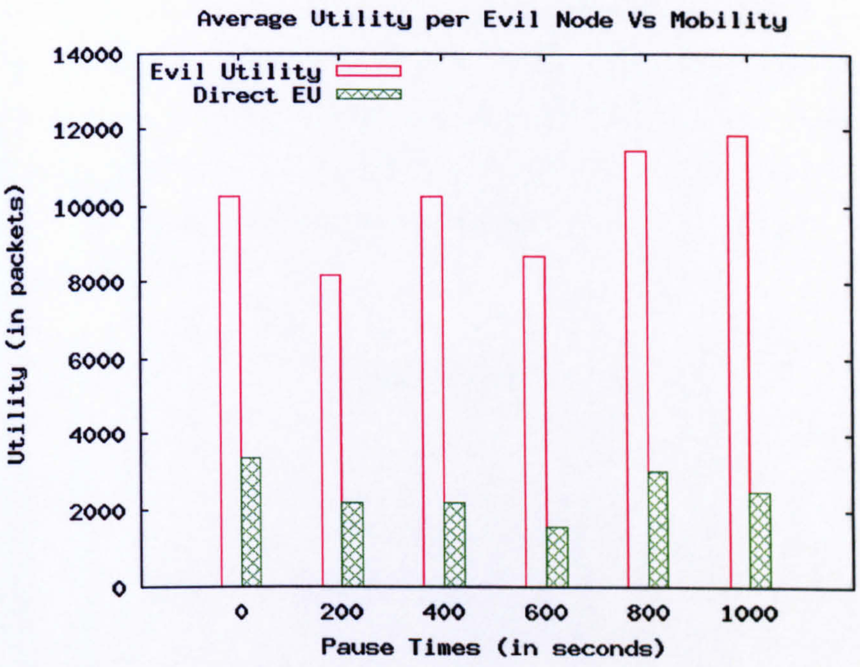

Figure 5-6: Direct and Indirect Evil Utility *vs.* Mobility

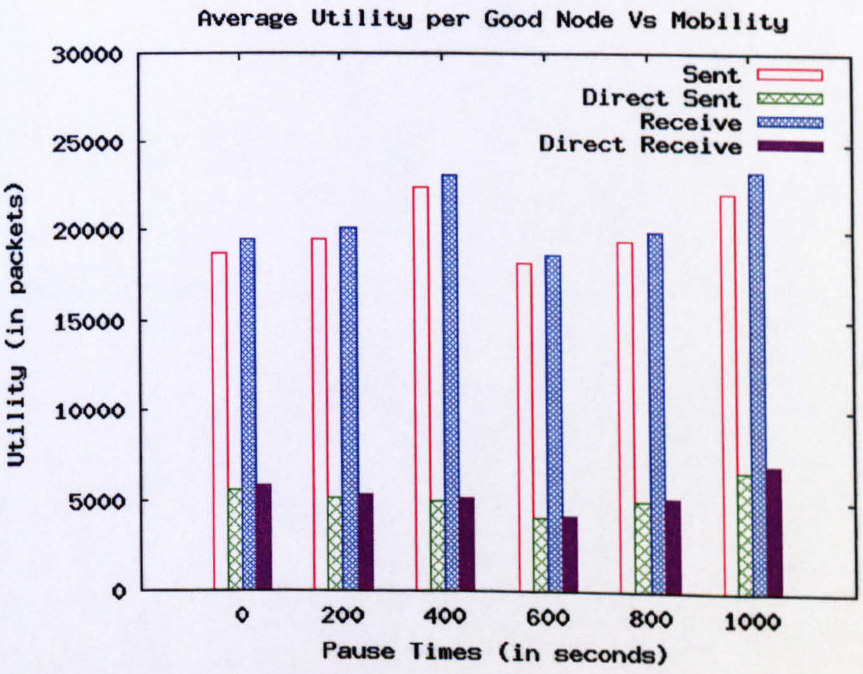

Figure 5-7: Direct and Indirect Good Utility *vs.* Mobility

When there is low direct throughput in the network, there are more forwarded events and hence higher average per packet delay. In other words, there is an inverse relationship between direct throughput and delay as well as forwarded events on the routing layer, as can be seen in Figure 5-8 and Figure 5-9. To evaluate the effect of a reputation-based scheme on the average delay and routing overhead in the network, it's also important to consider the direct throughput in the network.

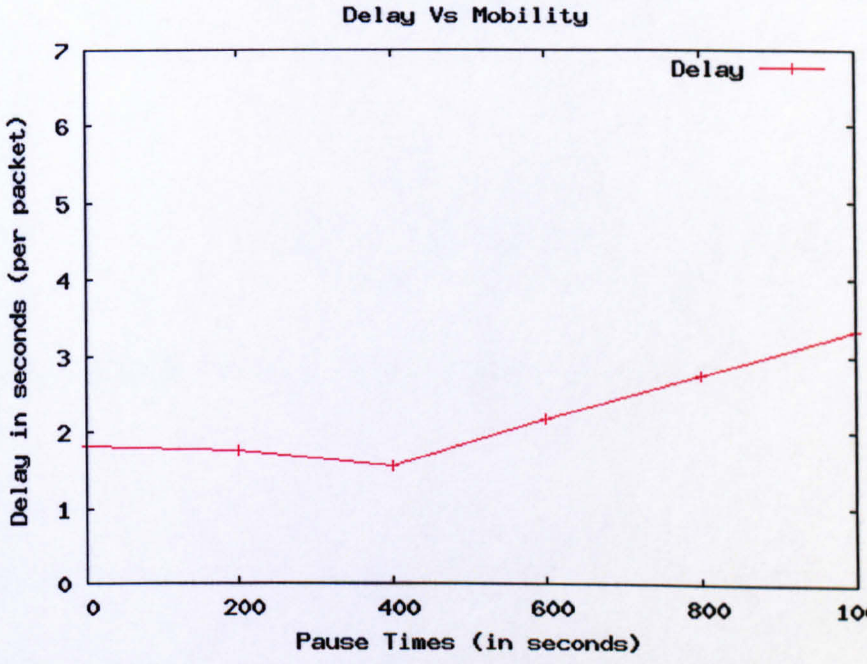

Figure 5-8: Delay Affected by Direct Throughput *vs.* Mobility

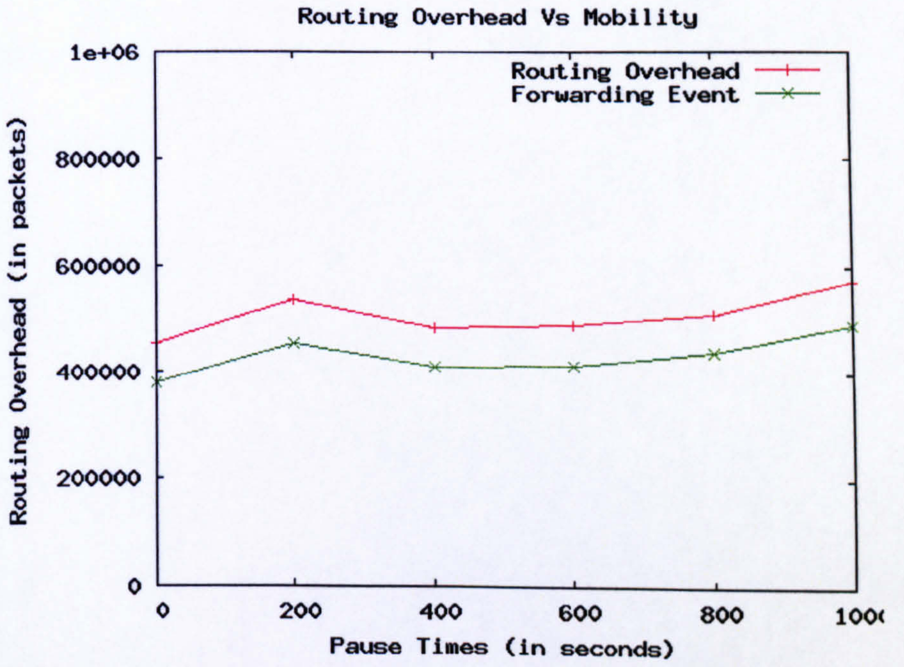

Figure 5-9: Routing Overhead Affected by Direct Throughput *vs.* Mobility

5.7 Summary

Mobility may cause strange problems in simulation-based evaluation of schemes, especially for reputation-based schemes. The main target of reputation-based schemes is to detect and isolate selfish nodes, increase good throughput and decrease evil throughput in the network. Due to mobility however, the chance of direct interactions of a source node and a destination node increases which can result in misjudgements about the effect on throughput in the presence of selfish nodes in the network. Apart from the natural occurrence of direct interactions, a selfish node can use them to increase their benefits. We have shown that there is a pronounced gain in throughput caused by direct interactions and that direct interactions also affect delay and routing overheads in the network. Based on the results provided, we can establish that it is important to reduce direct interactions or at least consider them while evaluating reputation-based schemes. Finally, we suggested some strategies that should be used to reduce direct interactions and to mitigate the potential confusion caused by them in the evaluation of reputation-based schemes. In the following chapters we will propose proactive and reactive schemes in order to tackle whitewashing and Sybil attacks.

Chapter 6 Deterring Whitewashing Attacks in MANETs

6.1 Background

The sole purpose of reputation-based schemes is to let selfish nodes bear the consequences of their bad actions in the form of isolation from the network while allowing good nodes to enjoy the network services. Through this selfish node isolation process, nodes are encouraged to cooperate. However, the open nature of MANETs enables a selfish node to change its identity and start over again with a fresh (new) identity; in this way a selfish node whitewashes its previous misbehaving history. This is called a whitewashing or identity changing attack. According to Hoffman *et al.* [60], "whitewashing attacks occur when attackers abuse the system for short-term gains by letting their reputation degrade and then escape the consequences of abusing the system by using some system vulnerability to repair their reputation". If identities are not persistent, this makes it difficult to hold selfish or malicious nodes accountable for their actions.

In order to enhance identity persistence some authors have suggested assigning the smallest possible initial reputation for newcomers or an entry fee per identity [68-70]. These techniques are efficient because they do not need any centralized trusted third party for identity management. However, the smallest initial reputation can still be manipulated because of the zero cost of identities. A monetary based entry fee per identity that might address this is, however, not suitable for open MANETs due to a number of problems. First, it causes complications in fee management. Second, it requires security mechanisms to secure the fee itself, such as tamper proof hardware. Third, fee payments and the fee structure itself represent extra burdens for users and the system; for example, incorporating a charged text message or monetary-based mechanism into the system is an extra activity for the users and the system itself. Due to these problems, we adopt an entry fee concept in our scheme but instead of a monetary fee system, we use fee as a form of work imposed on every newcomer. Each new entrant will expend some of its battery to pay an entry fee in the form of cooperation in the network before expecting the network to provide services. For a normal selfish node, it is no longer beneficial to perform a whitewash because it will pay an entry fee each time it re-enters the network. This represents a social cost [68] incurred by newcomers, *i.e.* newcomers are not welcomed; however it is still beneficial for the overall network performance. In monetary based fee systems, the payment process takes a specified or fixed time; however, in our non-monetary based fee system where cooperation is involved, the time required for payment varies and depends on the number of neighbours. Usually, in dense parts of the network the time variation is low. In addition to battery or cooperation, this unspecified or unfixed time may also be used as a deterrent to thwart the whitewashers and Sybil attackers.

Moreover our non-monetary based fee system is easily manageable on a technical level in MANETs (no virtual bank or centralized fee management system is required); it does not need any tamper-proof hardware in order to secure fees; and it uses packet forwarding as a form of fee payments, hence no extra entity needs to be incorporated into the system.

The scheme discourages whitewashers and Sybil attackers alike; because the fee enforced will be applied per new identity. Fee imposition makes whitewashing costly (in terms of battery power) for an attacker, the same is the case for Sybil attackers; hence an attacker can perform fewer whitewashes or identity rounds within the limits of its available battery power. Furthermore, the fee enforcement will also improve the overall system performance (*i.e.* network throughput and utility). This is due to the fact that fee payment implies contributing to the network by forwarding other nodes' packets.

In Section 6.3.6, we will show that our proposed scheme significantly reduces the evil throughput and evil nodes' utility as compared to our comparator reputation-based scheme CONFIDANT [25-26].

The rest of the chapter is organized as follows. In Section 6.2, we present our scheme, explaining how an entry fee is used as a deterrent against whitewashing attacks. In Section 6.3, we evaluate and compare our scheme through simulation results; we also highlight how the scheme is implemented in NS-2. The chapter is concluded in Section 6.4 with a discussion about future extensions to the scheme.

6.2 Deterring Whitewashing

6.2.1 Assumptions and System Overview

Firstly, we assume that the network is dense; hence we will not consider the issues related to boundary and network sparseness.

Secondly, we modified our designed reputation-based scheme from Chapter 4 in order to incorporate our non-monetary fee based scheme into it.

Thirdly, each node must pay the fee when joining the network; after this phase a node can start building its reputation, as depicted in Figure 6-1. Each node creates a fee count for the nodes to which it has either interacted directly or known about them from other neighbours when they publish their fee related (secondhand) information. The fee count is updated either through direct interaction experiences or when a forwarding event is observed through overhearing. Every forwarding event will be considered in the fee count (not a reputation count) until the fee is fully paid *i.e.* the fee threshold is met. In other words, if a node has not fully paid its fee and in the meantime it forwards a packet, then it will be considered as a fee update event. After the fee is paid, every forwarding event will be considered in the reputation count. Nodes will exchange their fee counts along with reputation information with their neighbours. We assume that nodes do not lie about these values; however we acknowledge this as an important issue that we aim to consider in future work. Route request messages and data packets will be dropped for the nodes that have not paid their fee.

Fourthly, fee enforcement will be initiated after the first detection. In other words, during bootstrapping, nodes will not consider each other as new nodes (we consider first identity to be valid); hence, two new nodes when interacting will ignore each other (we will explain this further in Section 6.3.3). They will either drop or forward packets based on their natural behaviour until our reputation-based scheme detects misbehaving nodes. This assumption is valid, since a whitewashing attack is usually meant to escape the consequences of misbehaviour or having a bad history.

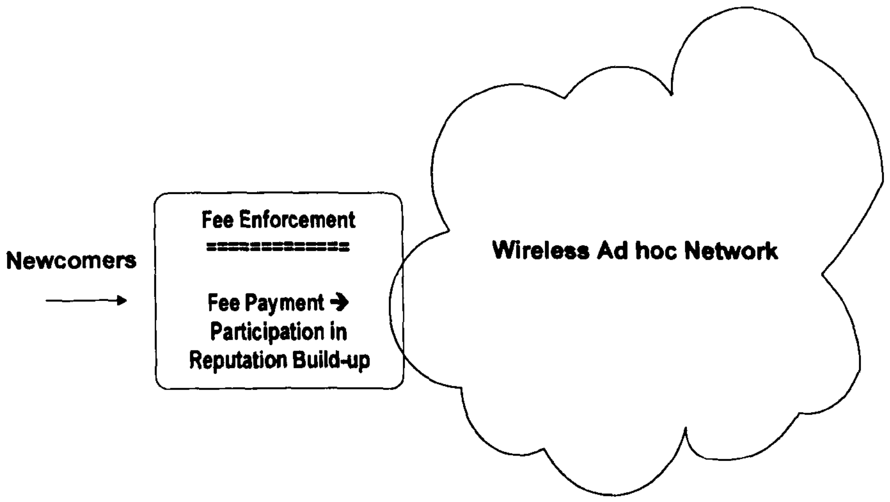

Figure 6-1: The process of a new node entering into the network

Finally, we categorize nodes in the network based on whether their fee has been paid. That is, we refer to nodes that have paid their entry fee as *mature* nodes and those who have yet to pay their fee as *new* nodes.

6.2.2 Design Rationale

In most existing schemes, newcomers are allocated some reputation to start with, called a *neutral reputation*, denoted by X as shown in Figure 6-2. The smallest reputation of a node, denoted by Z, is slightly greater than the detection threshold. A node having reputation Z (or higher) can exploit the services of the network without being detected as a selfish node. The chance of whitewashing increases when the current reputation n of a node is in the region α. The amount of the initial reputation, which is equal to $\alpha = X - Z$, can always be manipulated by an identity changer when there are no restrictions imposed on this region. In other words, in the absence of restrictions, this neutral reputation gives the opportunity for whitewashers to utilize network services without contributing to the network. This will further motivate selfish nodes to change their identities after they have been detected by a misbehaviour detection system. Some authors suggest assigning the smallest possible neutral reputation to newcomers; however, the smallest neutral reputation can still be manipulated because of the zero cost of identities with no restrictions.

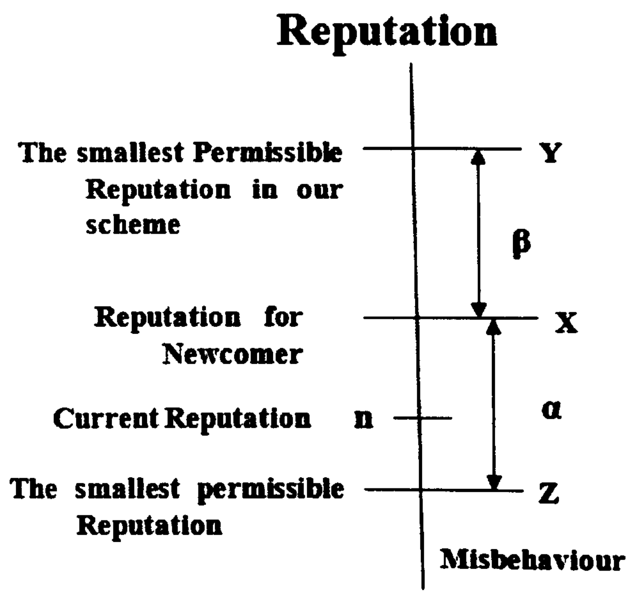
Reputation
The smallest Permissible Reputation in our scheme
Y
β
Reputation for Newcomer
X
Current Reputation n
α
The smallest permissible Reputation
Z
Misbehaviour

Figure 6-2: Reputation levels

For our scheme, we set the smallest permissible reputation *Z* to be greater than the node's initial reputation *X* by an amount *β*, denoted by *Y* (instead of *Z*). In our scheme we will use the threshold *Y*, the amount of reputation *β*, and the fee threshold interchangeably.

In light of the above modifications, we describe our scheme as follows. For a node to get services from the network its basic reputation must be above *Y*, *i.e.* it has to pay the entry fee. This is based on a simple principal: cooperate, before you are cooperated with. In other words, each newcomer will pay an entry fee in the form of spending its battery power for cooperation with other nodes that will consequently increase its reputation up to a certain threshold *Y* in its neighbours' reputation tables. In the meantime every node will periodically exchange fee and reputation information with its neighbours. After exceeding reputation *Y*, every node (except selfish nodes) will provide packet forwarding services to the new node because it has paid the entry fee. It is important to note here that after fee payment if a node misbehaves and crosses the misbehaviour threshold, it will still be liable to isolation from the network.

We will now discuss how the threshold Y will discourage whitewashing or identity changing attacks. As shown in Figure 6-2, we increase the basic level such that the smallest permissible reputation will be Y. In our design a node cannot manipulate the neutral reputation because services will only be provided to those nodes with reputation greater than Y. Neutral reputation or reputation less than Y is just an indication that the fee has not been paid or is yet to be paid. Mostly nodes are selfish in open MANETs and they are inclined towards their own benefits. We have designed our scheme in such a way that it will always be beneficial for a node not to change its identity after achieving Y provided that the attacker objective's cost is less than its benefits. In other words, a node will always incur a loss when changing its identity after gaining Y. By definition of our scheme

$$X < Y,$$

Therefore

$$X - Y = \beta < 0 \qquad \text{Eq.6-1}$$

Here there will always be an amount β of loss to reputation or the fee that has been paid, after a node changes its identity.

6.2.3 Analysis of the Deterrence

In generic communication models for wireless ad hoc networks, a source node creates a message and stores it in memory, in order to send it to a destination node. Assuming that the path to the destination is already established, the source node will transmit to the nearest neighbour that falls along the path. Suppose all this processing and transmission of a packet consumes an amount E_T of energy (*i.e.* battery power). We assume that on receiving and forwarding a packet, E_R and E_F amount of energy is consumed respectively. In order to maximize the life span, a selfish or a malicious node tries to engage itself in activities that cost it low battery consumption. Assuming that ΔE is the minimum energy a selfish or a malicious node can consume by choosing among the activities, such as transmission, reception or forwarding. Let $E_{\max}(n)$ be the maximum battery power available to node n.

Then the maximum lifetime of node *n*'s communication capability $C(n)$ can be defined as follows.

$$C(n) = \frac{E_{\max}(n)}{\Delta E} \qquad \text{Eq. 6-2}$$

Eq. 6-2 shows that the communication capability of node *n* depends on the maximum available energy $E_{\max}(n)$ and the energy required for each packet to be transmitted, forwarded or received; the selection from these three activities depends upon the attacker's choice for amount of life span it wants to achieve.

Let β be the fee node *n* has to pay, which is the number of packets forwarded for others as a fee payment. Furthermore, we suppose node *n* can use *I* identities simultaneously (acting as a Sybil attacker) or in sequence (acting as whitewasher). Then by incorporating β and *I* into Eq. 6-2, according to our scheme we get the following.

$$C(n) = \frac{E_{\max}(n)}{\Delta E} - (\beta \times I) \qquad \text{Eq. 6-3}$$

Eq. 6-3 shows that in our scheme (fee per identity), the communication capability of a node is also dependent on β and *I*, which can be explained by two points. First, if the amount of fee enforced is increased (β in the equation), then $C(n)$ will be reduced. This is due to the fact that the greater the number of packets forwarded as a fee, the lower the amount $C(n)$ will be leftover for node *n*'s own communications. Second, the greater the number of identities node *n* uses for Sybil attacks or whitewashing, the greater the number of times the fee must be paid; hence node *n* will have a lower amount $C(n)$ leftover for its own communication. This is how the fee enforcement will discourage nodes from using multiple identities.

The scheme discourages whitewashers and Sybil attackers alike, because the product $\beta \times I$ in Eq. 6-3 will be applied on both types of attackers alike. Furthermore, the product $\beta \times I$ will also improve the overall system performance (*i.e.* network throughput and utility), due to the fact that fee payment implies contributing to the network by forwarding other nodes' packets.

6.2.4 Node Interactions

Mobile ad hoc networks enable nodes to roam freely across the network and move from one neighbourhood to another. Nodes need other nodes in the network to forward their packets and this gives rise to situations in which nodes interact with other nodes that they already know and sometimes with strangers to whom they have not interacted before. To distinguish between strangers and previously known nodes, a node can check its own reputation table for the corresponding reputation entry that specifies the cooperation history of a node in the form of reputation values. If there is no entry found in the reputation table, the node will be considered a new node. Reputation tables are built from direct experiences with nodes and from secondhand information gathered from neighbours, as we discussed them in Section 4.3 and Section 4.4. We will describe using reputation tables how nodes should behave when they interact with other nodes. By checking its reputation table a node can categorize nodes that have paid their entry fee, we call them *mature* nodes, and nodes that have not yet paid their entry fee, called *new* nodes. As shown in Table 6.1, there are four different possibilities for node interactions; nodes in the columns are the sources of traffic while nodes in the rows are destinations for traffic. We describe the process as follows. Suppose node i starts communication with node j, *i.e.* $N_i \longrightarrow N_j$, where the arrow specifies the direction of traffic flow. The following are the four cases of interaction based on whether the nodes are mature or new. We also refer to each cell in Table 6-1 using the same letter (a to d).

Table 6-1: Node Interaction Matrix

	Receiver		
Sender	**Nodes**	**Mature**	**New**
	Mature	a) Cooperate	b) Cooperate
	New	c) Ignore	d) Ignore

a) Both nodes *i* and *j* are mature: they will provide services to each other.

b) Node *i* is mature and *j* is new: in this case if node *j* is not selfish, should forward *i*'s packets, in order to pay the fee.

c) Node *i* is new and node *j* is mature: node *j* will ignore node *i* provided that node *j* did not receive any secondhand information from its neighbours about node *i* (being paid the fee) or it did not overhear node *i* forwarding packets for others until its reputation becomes greater than *Y*.

d) Both *i* and *j* are new: they ignore each other's requests. An important point to note here is that when new nodes ignore each other will not cause the entire network to be in the deadlocked state because the fee enforcement will start in effect from the first detection, as we discussed the issue of bootstrapping in Section 6.2.1.

6.3 Simulation Based Analysis

6.3.1 Setup

We use Network Simulator NS-2.30 to implement and evaluate our scheme using the parameters listed in Table 6-2. In this simulation study our aim is to find out how the fee enforcement per identity affects the throughput and utility of good and evil nodes in MANET environments and to further assess whether it is beneficial. In either case we will compare the fee incorporated reputation-based scheme with the CONFIDANT scheme. Throughout our experimentation the percentage of selfish nodes is 10%, and these selfish nodes can use five identities in total. The value of reputation β, threshold Y, or the fee threshold, as shown in Figure 6-2, is set as 50. NS-2 assigns a sequential (starting from zero until the number of nodes) integer number for the identification of each node in the network, we use these identifiers as nodes' identities. All results are the average of 20 random scenarios or simulation runs.

Table 6-2: Simulation Parameters

Parameter	**Level**
Area	1000m × 1000m
Maximum Speed	10 m/s
Pause Time	0 to 600 seconds
Number of Nodes	30
Connections	25
MAC	802.11
Application	CBR
Packet Size	64 Bytes
Simulation Time	600s
Movement	Random Waypoint
Placement	Uniform
Selfish Node Population	10%
No. of IDs per Selfish Node	5

6.3.2 Mobility Model

We use a random waypoint movement model [94-95] for all of our scenarios in the simulations. We discussed this model in Section 5.6.2.

6.3.3 Attack Implementation

Using NS-2.30 we implemented two types of nodes: (i) nodes change identities one-by-one (whitewashers) and (ii) nodes change identities without discarding the previous one (Sybil attackers). We mentioned in Section 1.3.3, that our proposed methods thwart new identities, regardless of the notion of simultaneity. Since, in our simulations, only whitewashers are used; however, it is presented here to demonstrate that how our scripts can be used to create Sybil attackers in MANETs.

In order to create nodes having more than one identity, we bind n nodes together to represent one node having $n - 1$ identities. In other words, in the case of whitewashing, we bind 5 nodes together; node 1 is *up* while the rest of the 4 are ***shutdown***[7]. So when node 1 moves to a random location at random time, all the other 4 nodes are also moved to the same location following exactly the same dynamics. Each identity will work in rounds, for example one identity will be alive or up for a certain period of time while the rest of the identities will be down, after finishing its time it will be down and another identity will be up and so on.

In the case of whitewashing, each identity of the whitewasher will be up one at a time, as depicted in Figure 6-3. For example, Id_2 is up and will complete its round during time t_2. After t_2 expires, Id_2 will be put down and Id_3 will be up; if the connection to or from Id_2 is still active then it will be switched over to the next identity, *i.e.* Id_3. The last identity (Id_m) will stay up until the end of the simulation.

[7] In order to start up and shutdown a node, we use the node "on" and "off" options of the command() function of the mobile node class in NS2.

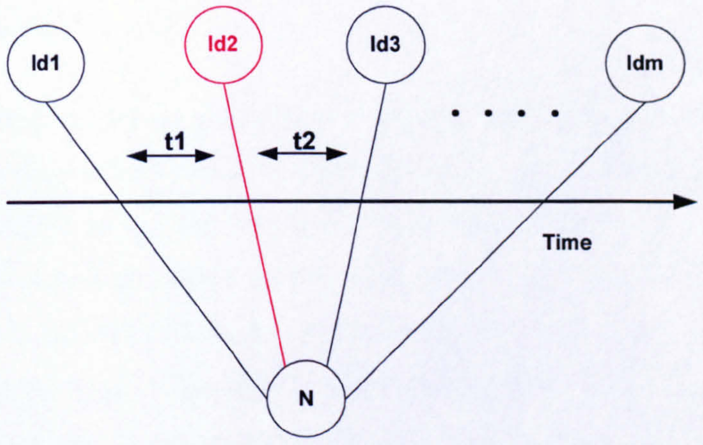

Figure 6-3: The whitewasher node process

In the case of Sybil attackers, each new identity will be added to the already up identities. For example, in Figure 6-4, after time t_1 expires, Id_2 is also added to the already up identity *i.e.* Id_1 and so on.

Throughout our experimentations, we used random scenarios (random mobility and traffic) using the commands provided in Appendix A. In Appendix A, we also provide the source code that has either been written from scratch or modified from existing code of NS-2 scripts for mobility and traffic generator.

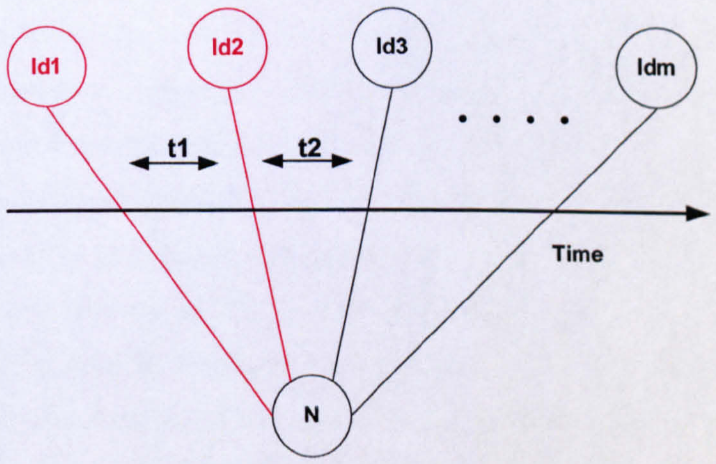

Figure 6-4: The Sybil node process

6.3.4 Attack Models

We classify the attacker nodes into two categories. However, the number of identities available for performing whitewashing is fixed. One class of evil nodes will misbehave (drop others' packets) *without* paying the fee. In the results we refer to this as Attack Model I or class-I attackers. In a realistic scenario, this type of evil node may be the result of normal users who have no expert knowledge of the field, but who have nonetheless installed misbehaviour software (*e.g.* claiming to extend battery life) programmed by others. We assume that they do not know the fee threshold and how to cross it. The second class of nodes are those evil nodes that commit misbehaviour and they *do* pay the fee as well. In our results we refer to this as Attack Model II or class-II attackers. In this case we are interested in the effect on throughput and utility of the network if, for example, some evil nodes pay the fee and then start committing misbehaviour.

6.3.5 Metrics

We evaluated our scheme using the following metrics, *i.e.* good throughput, evil throughput, good utility and evil utility; we discussed these metrics in Section 5.6.3.

6.3.6 Analysis

If there is no restriction imposed on identities in a network where users can acquire an unlimited number of new identities at zero cost, nodes performing whitewashing can get pretty good benefits from the network. As shown in Figure 6-5, the evil throughput of CONFIDANT is significantly higher than in our scheme because there are no restrictions imposed on newcomers and hence evil nodes can have multi-fold benefits in terms of throughput and utility. The greater the number of identities used, the greater the exploitation of network resources thereby increasing the benefits for whitewashers. Fee enforcement reduces the overall evil throughput (by about half) in the network. The throughput of the class-II attackers is marginally higher than that of the class-I attackers. This is due to the fact that class-II attackers pay the fee: they forward packets for other evil nodes; consequently, evil nodes obtain slightly higher throughput on average.

The good throughput of CONFIDANT on the other hand is slightly higher than our scheme in the scenarios where there is moderate mobility, as depicted in Figure 6-6. It is due to the fact that two new nodes ignore each other in our scheme which is likely to occur after the first detection of whitewashers.

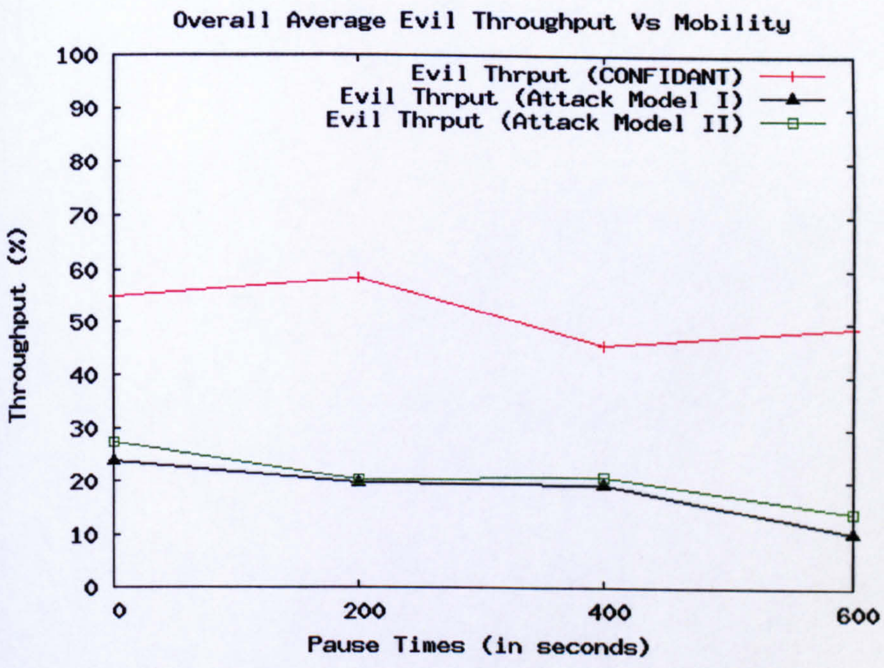

Figure 6-5: Overall average evil throughput *vs.* mobility

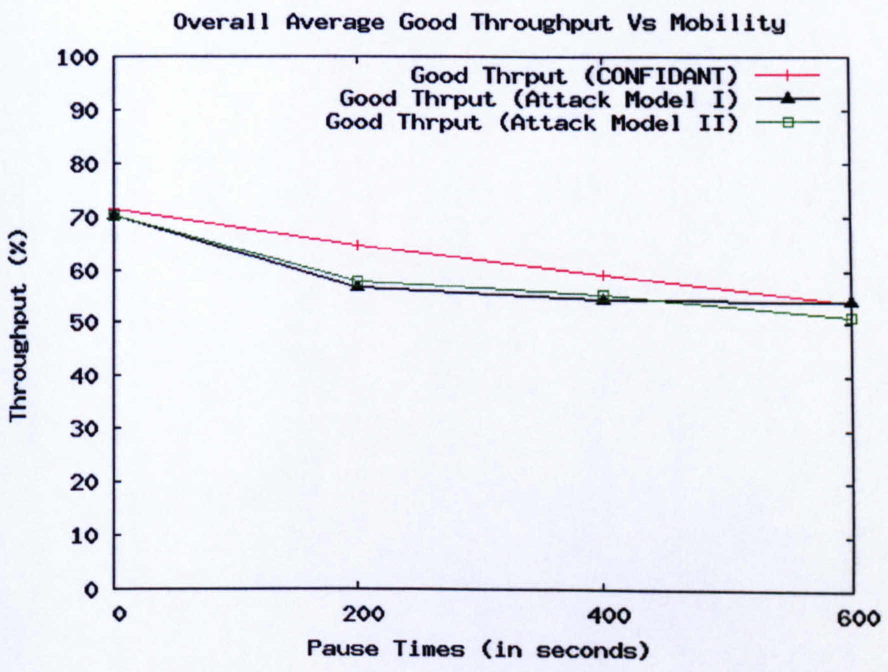

Figure 6-6: Overall average good throughput *vs.* mobility

The average utility of an evil node is considerably higher in CONFIDANT as compared to our scheme, as shown in Figure 6-7. This is due to the fact that in CONFIDANT evil nodes drop 100% of packets (no forwarding); however, they can still continuously enjoy the neutral reputation before their detection. Class-II nodes pay their fee *i.e.* their forwarded packet count is greater than that of the class-I attackers and hence their utility is lower than that of the class-I nodes.

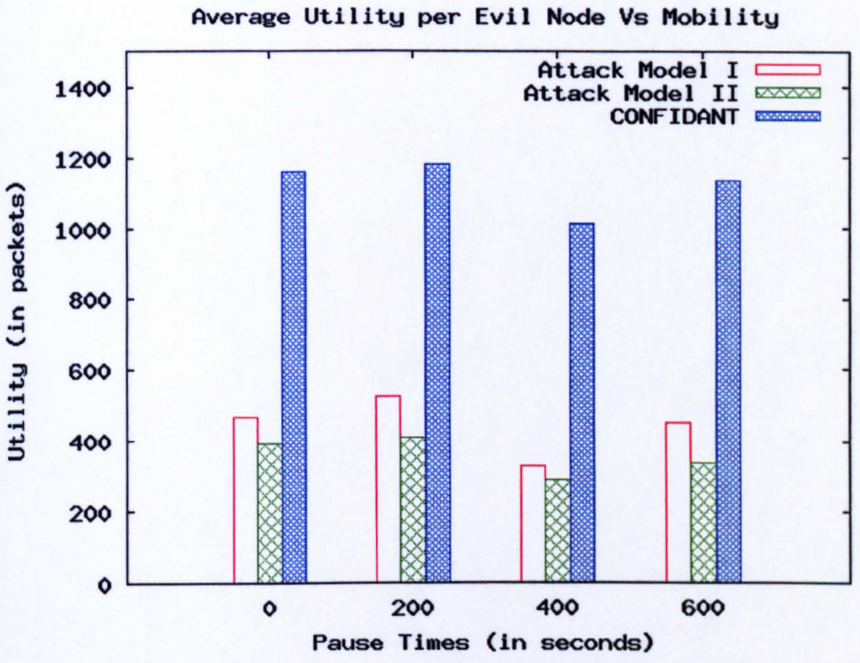

Figure 6-7: Average utility per evil node *vs.* mobility

Good nodes suffer considerably more in CONFIDANT. The more benefit the evil nodes get, the more the good nodes suffer. Because good nodes forward evil nodes' packets before detection, hence good nodes have low utility. As shown in Figure 6-8, good nodes in CONFIDANT enjoy considerably less utility than in our scheme (note that the y-axis represents negative values in Figure 6-8) as their packets rarely reach their destinations due to the selfish nodes in the network. Class-II nodes pay their fee in the form of forwarding packets for other nodes; hence in the presence of class-II attackers, the good nodes comparatively suffer less than in the scenarios where class-I nodes exist.

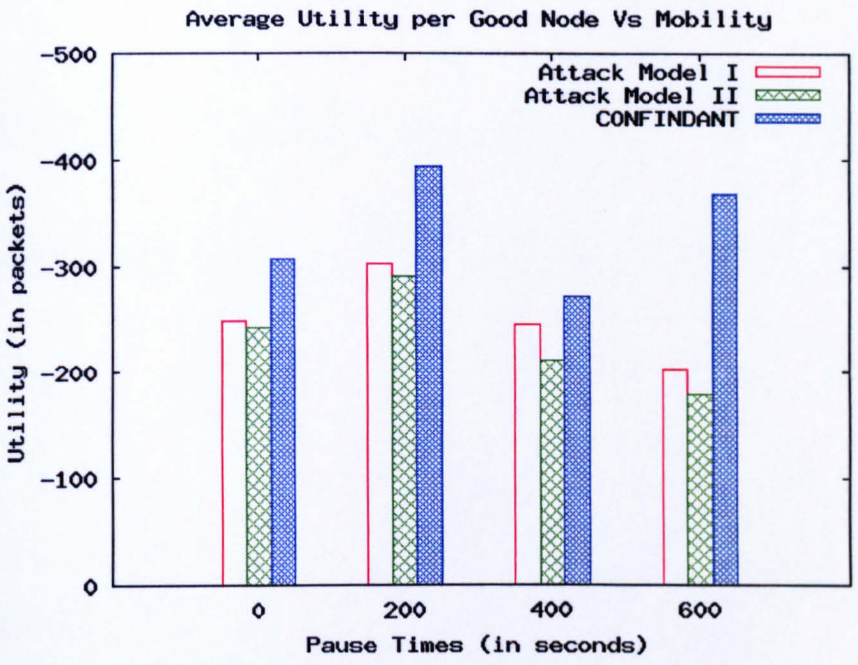

Figure 6-8: Average utility per good node *vs.* mobility

6.4 Summary

In this chapter, we discussed the potential issue of whitewashing and Sybil attacks in reputation-based schemes that, if not addressed, can make them impractical. We discussed our proposed scheme using an entry fee per identity to discourage normal whitewashers as well as Sybil attackers without using any costly method, for example, PKI or a centralized trusted third party. The simulation results show that our scheme performs well in reducing evil nodes' benefits as compared to the CONFIDANT scheme in the presence of whitewashing nodes. In addition to improving the overall system performance, our proposed fee enforcement scheme discourages whitewashers and Sybil attackers and makes new identity creation costly (in terms of battery power) for an attacker and hence reduces the number of whitewashes in the network. The downside of our approach is that newcomers are not welcomed *i.e.* they are not given services immediately. Finally, our scheme relies on secondhand information about reputation and fees. Safeguards are therefore needed to ensure nodes do not lie about these values (or are sanctioned if discovered), and we also aim to consider this in our future work.

Chapter 7 Whitewashing Detection in MANETs: A Reactive Approach

7.1 Background

In the previous chapter, we discussed how to deter whitewashers and Sybil attackers; however, an opportunity still exits for attackers to launch an attack, for example after paying the fee. This can occur if the benefit an attacker can gain is greater than the cost incurred by the fee enforced. This would represent an ample motivation for such attackers to launch an attack. In short, a second line-of-defence is necessary in order to counteract such attacks by detecting them and further isolating them, if they are not deterred by the proactive technique.

In this chapter, we will present our scheme, that detects whitewashers' and Sybil attackers' newly created identities. However, for tests and experimentation purposes we deployed only whitewashers in our network scenarios. The scheme can be applied to both cases alike, whether the new identities are created either one after the other or simultaneously make no difference to the detection process. Our detection scheme can work as a standalone scheme, but could equally be deployed as an add-on to existing schemes, for example it could be incorporated into a reputation-based system. Unlike other related work [24, 96], our proposed scheme does not need any directional antennae or any GPS (Geographical Positioning System) equipment [84, 97].

The rest of the chapter is organized as follows. In Section 7.2, we will explain our scheme with the help of some experiments along with an overview of the IEEE 802.11 protocol. In Section 7.3, we describe how we conducted the scenarios (from Section 7.2) as real-world testbed experiments using Java Sun Spot sensors [98] in order to confirm our rationale. In Section 7.4 we discuss the issue of tuning the detection threshold for worst case scenarios. Since our scheme can act as a stand alone, however it can be integrated into reputation-based schemes and we will discuss in Section 7.5 how it works with the reputation system. In Section 7.6, we evaluate our scheme, through extensive simulations and analyse the results. The chapter is concluded in Section 7.7.

7.2 Detection of Whitewashing Identities

7.2.1 Overview of IEEE 802.11 Protocols

The IEEE 802.11 protocol is the most widely adopted wireless networking protocol used in real applications [99-100]. Basically, IEEE 802.11 is the standard for medium access control (MAC) in wireless LANS and its main job is to provide carrier sensing multiple access with collision avoidance (CSMA/CA). The standard specifies two medium access control mechanisms, *viz.* distributed coordination function (DCF) and point coordination function (PCF). The PCF has been developed to use in infrastructure network configuration, whereas the DCF applies to both infrastructure as well as ad hoc mode. Since the DCF is based on CSMA/CA which is used in ad hoc network configurations, in this chapter we will confine our focus only to this mechanism.

The main purpose of the DCF protocol is to maximize throughput while preventing packet collisions by using carrier sensing with a 4-way handshake, as shown in Figure 7-1. Collisions occur when a node receives more than one packet at the same time; as a result neither packet is correctly received.

The basic mechanism for 802.11 is as follows. Consider a node has data to send to a destination node in the network. It waits for a random backoff time. This is a random deferral time before transmission is started and its value is taken from the interval between zero and the size of a contention window (which is the maximum amount of time a node can wait, will be explained later). If the node senses, at any time, that another node is using the channel, it pauses its timer until the other node completes its transmission. After the backoff time expires, the node will sense the channel in order to determine whether it is idle. After establishing that the channel is clear, it waits for a short period of time, called DIFS (DCF interframe spacing) and senses the channel again. If the channel is still free, the node will transmit a Request-To-Send (RTS) control frame to the destination. If the destination is free to receive data, it will respond with a Clear-To-Send (CTS) control frame. A Network Allocation Vector (NAV) is also transmitted along with both the RTS and CTS frames. We will explain the purpose of the NAV shortly. After the data is successfully received by the destination, the destination will transmit an Acknowledgement (ACK) frame back to the sender. If the sender still has more data to send, it would initiate its backoff time again while repeating the rest of the process.

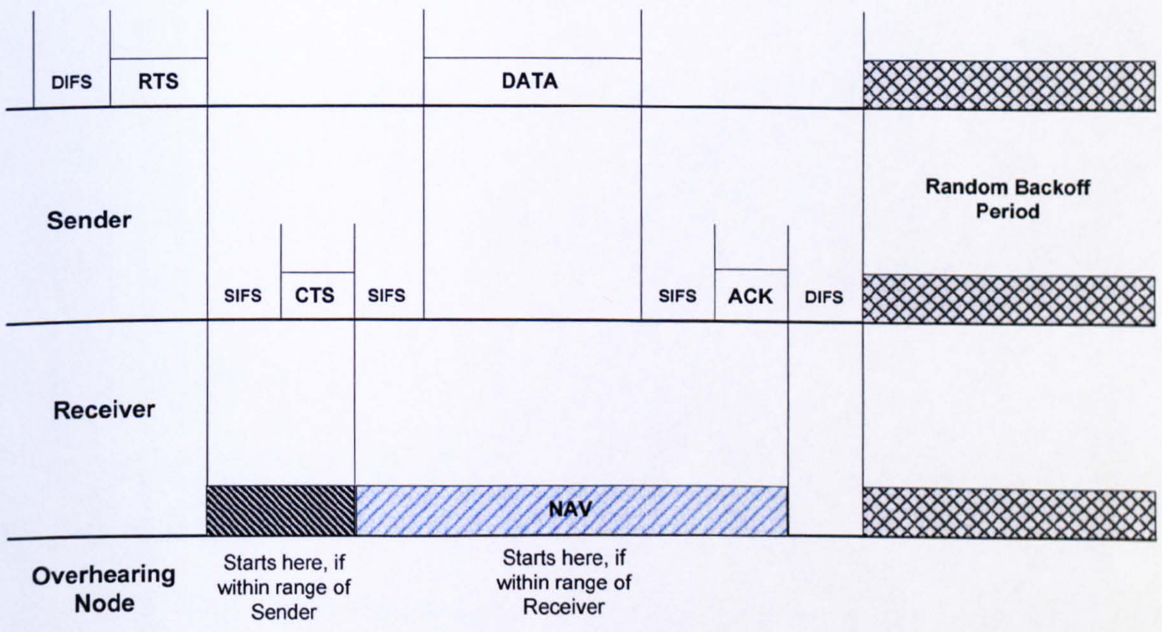

Figure 7-1: Functionality of CSMA/CA

The main purpose of the NAV is to inform other nodes about the length of the channel reservation time. Any node that overhears the NAV will sense the channel after the NAV expires. Since persistent (useless) sensing of the channel is one of the biggest uses of energy, the NAV diminishes the amount of this idle sensing; hence energy is saved across all nodes in the network.

The contention window is used to promote fairness in the network, so that no node monopolizes the channel for a long period of time. The size of the contention window usually varies based on the condition of the network. A window size that's too small implies that the chance of two nodes transmitting simultaneously will be increased. Whereas a window size that's too large means that the nodes might be idly waiting (unnecessarily) for a long time before their transmissions can proceed.

IEEE 802.11 defines interframe spacing periods of time between frames, in order to ensure that the channel is truly freed (idle). In our case, DIFS (DCF interframe spacing) and SIFS (short interframe spacing) are used. During DIFS, nodes sense the channel and start the frame exchange, if the channel has been sensed as free and their backoff times have also expired. In other words, when a node senses the channel, it must be free for the length of the DIFS period before the transmission starts. The SIFS is the time period that a node will wait between the RTS, CTS, DATA and ACK frames in the 4-way handshake. SIFS has shorter duration than that of the DIFS, which ensures that another node does not incorrectly determine that the channel is idle during the 4-way handshake; hence priority is given to the transmission in progress[8].

7.2.2 Assumptions

We assume that all nodes transmit with a constant power, *i.e.* nobody should increase or decrease their transmit power.

Our scheme is lightweight and does not generate any extra overhead caused by the periodic broadcasting of beacons or other control frames. Nodes will only capture the signal strength values of the transmissions occurring in their neighbourhoods. We do not require any form of node localization, hence no GPS is needed. Moreover there is no need for any centralized identity management or any extra hardware, such as directional antennas.

[8] Other MAC 802.11 issues, such as hidden and exposed terminals, fall out of the scope of this thesis and are therefore not discussed here.

7.2.3 Signal Strength Based Analysis

The distinction between a new legitimate node and a whitewasher's new identity can be made based on the distance from the receiver. That is new nodes become neighbours as soon as they enter inside the radio range of other nodes; hence their signal strength at the receiver node is recorded in a gradually increasing fashion when they move closer. In contrast a whitewasher, which is already a neighbour, will cause its new identity to appear abruptly in the neighbourhood. When it changes identity the signal strength of that identity will be high enough to be distinguished from the newly joined neighbour.

Our experiments show that a signal strength based threshold exists which can help us detect whitewasher identities. Our simulation results further demonstrate that when this threshold is properly tuned based on speed; it produces about 90% true positives with about 10% false positives.

Each node maintains a list of neighbours in the form <Address, Rss-List <time, rss>>, as shown in Table 7-1, and records the RSS (received signal strength) values of any directly received or overheard frames *i.e.* RTS, CTS, DATA and ACK messages. In other words, each node will capture and store the signal strength of the transmissions received from its neighbouring nodes. This can be performed when a node either takes part in the communication directly with other nodes acting as a source or a destination or when a node does not take part in the direct communication. In the latter case it will capture the signal strength values of other communicating parties through overhearing the control frames. Each Rss-List in front of the corresponding address contains R_n RSS values of recently received frames along with their time of reception, T_n. Where n is the number of elements in the Rss-List that can be increased or decreased depending upon the memory requirements of a node. In our simulation, we used n to be five elements; however, for real-world scenarios, it should be 10 to 20 elements because of the time varying nature of RSS.

In the following experiment, we plot the RSS of nodes in order to determine and visualize the behaviour of the new legitimate nodes and the whitewashers.

Table 7-1: Neighbour list based on received signal strength

Node ID	Rss-List
1	$\boxed{R1 \mid T1} \rightarrow \boxed{R2 \mid T2} \rightarrow \boxed{R3 \mid T3} \dashrightarrow \boxed{Rn \mid Tn}$
2	
3	
	• • •
N	

Experiment 1

This experiment is designed to allow us to compare the behaviour of new legitimate nodes with whitewash identities. As shown in Figure 7-2(a), when a new node *B* enters into another node *A*'s neighbourhood or radio range, node *B* gradually enters over time. This is the natural behaviour of nodes entering into one another's radio ranges and becoming neighbours in mobile environments. Due to this natural behaviour of entrance and exit, when node *A* stays static and node *B* entering into *A*'s radio range with speed *s*, node *A* will observe its received signal strength continuously increasing. When *A* plots *B*'s received signal strength readings as *B* moves towards *A* and then ultimately goes out of range on the other side; assuming that *B* is continually communicating with another node *C* or *A*. In graphical form, the RSS of *B* will produce more or less a *complete* elliptic curve, as shown in Figure 7-3. The diagram shows RSS plots for several arbitrary nodes in a random mobile scenario (this is taken from our simulation work presented in Section 7.6). The interesting characteristic of these plots is that the curve for each legitimate new node starts from the smallest readable RSS value (in an ideal situation), in this case it is GID (good identity) 17 indicating that GID 17 entered into 36 node's radio range *normally*. Whereas whitewash identities start from higher RSS values, such as WID (whitewash identity) 6 and 8 indicating that WID 6 and WID 8 did not enter normally into radio ranges of node 36 and node 21 respectively. So it can be deduced that these identities are the whitewashed one and their previous identities were roaming deep inside the radio ranges of the receivers, *i.e.* 36 and 21. This smallest readable RSS value could be used as a detection threshold; however mobility and velocity make things more complicated. For example, the common questions which may arise are, when *A* will receive *B*'s first signal strength value and at that particular moment, what would be the location of *B* inside *A*'s radio range? The answer to both of these questions is that it depends on the speed and transmission rate of node *B*. Nodes with lower transmission rates can penetrate more into the radio ranges before their presence being acknowledged. In other words, the greater the transmission rate of the nodes the sooner (and close to the boundary of radio range) their presence will be acknowledged and vice versa. The greater the speed of *B*, the further it will penetrate into the radio range of *A* before *A* acknowledges the presence of *B*. In order to refine our detection threshold, we conducted further experiments for speed. We do not conduct experiment for transmission rate because our aim here is to demonstrate, at what distance node *B* first acknowledged. This will potentially be affected by speed and transmission rate. We can get our aim by using speed and keeping transmission rate constant (or vice versa).

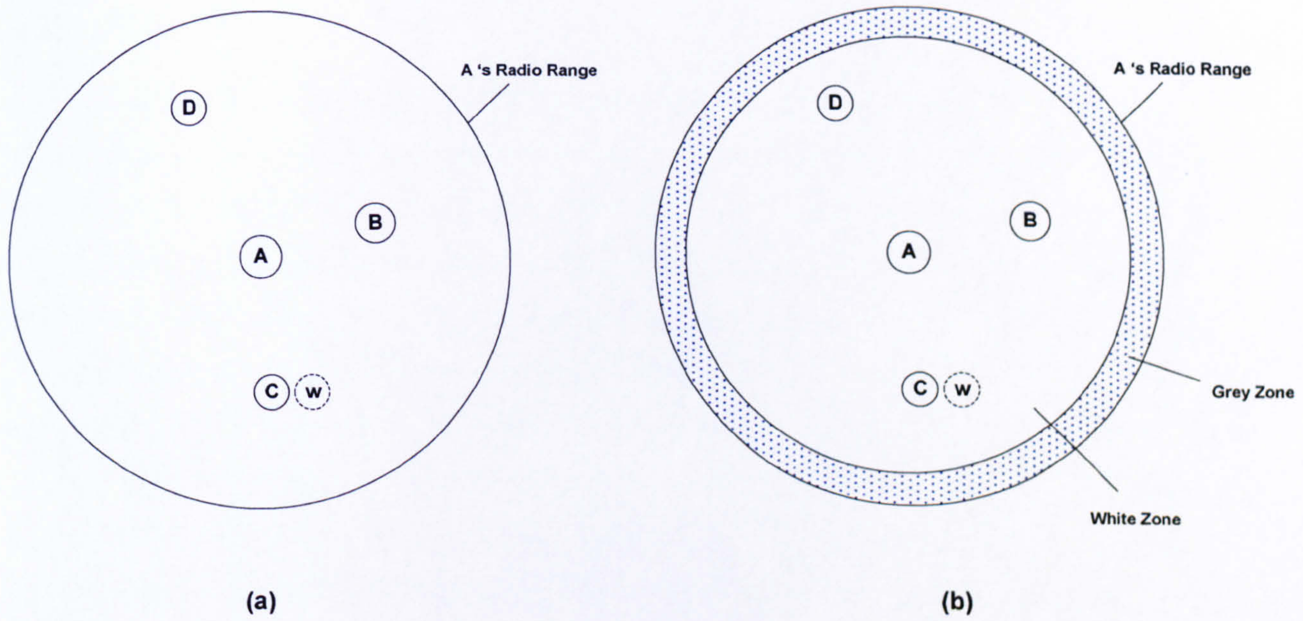

Figure 7-2: (a) Without and (b) with categorization of radio range

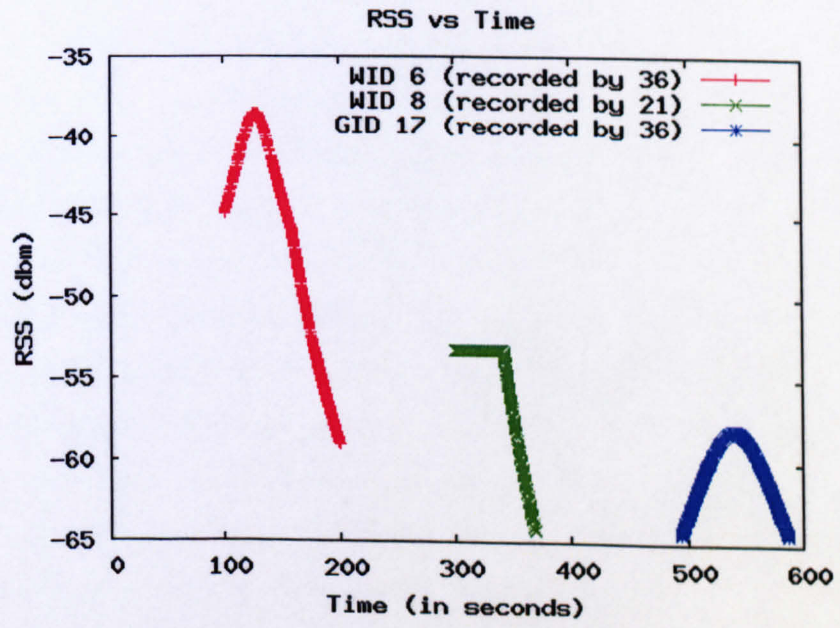

Figure 7-3: Plots of three arbitrary nodes' RSS values

Experiment 2

We conducted this experiment in order to establish how far node *B* penetrates into node *A*'s radio range before *A* acknowledges *B*'s presence. For this purpose we simulate the same scenario as shown in Figure 7-2(a) using NS-2.30. First *A* establishes a connection with *B*, where both the nodes are static. Then *B* starts moving in the opposite direction at a speed of 2 m/s speed until it goes out of range. After taking a pause, node *B* starts moving towards its original location with four different speeds one by one *i.e.* 2, 4, 10 and 15 m/s. The resulting RSS values received at node *A* can be seen in Figure 7-4. It is evident from the graphs that the greater the incoming speed of *B*, the greater the first RSS value of *B* that will be received by *A*. In other words, as the incoming speed of *B* increases, it penetrates further into *A*'s radio range before *A* acknowledges its presence. Hence, the first presence or RSS signal varies with speed.

We will set our detection threshold based on the maximum speed of the network; assuming that no node can move faster than this maximum speed. Now the question becomes, which speed should we adopt as the upper bound or our detection threshold, from Figure 7-4? To answer this question and for clarity purposes, we logically partition the radio range of node *A* into two zones: a grey zone and a white zone, as shown by Figure 7-2(b). Please note that this partitioning is based on the speed-based detection threshold. If we incorporate various speed-based thresholds from Figure 7-4 into Figure 7-2(b), it would become clear that higher speed thresholds produce wider grey zones. However, wider grey zones produce low true positive rates because wider grey zones increase the chance of nodes being able to commit whitewashing in this area. Whitewashing in this area cannot be detected, since the first appearance (or acknowledgment) of a node in the grey zone would usually represent a normal entry into the radio range of the node. In other words, if the threshold is based on 10m/s (*i.e.* where this is the maximum speed of the network), any node entering into *A*'s radio range, with a speed of 10m/s or below cannot produce its first appearance to node *A* inside the white zone of it (assuming no variance in RSS). So any new identity creation in the white zone will be detected as a whitewashing or Sybil identity, because normal nodes can't produce their first appearance in this area. From the above discussion, we can deduce that smaller speed-based thresholds will work better than larger ones because they will produce high true positives. Please note that we adopt a 10m/s threshold in our simulation based evaluation in

Section 7.6, and for this speed the simulation produced sound results. We believe that detection will be improved by using a lower speed threshold than 10m/s.

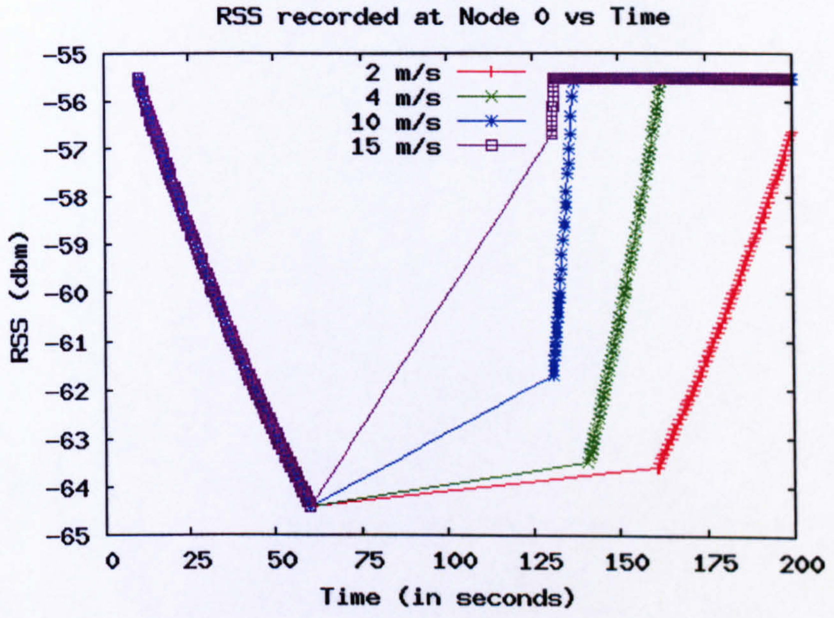

Figure 7-4: Determining node presence with respect to different speeds

There are two problems with this approach. The first problem is that if node *C* changes identity in the grey zone, node *C* will not be detected as a whitewasher. This effect can be reduced by collaboration with nearby nodes. For example, if *C* whitewashes in the grey zone of *A*, it may nonetheless be detected by *B* if *C* is in the white zone of *B*. This implies that a greater node density is likely to produce a higher true positive rate, our results also support this. The second problem is related to the low number of connections in the network. For example, suppose *C* is the destination-only node in the white zone of node *A* (node *A* is not aware of node *C*), and currently *C* is not receiving any traffic from its source node due to the connection being broken (due to mobility or other reasons) and that it has not re-established a connection yet. Node *A* will detect *C* (being a good node) as a whitewasher when node *C*'s previously broken connection is re-established, resulting in a false positive. The solution to this problem is that each node should transmit periodic beacon messages in order to indicate their presence; however this is costly in terms of communication overhead. A promising solution to mitigate this issue is to increase the number of connections in the network which will decrease these types of false positive. In other words, connections play an inverse relation to that of false positives, a fact we will demonstrate in our simulation results.

We have shown the natural behaviour of new entrants. Now node *A* can easily differentiate between a new node *B* that is coming into its neighbourhood and a whitewasher's new identity, pretending to be a new node joining the neighbourhood. This is done as follows. Node *A* will make a decision based on the RSS values of the nodes. If the first RSS value captured is greater than the threshold, *i.e.* a node is in the white zone, *A* will deem that identity as a new identity from a whitewasher, since no node can penetrate into white zone within the specified speed. If the first RSS value received is less than the threshold *i.e.* a node is in the grey zone, it will be considered as a normal new entrant and will be added to the neighbour list. The following are pseudo code for our scheme.

Algorithms

In order to detect new identities spawned by a whitewasher or Sybil attacker, Algorithm 7-1 checks every received RSS by passing it to the addNewRss function, along with its time of reception and the address of the transmitter. If the address is not in the RSS table, meaning that this node has not been interacted with before, *i.e.* it's a new node and the RSS received is its first acknowledged presence. This first received RSS is compared against an UB_THRESHOLD (this threshold is used to check from the RSS whether the transmitter is in white zone, *i.e.* whitewasher). If it is greater than or equal to the threshold, indicating that the new node lies near in the neighbourhood and did not enter normally into the neighbourhood; the address is added to the malicious node list. Otherwise, the address is added to the RSS table and a link list is created for that address in order to store the recently received RSS along with its time of reception in it. Finally, the size of the link list is checked, if it is greater than the LIST_SIZE, the oldest RSS is removed from the list.

Algorithm 7-1: Detection of New Malicious IDs

```
addNewRss (Address, rss, time_recv)
BEGIN SUB:
IF: Address is not in the Table
THEN:
        IF: rss >= UB_THRESHOLD
        THEN: Add Address to Malicious list
        ELSE: Add Address to the Table
END_IF
        Create a link list element for the Address
        Push_back the rss value and its time_recv to the link list
        IF: Size of the list > LIST_SIZE
        THEN: Pop_front
END SUB:
```

It is important to control the size of Table 7-1 otherwise it would grow invariably. In order to control its size, the unused records are needed to be deleted. These unused records are due to certain reasons. First, when a malicious node changes its identity, its previous identity record stays in the RSS table. Second, nodes join and leave the network at any time; hence the nodes that depart from the network leave behind a record of their RSS histories. In order to control the size, a global timer, called RSS_TIMEOUT shown in Algorithm 7-2, is maintained to flush the unnecessary records. When this timer expires, the rssTableCheck function is called, which checks the time of the *last* received RSS against the TIME_THRESHOLD for every address of the RSS table. If the time obtained is greater than this threshold, indicates that it is enough time past since it is not heard from this node. Now to check the reason of disappearance of nodes, the strength of the last RSS is checked against the UB_THRESHOLD, if it is greater, indicates that it is the previous identity of a whitewasher; otherwise it is concluded as a normal out of range scenario.

Algorithm 7-2: Table Refreshing: Detection of Previously Whitewash IDs

```
IF: RSS_TIMEOUT
THEN: Call rssTableCheck( )

            ==========================================

rssTableCheck( )
BEGIN SUB:
      FOR: for each Address in the Table
      DO:
         Pop the recent link list element
         IF: (Current_Time - getTime()) > TIME_THRESHOLD
          //Indicating that we did not hear from this Address since the
TIME_THRESHOLD
         THEN:
              IF: getRss() > UB_THRESHOLD
              Log ID to Malicious list: Previous ID of a Whitewasher
              ELSE: Normal out of Range
       END FOR:
END SUB:
```

7.3 Sun Spot Testbed

We conducted several real-world experiments using Java Sun Spot sensors [98], in order to confirm our results; since various authors, such as [101], pointed out that the received signal strength is unreliable. The aim of this testbed is to check the entrance and exit behaviour of a node from its RSS values. In addition, it is also important to check and see the dynamics of a node from its RSS values. For this purpose we conduct a testbed of Sun Microsystems newly developed Sun Spot sensors. There are two types of Sun Spots used in our experiments. One is the base station that is directly connected to a computer or a laptop via a wired link, in order to collect data from the transmitting sensors. The second is a free range Sun Spot, shown in Figure 7-5 that is used to transmit data to the base station wirelessly while being static or mobile. We configure these free range sensors to transmit packets to the base station after each 100 millisecond via a radiogram connection. The base station is configured to capture the RSS value of the received packets and log them into a file along with their time of reception.

In the next sub sections, we will show the experiments conducted in an indoor environment based on different mobility patterns, *i.e.* walking patterns and random waypoint like pattern. In the latter we used a Lego Robot for mobility as shown in Figure 7-6. Finally, we will summarise the observations made from the experiments.

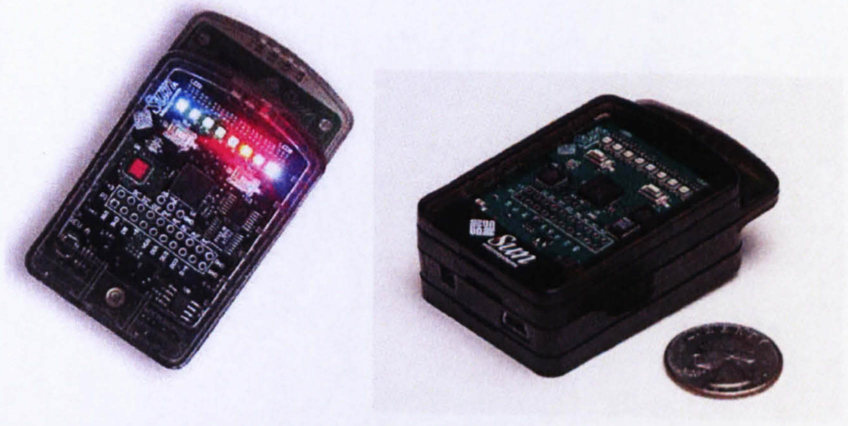

Figure 7-5: Free range Sun Spot sensors

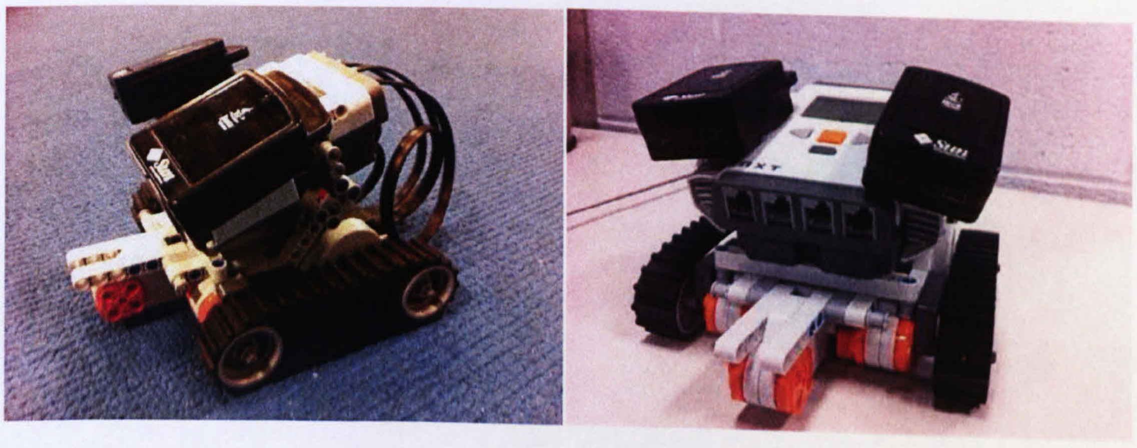

Figure 7-6: Sun Spot and Lego Robot in experiment

7.3.1 Walking Patterns

The purpose of this experiment is to check the behaviour of a sensor's motion as if it were a PDA or smart phone held by humans. We constructed the following scenarios. First, a sensor normally enters into the radio range of a base station, moves randomly with random pauses and finally goes out of range, shown in Figure 7-7. It is observed from the figure that RSS increases when the distance between the sensor and the base station decreases and vice versa. Second, the sensor node moves out of range from a location near the base station and then moves towards that location with the same speed, as shown in Figure 7-8, and with greater speed than the previous one, as shown in Figure 7-9. It is observed that while entering the radio range with higher speed, the base station acknowledges the first presence of the sensor node a little bit latter; hence a little bit higher RSS is received. The same fact has been shown in Figure 7-4. Third, a sensor node enters into the radio range and after some random movements and pauses changes its identity and with this new identity, it goes out of range. This is depicted in Figure 7-10.

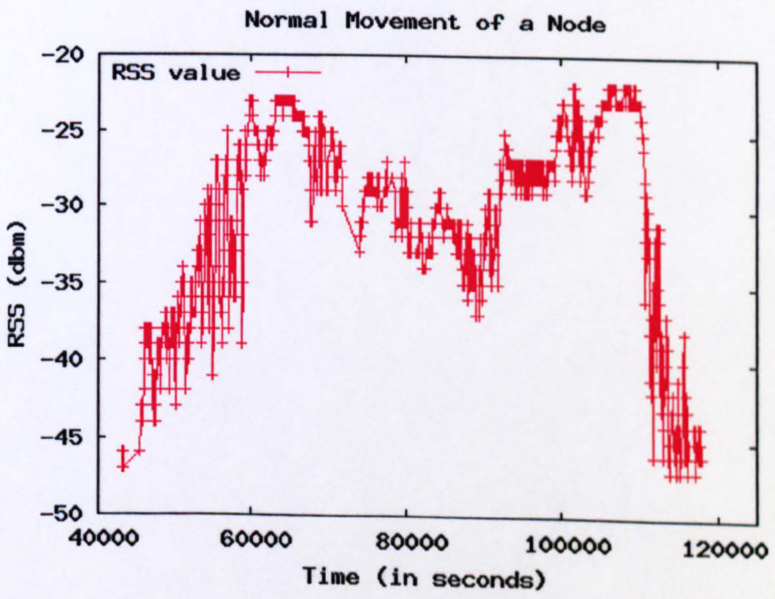

Figure 7-7: Normal movement of a node

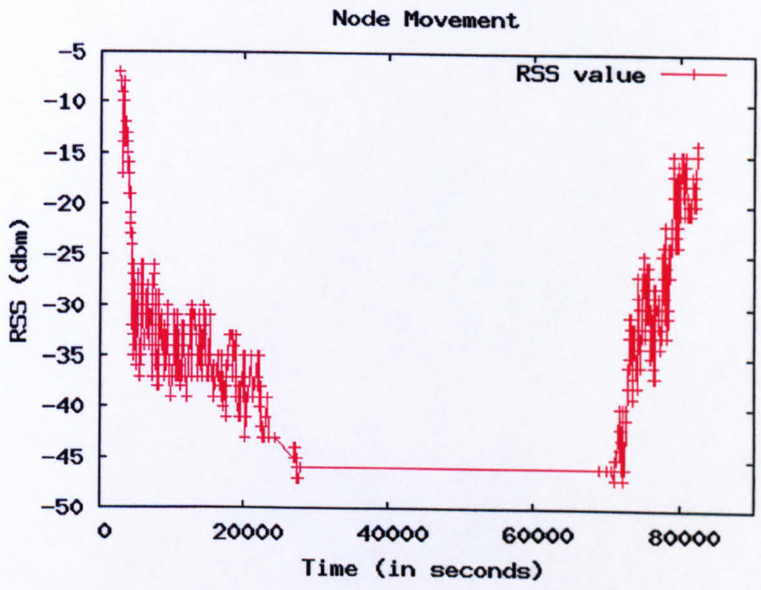

Figure 7-8: Node movement with same in/out speed

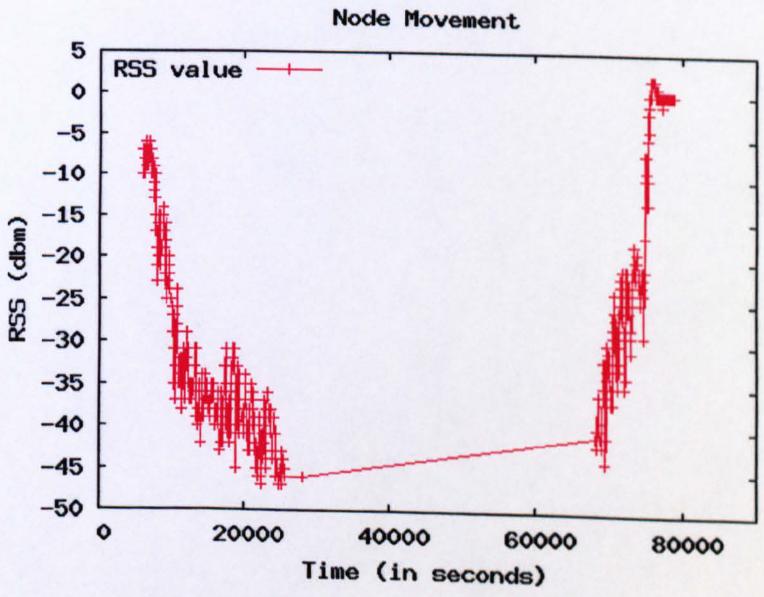

Figure 7-9: Node movement with different in/out speed

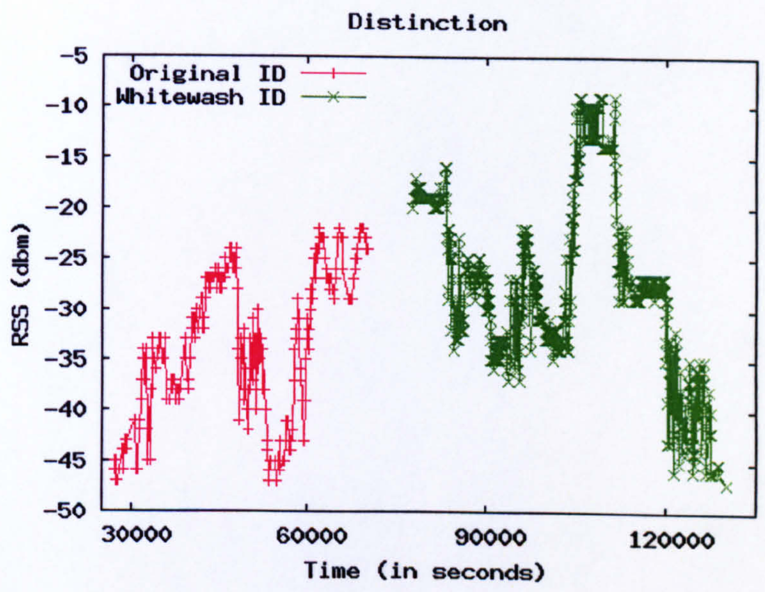

Figure 7-10: Distinction between original and whitewash identity

7.3.2 Random Waypoint like Pattern

We used a Lego Robot, as shown in Figure 7-6, in order to repeat the above four scenarios for random waypoint like patterns. The results of the experimentation are as follows. Figure 7-11 shows the RSS plot of normal entrance and exit behaviour of a node; Figure 7-12 and Figure 7-13 depicts the scenarios when a node uses two different speeds to enter the radio range of the base station. Finally, an identity changer node is depicted by Figure 7-14.

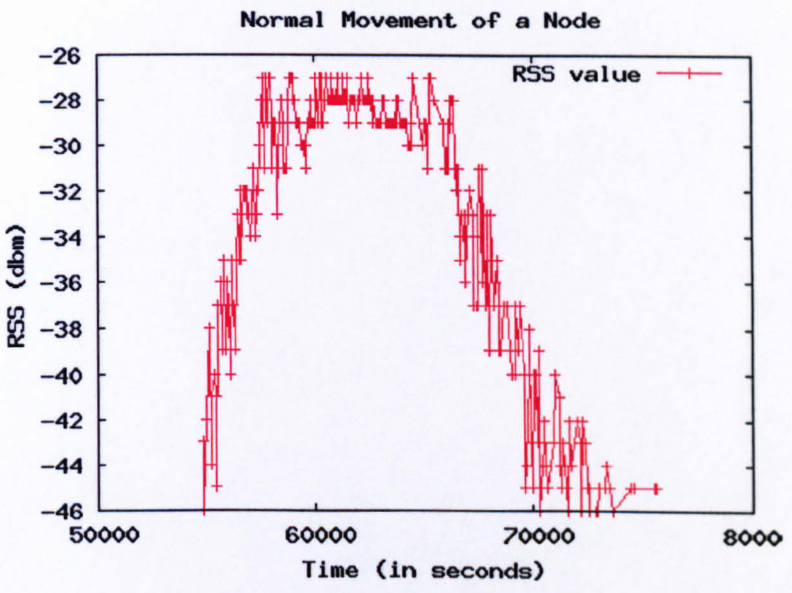

Figure 7-11: Using Robot, normal movement of a node

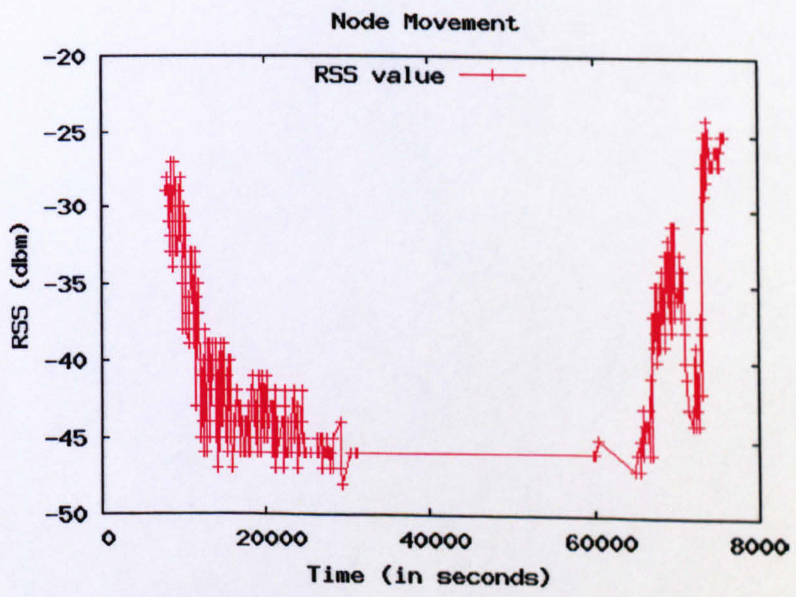

Figure 7-12: Using Robot, node movement with same in/out speed

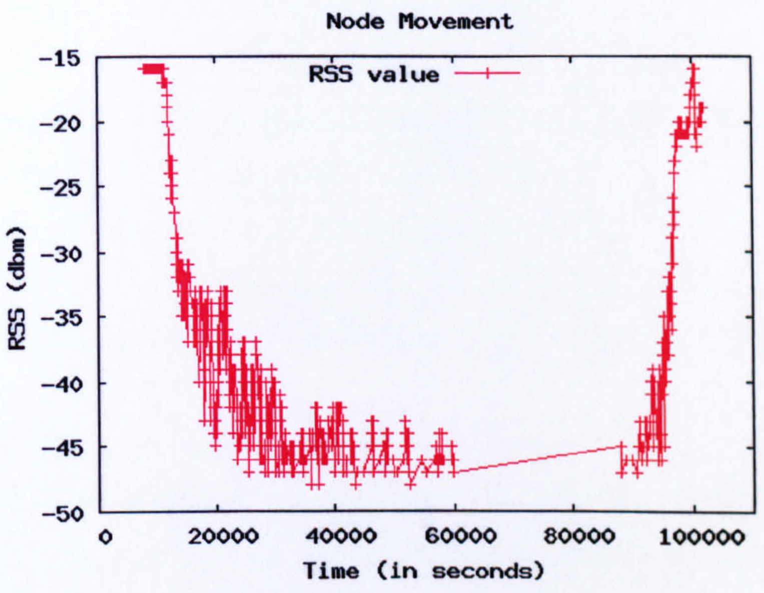

Figure 7-13: Using Robot, node movement with different in/out speed

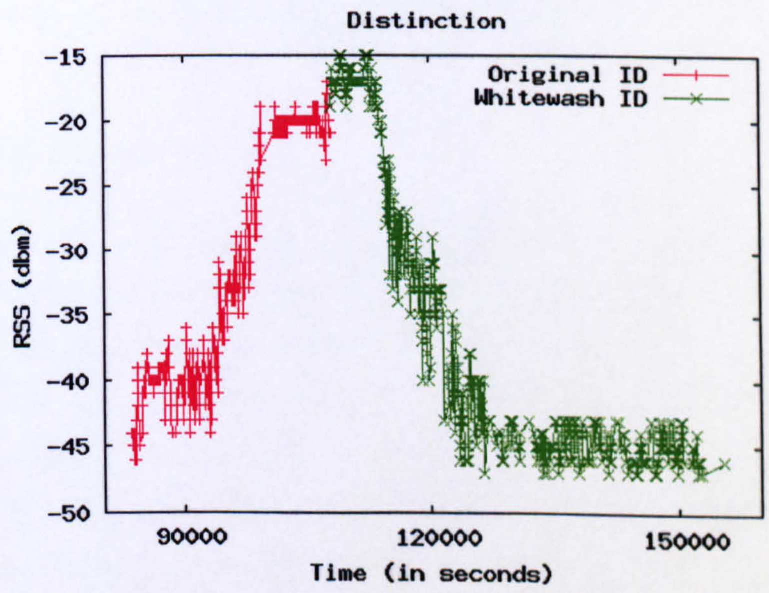

Figure 7-14: Using Robot, distinction between original and whitewash identity

7.3.3 Observations

The following are the main observations that we recorded about the above experiments.

- The radio range of the Sun Spot sensors in the indoor environment is 33 feet or 10 meters approximately.
- We are concerned about the entrance and exit behaviour of a node, so mostly the entrance and exit occurs in the range of -47 to -45 dbm.
- As we discussed above that the acknowledgment of the first presence depends on speed. When the sensor node moves into the radio range with higher speeds in our experiments, the first presence is acknowledged in the higher RSS range, *i.e.* from -45 to -40 dbm.
- If the identity change occurs near the boundary, such as the recorded RSS in the range of -45 to -40, it is not sure whether the received RSS is from a new legitimate identity (with higher speed) or an identity of a whitewasher; more specifically at -40 dbm.
- There is considerable amount of fluctuation in the RSS data.

In order to setup a detection threshold, the last three points are very important. In the next section we analyse these points to obtain a refined detection threshold.

7.4 Tuning the Threshold

The main difference we found between the results obtained from our simulations and from our testbed experiments is the variation in received signal strength values. As RSS varies, for a node B at a fixed distance d from a node A, the receiving node A can receive multiple different RSS values in the fluctuation range [$-v$, $+v$] (assuming $+v$ is greater than $-v$) and hence these values do not represent an *exact* indication of distance. If the detection threshold is based on a single RSS value, node A can receive, at any particular time RSS from B (while B is good node just outside the white zone) with $+v$ variance, shown in Figure 7-15 (a). As a result, A incorrectly detects B as a whitewasher, hence false positive (i.e. while the node might stay in grey zone). Another case can occur when node B is a whitewasher in the white zone near the boundary on its way out performs a whitewash, however, due to the variations in signal strength node A might receive RSS with $-v$ variation, as shown in Figure 7-15 (b), considering it a signal coming from the grey zone and hence will not detect it. In Appendix C, we determine the real fluctuation in the RSS data through real-world experiments. One way of mitigating the effect of this variation is to base our detection on an average RSS across *n* values (moving average), instead of basing our detection on a single RSS value.

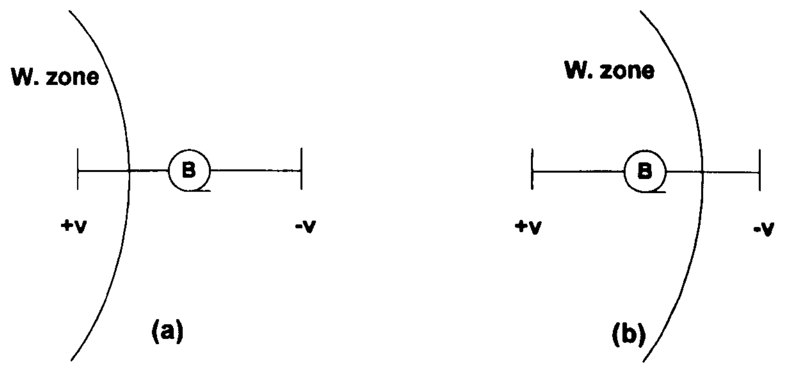

Figure 7-15: RSS with fluctuation

It is important to tune our detection threshold based on the speed and variation of the RSS values. This threshold logically partitions the radio range into white and grey zones: greater (or equal) signal strength than the threshold means white zone and grey zone otherwise. Let node *B* be approaching node *A*'s radio range with velocity *s* (ms^{-1}), assuming that d_b is the boundary of *A*'s radio range (in meters), Δt is the time (in seconds) between two packet transmissions and d_v is the inaccuracy in distance caused by *v* (in meters), where *v* is the variation in the RSS in the range of [-*v*, +*v*]. We assume for the sake of simplicity that just before the boundary, *B* transmits a packet which is not captured by *A*, so the worst case reading *A* will capture as a first acknowledgement or the white zone will be as the following.

$$Z_W = (d_b + d_v) + (s \times \Delta t) \qquad \text{Eq. 7-1}$$

The value of d_v and *s* will be negative when node *B* is moving towards node *A* and positive otherwise. From Appendix C we can know the value of d_b is 10 meters and *v* is 2.24 dbm which is the two standard deviation at position 3 (on the boundary), shown in Appendix C, Table C-3.

In summary, we can know from the calculation and discussion from Appendix C that for 95% confidence interval, the received signal strength can fall in the range [-45.16, -47.4], which is approximately two standard deviation or 2.24 dbm. So the distance produced from this variation in signal strength will be approximately 1.8 meters. Putting the above values in Eq. 7-1, we can calculate the threshold for the white and grey zones as the following.

$$Z_W = (10m - 1.8m) + (1\,ms^{-1} \times 0.1s)$$

$$Z_W = (8.2m) + (-\,0.1m)$$

$$Z_W = 8.1m$$

$$Z_G = (10m) - (8.1m)$$

$$Z_G = 1.9m$$

7.5 Response to the Reputation System

When a whitewasher's or Sybil attacker's new identity is detected, that identity is passed to the reputation system in order to record it in the detecting node's malicious node list. The detected identity will then be isolated from the network, *i.e.* other nodes will drop data and route request packets originating from the malicious identity, as we discussed in relation to the node isolation process in Section 4.6. With the passage of time when all nodes update their malicious lists by exchanging malicious list information with each other, the malicious identity will become completely isolated from the network. This also reduces the opportunity for whitewashers as well as Sybil attackers to launch their attacks.

7.6 Evaluation

7.6.1 Simulation Setup

In order to implement and evaluate our scheme, we use Network Simulator NS-2.30 using the parameters listed in Table 7-2. In this simulation study our aim is to establish the detection percentage of our proposed scheme in different scenarios. As we discussed above, two attributes of the network are mainly responsible for affecting the accuracy of detecting malicious nodes. These attributes are number of network connections, node density and transmission rate. In each of our scenario we take speed as our main attribute. All of the results we present here have been calculated as an average of 25 different random scenarios (or simulation runs).

In the next section, we will analyse our simulation results that are based on a variety of node speeds, connections and node densities.

Table 7-2: Simulation Parameters

Parameter	**Level**
Area	1000m × 1000m
Speed	2 to 16 m/s
Pause Time	10 s
Radio Propagation Model	Two-ray Ground Reflection
Radio Range	250m
Carrier Sense Range	550m
Number of Nodes	20 to 60
MAC	802.11
Application	CBR, 10 to 40
Packet Size	64 B
Simulation Time	600s
Movement	Random Waypoint Model
Placement	Uniform
Malicious Population	10%
Whitewash Ids per Malicious Node	5
RSS_TIMEOUT	100 seconds
TIME_THRESHOLD	30 seconds
UB_RSS_THRESHOLD	6.45×10^{-10} Watts
LIST_SIZE	5

7.6.2 Metrics

We use two main metrics in order to determine the detection accuracy of our scheme in different environments, *i.e.* True Positive Rate (TPR) and False Positive Rate (FPR). True positive means a malicious node is correctly detected and false positive means a good or legitimate node is incorrectly detected as a malicious one, as given below.

$$True\ Positive\ Rate = \frac{Correctly\ detected\ whitewash\ ids}{Total\ whitewash\ ids}$$

$$False\ Positive\ Rate = \frac{Incorrectly\ detected\ good\ ids}{Total\ good\ ids}$$

7.6.3 Analysis

As shown in Figure 7-16, data connections in the network are inversely proportional to the false positives of our scheme. For detection, movement sensing or the reception of frequent RSS values are important. In order to obtain RSS values from a node, that node should be involved in some form of communication, for example by acting as a source, forwarder, or destination. The more frequent a node sends or receives packets, the more efficiently a neighbouring node will detect it in the event that it tries to whitewash its identity. Fewer connections in a network imply fewer source and destination nodes, and greater difficulty for a node to distinguish other nodes' positions (*i.e.* their position as being either in a grey or white zone, or neither). Consequently a greater number of false positives will result. However, connections have no apparent effect on the true positives and for most of our experiments the true positives remained around the 90% level, as depicted in Figure 7-17.

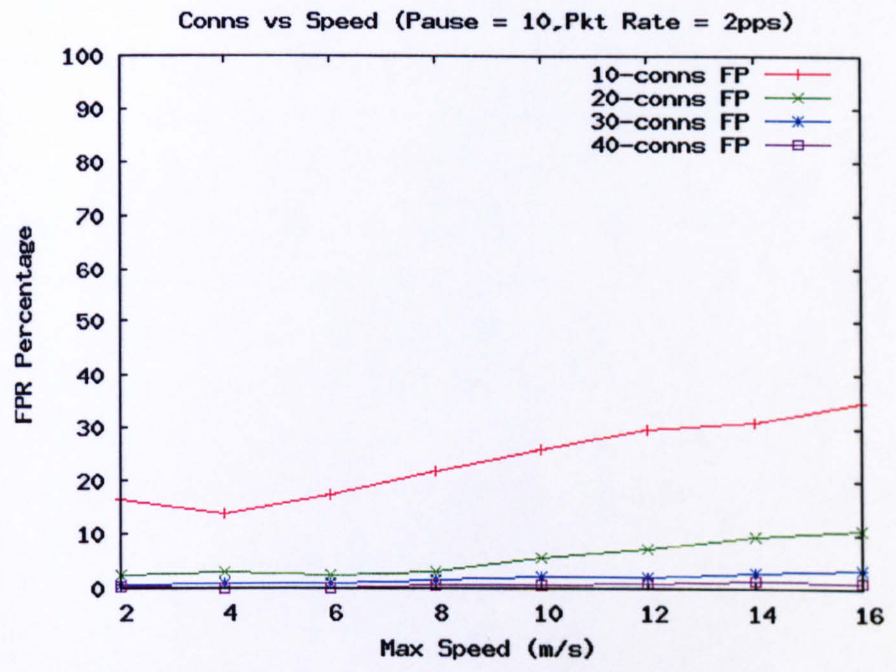

Figure 7-16: False positives with various speeds and connections

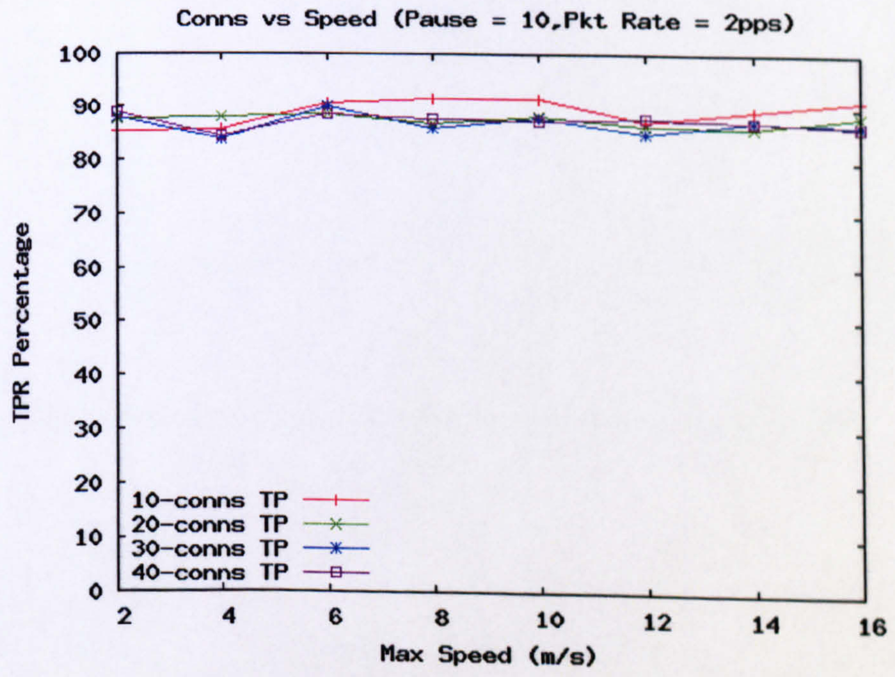

Figure 7-17: True positives with various speeds and connections

As shown in Figure 7-18 and Figure 7-19, density has comparatively little effect on the false positives as compared to the true positives. The increase in false positives in Figure 7-18 is due to the speed of the nodes. As we mentioned earlier we setup the threshold for the nodes assuming a maximum speed of 10m/s. For this reason we can see that the increase in false positives beyond 10m/s is significantly greater than the level below the 10m/s mark. In Figure 7-19, we can observe how a high density produces high true positives. The reason for this is that it is possible for a node to change identity in a grey zone, and hence not be detected. Increasing the node density would increase detection because if a node whitewashes in the grey zone of one node, it's unlikely to be not in the grey zone of all other neighbouring nodes in the network (meaning other nodes will detect it); hence the high true positive rate.

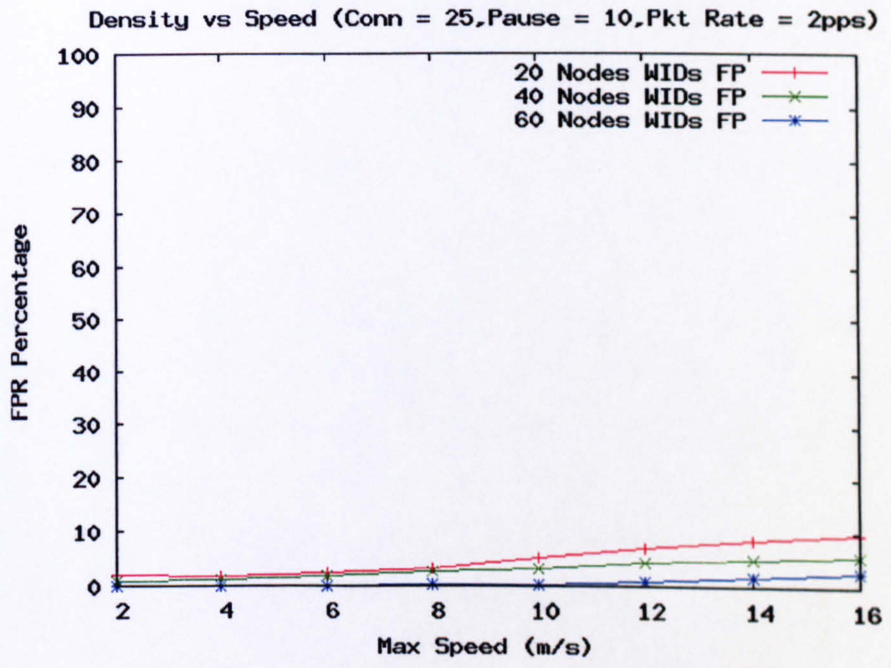

Figure 7-18: False positives with various speeds and densities

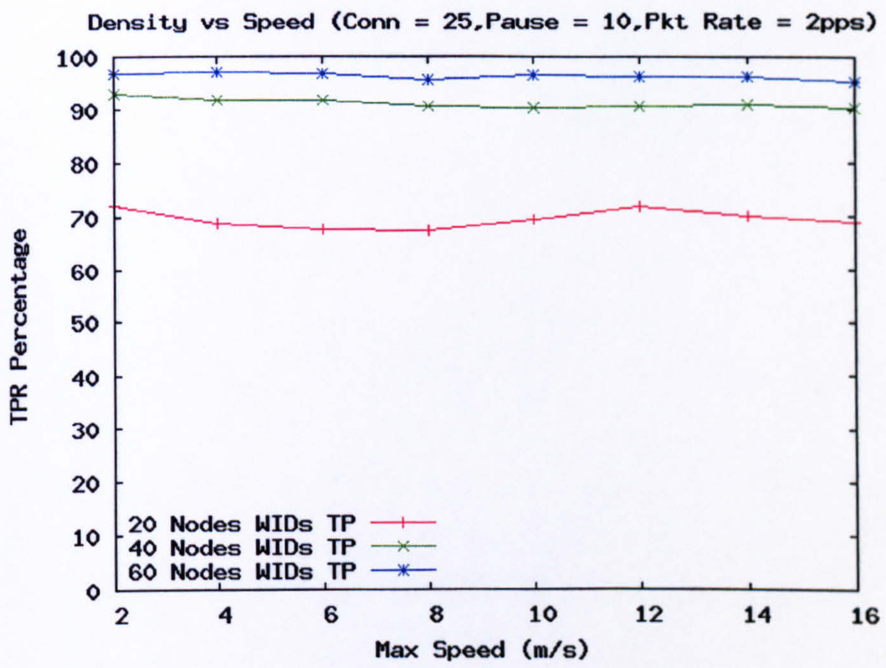

Figure 7-19: True positives with various speeds and densities

Low transmission rates produce false positives, especially when the speed is high, as shown by Figure 7-20. It is due to the fact that a legitimate node with low transmission rate and high speed can penetrate into the white zone of the receiver node. The receiver node will acknowledge its first appearance in the white zone and hence the legitimate node is detected as a whitewasher node. True positives are still about 90%, as shown in Figure 7-21; hence transmission rates do not have a big impact on true positives.

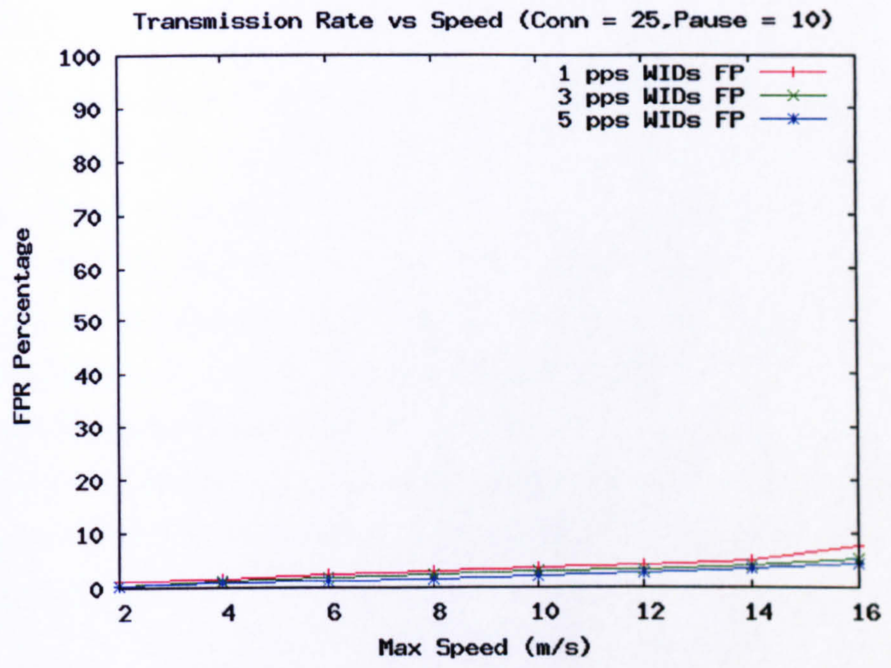

Figure 7-20: False positives with various speeds and transmission rates

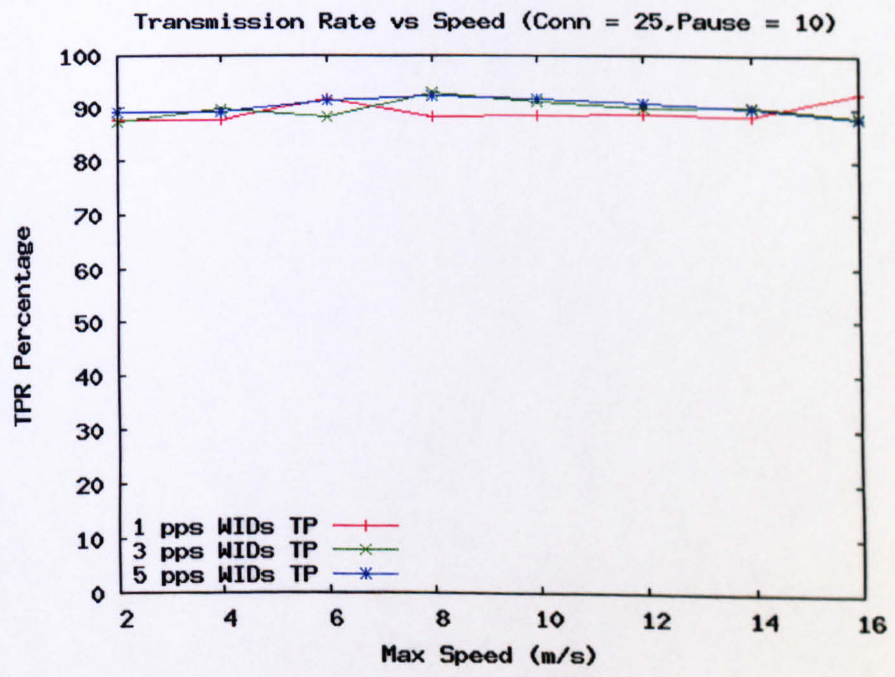

Figure 7-21: True positives with various speeds and transmission rates

7.7 Summary

In order to safeguard the network against whitewashing and Sybil attacks, a second line-of-defence is very important, even if a first line-of-defence already exists. In this chapter, we presented our detection mechanism for these attackers in mobile environments. We demonstrated through various experiments that a detection threshold exists for the distinction of legitimate new nodes and new malicious identities. We confirmed this distinction rationale through simulations as well as through the use of a real-world testbed of Sun Spot sensors. We also showed the various factors affecting detection, such as network connections, packet transmission rate, node density and node speed. The simulation results show that our scheme works better even in mobile environments and can detect whitewashers and Sybil attackers with a high degree of accuracy.

Chapter 8 Conclusions and Future Work

This thesis has presented security flaws in cooperation enforcement schemes, specifically reputation-based schemes. These issues are direct interactions and identity-based attacks. A number of novel countermeasures have been proposed in order to deter and detect these attacks. In this chapter, we provide summary and our research contributions of this thesis; we will also discuss the issues that need to be worked on in the future research work.

This chapter is organized as follows. Section 8.1 and 8.2 presents the overall thesis summary and the research contributions respectively. In Section 8.3 and 8.4, we outline the comparison with existing schemes and future work respectively. The chapter is concluded in Section 8.5.

8.1 Thesis Summary

Mobile ad hoc networks are constructed with the help of a large number of wireless nodes, such as laptops, tablets, PDAs and sensors, usually having limited transmission, battery and computation powers. The key advantage of these networks is their capability to work without relying on any centralized architecture or control. Each node provides packet forwarding services to other nodes in the network, since there are no dedicated routers present in the network for packet forwarding. However, due to a variety of reasons nodes may not always return such altruistic behaviour. One reason for this might be that nodes do not belong to a single authority. This is quite a common scenario within civilian applications. Nodes have different interests and objectives and they are therefore sharing their resources for the sake of global connectivity. Another reason might be that nodes belong to a single authority such as the military and have common goals; however they nonetheless operate in a physically insecure environment, where they are vulnerable to being captured or compromised. In any case, nodes show reluctance to spend their precious resources on forwarding other nodes' packets and therefore they may exhibit selfish behaviour. This selfishness can ultimately lead to network partitioning and performance degradation. Reputation-based schemes have been proposed to counteract selfish nodes in mobile ad hoc networks.

The main purpose of reputation-based schemes is to discourage selfish nodes by isolating them from the network once they have been detected. In other words, these schemes ensure that selfish nodes bear the consequences of their bad actions. However, selfish nodes can compromise this policy of the reputation-based systems by applying certain techniques. For example, they can still gain benefit from the network after being detected through direct interactions by exploiting mobility. Nodes can change identities to escape the consequences and whitewash their bad histories. They can also create and control more than one identity on a single physical device in order to self-promote themselves by publishing bogus positive recommendations. Malicious nodes can also use these Sybil attacks to defame good nodes.

In order to help secure reputation-based schemes and deal with the above mentioned issues, we have proposed solutions in order to counteract them in this thesis. As shown in Figure 8-1, we have proposed a layered defence in order to tackle whitewashers and Sybil attackers. Since a single layer or defence is usually easy for attackers to compromise, a double defence based on deterrence and detection would make the system more robust and efficient against the attacks. In the figure below, the detection mechanism detects whitewashers' and Sybil attackers' new identities and pass them on to the reputation-based system. The reputation-based system then isolates those identities from the network (as we discussed in Section 7.5). The direct interactions are used by the selfish nodes to gain benefit after being detected by the reputation-based system or without paying the fee in the fee-based scheme. We have proposed a scheme in order to thwart those attackers using a contribution to consumption ratio (C2CR) that works along with the fee-based mechanism.

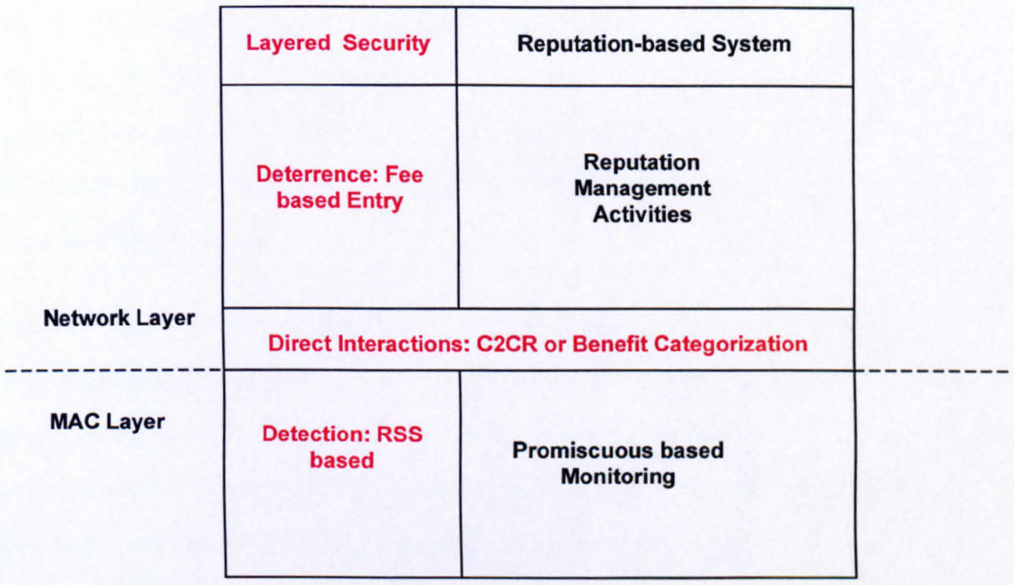
Layered Security
Reputation-based System
Deterrence: Fee based Entry
Reputation Management Activities
Network Layer
Direct Interactions: C2CR or Benefit Categorization
MAC Layer
Detection: RSS based
Promiscuous based Monitoring

Figure 8-1: Layered Security for Reputation-based System

A chapter wise summary of the thesis is given below.

Chapter 1 outlined the potential issue of selfishness in MANETs and the schemes proposed in the literature, such as reputation and trust based schemes, to counteract it. We highlighted issues in the reputation-based schemes, such as direct interactions, whitewashing and Sybil attacks. Finally, we outlined research aims and novel contributions of the thesis.

Chapter 2 presented a background and preliminary information about general network security and discussion about why the traditional network security techniques are not suitable for MANETs. In addition to packet relay problem or selfishness, we discussed routing modes of operation, the DSR protocol and the threats posed against the ad hoc routing protocols. Finally, we discussed the widely used network simulators for MANETs and discussed the reasons and requirements based on which we chose NS-2.

Chapter 3 presented a survey of cooperation enforcement schemes, such as reputation, trust and credit based schemes. We discussed the suitability of reputation-based schemes for MANETs. Finally, the limitations of these schemes in general were presented and the countermeasures proposed in the literature for the whitewashing and Sybil attacks were also surveyed.

In Chapter 4, we presented our proposed reputation-based scheme and discussed each of its components, *i.e.* the monitor, the reputation system and the path manager. The main purpose of this scheme was to use this for our layered security architecture presented in Chapter 6 and 7. We evaluated the scheme in order to confirm its working and validity which was shown through the simulation results.

Chapter 5 highlighted the problem of direct interactions from two different facets. They might be a selfish node's strategy to increase its benefits after the detection or they might be produced by inappropriate simulation parameters. In the former case, they are a threat to the security of reputation system and in the latter case they cause misjudgements and confusion in the results during the evaluation. We proposed methods to mitigate their affects for both of the above cases. We adopted a widely used reputation-based scheme as a case study for our simulation based evaluation.

Chapter 6 presented the first part of our proposed layered security architecture, *i.e.* pro-active deterrence. In this chapter we demonstrated where and why in the reputation range a whitewasher can obtain the benefits. We then proposed a novel non-monetary fee per identity based scheme in order to discourage these attackers by reducing their benefits. We evaluated and compared our scheme with a widely used reputation-based scheme. The simulation results have shown that our scheme works well in reducing the evil nodes' throughputs and utility in the network.

Chapter 7 outlined the second part of our proposed layered security architecture, *i.e.* the detection. We proposed our scheme that used received signal strength to detect the new identities of whitewashers or Sybil attackers. Through extensive simulations and real-world measurements using sensors testbed, we have shown that our proposed solution improves the overall system performance thereby detecting such malicious nodes with good accuracy even in the presence of mobility.

8.2 Research Contributions

In this section we present the research contributions of this thesis. The first one is related to the direct interactions which we covered in Chapter 5. While the last two is related to identity-based attacks, *i.e.* whitewashing and Sybil attacks which we covered in Chapter 6 and 7.

- We have proposed methods to cope with direct interactions in the network, whether they are natural or deliberate. We have established that commonly used simulation parameters produce a considerable amount of direct interaction as a by-product, which could affect the interpretation and evaluation of any results obtained. We demonstrated through simulations that this confusion can be mitigated by categorizing the network throughput and utility. We also proposed proactive measures for use in simulations to help reduces direct interactions. In case a selfish node decides to capitalise on direct interactions as part of a strategy, this can be tackled by promoting fairness in the network by employing service consumption to contribution ratio in the network.
- In order to tackle Sybil attacks and whitewashing, we proposed a layered security approach in which the attackers are thwarted in layers or line-of-defences, *i.e.* proactive measures (before the attack occurs) are used as first line-of-defence and reactive measures (after the attack occurs) are employed as a second line-of-defence. The former deters the attackers and the latter detects the attackers. As a proactive measure, we deter and discourage these attacks by proposing a non-monetary fee (in the form of node cooperation) incorporating a reputation-based scheme that acts as a first line-of-defence. This type of fee enforcement has several advantages. Unlike other fee-based approaches that have been previously proposed in the literature, our proposed fee structure is easily manageable in a MANET environment. It does not need any tamper-proof hardware and uses packet forwarding as a fee payment; hence no extra hardware needs to be incorporated into the system. We have shown through simulations and mathematical formulations that the scheme discourages these attackers by reducing their benefits and makes identity creation costly for these attackers.
- We proposed a scheme that detects the new identities of whitewasher or Sybil attackers on the MAC layer using the 802.11 protocol without the use of any extra overhead or extra hardware. This scheme acts as a second line-of-defence of our layered approach. After setting up the threshold, the scheme distinguishes between new legitimate nodes and new identities spawned by whitewashers or Sybil attackers.

We have shown through extensive simulations and real-world experimentations (using sensors testbed) that our proposed solutions detect these attackers with good accuracy even in mobile environments.

Our layered security approach use lightweight and distributed methods to deter (in the first place) and detect whitewashers and Sybil attackers without incurring any extra overhead or any hardware. As a result, these proposed methods help reducing and thwarting these attackers in various schemes developed for MANETs, especially reputation and trust based schemes.

8.3 Comparison with Existing Schemes

In this thesis, we have proposed novel schemes to counteract the problems of DIs, whitewashing and Sybil attacks. We compared our schemes with the existing benchmark schemes through simulation or by providing theoretical arguments.

In Chapter 6, we have proposed a non-monetary fee based scheme in order to discourage whitewashers and Sybil attackers. We compared our scheme with a widely used reputation-based scheme, CONFIDANT [25, 55], and the simulation results have shown that our scheme worked better than the above scheme in reducing the evil nodes' benefits, such as throughput and utility; while at the same time maintaining high throughput and utility for the good nodes as well. Our proposed scheme makes identity creation costly for these attackers and hence they can perform fewer whitewashes or generate fewer identities within the limits of their battery capacity, as compared to the above scheme.

In Chapter 7, we proposed a scheme to detect whitewashers and Sybil attackers' new identities. Unlike [24, 96], our scheme does not need any extra hardware, such as directional antennae and unlike [84, 97], our scheme does not use GPS (Geographical Positioning System) equipment or other costly localization techniques.

8.4 Future Work

A number of potential areas of further research will be discussed in this section.

8.4.1 Securing Fee Information Dissemination

Our work related to non-monetary fee enforcement, discussed in Chapter 6, raises a number of additional challenges. Nodes disseminate fee related information with their neighbours; however, a malicious node may generate its own fake fee information in collusion with other malicious nodes in the network, *i.e.* a malicious node can lie about another malicious node as being mature without paying the fee. In order to mitigate the effect of nodes' lies about fee payments, we propose a voting based scheme. We assume that the number of good nodes is greater than the malicious liar nodes in the network and only mature nodes' information will be considered. That is, when N neighbours of a node share fee information about it, m malicious nodes out of N lied about the fee, where $N > 2\ m$. Each receiving node of these fee based information will decide about its credibility based on voting, *i.e.* a greater number of votes implies the truthfulness of the information. However, more work needs to be done in order to improve the above mentioned idea and to implement and evaluate it.

8.4.2 Whitewashing Detection in Non-homogenous Transmit Power Systems

More experimentation needs to be conducted with our whitewashing and Sybil attack detection scheme, presented in Chapter 7. In future work we aim to deploy more sophisticated attackers who can tamper with their radio device in order to transmit on non-homogenous transmit powers in the network. An adaptable threshold should be devised to detect such types of attacker in the network.

8.4.3 Identity Spoofing

A particularly important issue related to identity in MANETs is the detection of spoofing attacks. In an identity spoofing attack, a node can adopt another node's identity for malicious purposes, a process that is also often referred to as masquerading. Depending upon the motivation of the attacker, an attacker can spoof good as well as evil nodes' identities. However, the basic motivation usually is to escape detection and accountability. Work therefore needs to be done in order to develop a distributed detection mechanism for identity spoof detection in MANET environments. Since there is no centralized identity management or other controlling body for the management of the network, distributed identity spoof detection mechanisms represent an interesting and challenging task.

8.4.4 Tuning the Thresholds

In order to establish accurate threshold for the proposed schemes, such as fee count in Chapter 6, upper bound threshold for received signal strength, etc., further analysis should be carried out in future. Moreover, the proposed methods are needed to be deployed on real-world and large-scale networks in order to establish a relationship between the thresholds in simulated as well as in real-world scenarios.

8.4.5 Effect of Direct Interactions on Mobility Models

We have only shown in this thesis the effect of direct interactions on reputation-based schemes using random way point mobility model. Other mobility models should also be used in order to check its effect. Another work related to direct interactions is the contribution and consumption ratio metric, that should also need to be implemented in order to confirm its validity.

8.4.6 Combining Layers in a Single System

In our future work we will combine both of the schemes, *i.e.* fee-based deterrence and received signal strength based detection of whitewashers and Sybil attackers in a single system, in order to demonstrate the overall benefits of our proposed work.

8.5 Concluding Remarks

The problem of node misbehaviour or selfishness is serious in the self-organized MANETs. Since, there are no dedicated routers in the network; intermediate nodes forward other nodes packets in order to send data to nodes that do not fall in their direct communication ranges. In such an environment selfish behaviour of nodes can severely degrade network performance. Cooperation enforcement schemes have been proposed to tackle this issue; however, they by themselves are not secured. In this thesis, we presented the attacks that jeopardize the security of these schemes. These issues are direct interactions, whitewashing and Sybil attacks. We proposed methods to counteract these attacks. In addition to direct interaction, we also proposed a layered security architecture that tackles the whitewashing and Sybil attacks efficiently in a layered fashion. In the first line-of-defence, we proposed a scheme that pro-actively deters these attacks. While in the second line-of-defence, we proposed a method that efficiently detects the new identities of these attackers, when the attack is launched. The simulation results demonstrated that our proposed schemes worked better in discouraging and detecting these attackers while maintaining good network performance.

Appendix A Implementation of the Attacker Nodes

Commands

We modify the mobility and traffic generation scripts of NS-2 (which will be shown later) in order to implement our attack models[9]. We also modify their parameter list to pass the number of the attacker nodes along with the identity used per attacker. The modified commands are shown below.

For Mobility:

```
./setdest -v 1 -n 41 -p 10.0 -M 10.0 -t 600 -x 1000 -y 1000 -w 3 -i 5 > output
```

```
Note: -w is used for the number of Whitewashers and -i is used for the number of
identities used by a whitewasher node.
```

For Traffic:

```
ns modifiedcbr.tcl -type cbr -nn 42 -seed 1 -mc 20 -rate 2.0 -wn 3 -wids 5 >
output
```

```
Note: wn is used for number of Whitewashers, wids specifies the number of IDs used
for Whitewash.
```

Source Code of TCL Script

The TCL script is used as a front-end script for the NS-2 simulations. Its purpose is to specify the values the simulation scenarios, for example specifying network area, routing protocol, number of

[9] Due to the limitation of space, we do not include the source code for the reputation incorporated DSR, fee per identity and signal strength based detection which is presented in Chapter 4, 6 and 7 respectively.

nodes, simulation time, etc. The following is the TCL script we used for our Chapter 6 and 7 simulations.

```
# ======================================================================
# Define options
# ======================================================================

set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            CMUPriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(x)                  1000    ;# X dimension of the topography
set val(y)                  1000    ;# Y dimension of the topography
set val(ifqlen)             1000            ;# max packet in ifq
set val(seed)               1.0
set val(adhocRouting)       DSR
set val(nn)                 xxx             ;# how many nodes are simulated
set val(cp)                 "trgen_1"
set val(sc)                 "scen_1"
set val(stop)               600.0           ;# simulation time

# ======================================================================

Agent/Null set sport_        0
Agent/Null set dport_        0
Agent/CBR set sport_         0
Agent/CBR set dport_         0

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it

Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# the above parameters result in a nominal range of 250m

#Phy/WirelessPhy set RXThresh_ 8.91754e-10

#set nominal_range 250.0
#set configured_range -1.0
#set configured_raw_bitrate -1.0

Mac/802_11 set basicRate_ 1Mb
Mac/802_11 set dataRate_  11Mb


#
# Number of Whitewashers (WNODES) and Number of IDs per Whitewasher (WIDS)
#
```

```
set WIDS 5
set WNODES 3



# ======================================================================
# Main Program
# ======================================================================

#
# Initialize Global Variables
#

# create simulator instance

set ns_           [new Simulator]

# setup topography object

set topo    [new Topography]

# create trace object for ns and nam

set tracefd [open wtr_1.tr w]
set namtrace    [open wtr_1.nam w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
$ns_ use-newtrace

# define topology
$topo load_flatgrid $val(x) $val(y)

#
# Create God
#
set god_ [create-god $val(nn)]

#
# define how node should be created
#

#global node setting

$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                  -topoInstance $topo \
                  -agentTrace ON \
                 -routerTrace OFF \
                 -macTrace OFF \
                  -movementTrace OFF \
                  -energyModel EnergyModel \
                  -initialEnergy 100 \
                   -rxPower 0.3 \
                    -txPower 0.4

#
# Create the specified number of nodes [$val(nn)] and "attach" them
# to the channel.

for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
      $node_($i) random-motion 0           ;# disable random motion
}


#
# Define node movement model
#
puts "Loading connection pattern..."
source $val(cp)

#
# Define traffic model
#
puts "Loading scenario file..."
source $val(sc)


#
# Whitewashing nodes: IDs on off mechanisms.
#

for {set i 0} {$i < [expr { ($WNODES * $WIDS) }] } { incr i $WIDS } {
    for {set j 1} {$j < $WIDS} {incr j} {
        # puts "\$ns_ at 1.00000 \"\$node_([expr {\$i + \$j}]) off\""
         $ns_ at 1.00000 "$node_([expr {$i + $j}]) off"
      }
 }

set z 1

for {set k 0} {$k < $WIDS} { incr k } {
   for {set i 0} {$i < [expr { ($WNODES * $WIDS) }] } { incr i $WIDS } {
     for {set j 0} {$j < $WIDS} {incr j} {
       if {$j == $z && $k < $WIDS} {
         #puts "\$ns_ at [expr {(\$z * 100) + 0.500}] \"\$node_([expr {\$i +
\$j}]) on\""
         $ns_ at [expr {($z * 100) + 0.500}] "$node_([expr {$i + $j}]) on"
         }
       if {$j == [expr { $z - 1 }] && $k < [expr { $WIDS - 1 }] } {
         #puts "\$ns_ at [expr {(\$z * 100) }] \"\$node_([expr {\$i + \$j}])
off\""
          $ns_ at [expr {(\$z * 100) }] "$node_([expr {$i + $j}]) off"
          }
      }
    }
     incr z
}



# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {
    # 20 defines the node size in nam, must adjust it according to your scenario
    # The function must be called after mobility model is defined

    $ns_ initial_node_pos $node_($i) 40
}


#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."
$ns_ run
```

Source Code of Mobility Script

The purpose of the mobility script is to generate random mobility patterns for the nodes taking part in the simulation. This is file is used then within the TCL script. We modified this script (shown below) to implement our attack model of whitewashing and Sybil attacks.

```
/*


  (1) Input parameters
      <Original version>
            => -M maximum speed (minimum speed is zero as a default)
            => -p pause time (constant)
            => -n number of nodes
            => -x x dimension of space
            => -y y dimension of space

      <Modified version>
            => -s speed type (uniform, normal)
            => -m minimum speed > 0
            => -M maximum speed
            => -P pause type (constant, uniform)
            => -p pause time (a median if uniform is chosen)
            => -n number of nodes
            => -x x dimension of space
            => -y y dimension of space

  (2) In case of modified version, the steady-state speed distribution is applied
 to the first trip to eliminate any speed decay. If pause is not zero, the first
```

```
trip could be either a move or a pause depending on the probabilty that the first
trip is a pause. After the first trip regardless of whether it is
a move or a pause, all subsequent speeds are determined from the given speed
distribution (e.g., uniform or normal).

  (3) Refer to and use scenario-generating scripts (make-scen.csh for original
version, make-scen-steadystate.csh for modified version).

 */



extern "C" {
#include <assert.h>
#include <fcntl.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
#if !defined(sun)
#include <err.h>
#endif
};
#include "../../../tools/rng.h"

#include "setdest.h"
#include <iostream>

// #define        DEBUG
#define           SANITY_CHECKS
//#define         SHOW_SYMMETRIC_PAIRS


#define GOD_FORMAT      "$ns_ at %.12f \"$god_ set-dist %d %d %d\"\n"
#define GOD_FORMAT2     "$god_ set-dist %d %d %d\n"
#define NODE_FORMAT     "$ns_ at %.12f \"$node_(%d) setdest %.12f %.12f %.12f\"\n"
#define NODE_FORMAT2    "$node_(%d) setdest %.12f %.12f %.12f\n"
#define NODE_FORMAT3    "$node_(%d) set %c_ %.12f\n"

#define           RTG_INFINITY      0x00ffffff
#ifndef min
#define           min(x,y)    ((x) < (y) ? (x) : (y))
#define           max(x,y)    ((x) > (y) ? (x) : (y))
#endif
#define           ROUND_ERROR 1e-9
#ifndef PI
#define     PI          3.1415926
#endif


//#define WIDS 5              //********* Sohail **********

static int count = 0;

/* ======================================================================
   Function Prototypes
   ====================================================================== */
```

```
void        usage(char**);
void        init(void);
double      uniform(void);


/* ======================================================================
   Global Variables
   ====================================================================== */
const double      RANGE = 250.0;     // transmitter range in meters
double            TIME = 0.0;        // my clock;
double            MAXTIME = 0.0;     // duration of simulation
double            MAXX = 0.0;        // width of space
double            MAXY = 0.0;        // height of space
double            PAUSE = 0.0;       // pause time
double            MAXSPEED = 0.0;    // max speed
double            MINSPEED = 0.0;    // min speed
double            SS_AVGSPEED = 0.0;// steady-state avg speed
double            KAPPA = 0.0;       // normalizing constant
double            MEAN = 0.0;        // mean for normal speed
double            SIGMA = 0.0;       // std for normal speed
double            EXP_1_V = 0.0;     // expactation of 1/V
double            EXP_R = 0.0;       // expectation of travel distance R
double            PDFMAX = 0.0;      // max of pdf for rejection technique
u_int32_t         SPEEDTYPE = 1;     // speed type (default = uniform)
u_int32_t         PAUSETYPE = 1;     // pause type (default = constant)
u_int32_t         VERSION = 1;       // setdest version (default = original by CMU)
u_int32_t   NODES = 0;               // number of nodes
u_int32_t   RouteChangeCount = 0;
u_int32_t   LinkChangeCount = 0;
u_int32_t   DestUnreachableCount = 0;

//************** Sohail ************

int   wids = 0;   // Number of Identities per Whitewasher node
int   fw = 0;     // ID of first Whitewasher, in our case it will always be 0
int   wnodes = 0; // Number of Whitewasher nodes


Node        *NodeList = 0;
u_int32_t   *D1 = 0;
u_int32_t   *D2 = 0;


/* ======================================================================
   Random Number Generation
   ====================================================================== */
#define M         2147483647L
#define INVERSE_M ((double)4.656612875e-10)

char random_state[32];
RNG *rng;

double
uniform()
{
        count++;
        return rng->uniform_double() ;
}
```

```
/* =================================================================
   Misc Functions...
   ================================================================= */


/* compute the expectation of travel distance E[R] in a rectangle */
void
compute_EXP_R()
{
      #define csc(x)          (1.0/sin(x))            // csc function
      #define sec(x)          (1.0/cos(x))            // sec function
      #define sin2(x)         (sin(x)*sin(x))         // sin^2
      #define sin3(x)         (sin2(x)*sin(x))  // sin^3
      #define cos2(x)         (cos(x)*cos(x))         // cos^2
      #define cos3(x)         (cos2(x)*cos(x))  // cos^3

      double x = MAXX, y = MAXY;              // max x and max y
      double x2 = x*x, x3 = x*x*x;            // x^2 and x^3
      double y2 = y*y, y3 = y*y*y;            // y^2 and y^3

      double term1 = sin(atan2(y,x)) / 2.0 / cos2(atan2(y,x));
      double term2 = 0.5 * log( sec(atan2(y,x)) + y/x );
      double term3 = -1.0 * x3 / y2 / 60.0 / cos3(atan2(y,x)) + 1.0/60.0 * x3 /
y2;
      double term4 = (term1 + term2) * x2 / 12.0 / y + term3;

      double term5 = -1.0 * cos(atan2(y,x)) / 2.0 / sin2(atan2(y,x));
      double term6 = 0.5 * log( csc(atan2(y,x)) - x/y );
      double term7 = -1.0 * y3 / x2 / 60.0 / sin3(atan2(y,x)) + 1.0/60.0 * y3 /
x2;
      double term8 = -1.0 * (term5 + term6) * y2 / 12.0 / x + term7;

      EXP_R = (4 * (term4 + term8));                   // E[R]
}

void
usage(char **argv)
{
      fprintf(stderr, "\nusage:\n");
      fprintf(stderr,
            "\n<original 1999 CMU version (version 1)>\n %s\t-v <1> -n <nodes> -p
<pause time> -M <max speed>\n", argv[0]);
      fprintf(stderr,"\t\t-t <simulation time> -x <max X> -y <max Y> -w <wnodes> -
i <wids>\n");
      fprintf(stderr,"\nOR\n<modified 2003 U.Michigan version (version 2)>\n %s\t-
v <2> -n <nodes> -s <speed type> -m <min speed> -M <max speed>\n", argv[0]);
      fprintf(stderr,"\t\t-t <simulation time> -P <pause type> -p <pause time> -x
<max X> -y <max Y>\n");
      fprintf(stderr,"\t\t(Refer to the script files make-scen.csh and make-scen-
steadystate.csh for detail.) \n\n");
}

Void init()
{

      /*
       * Allocate memory for globals
       */
      NodeList = new Node[NODES];
      if(NodeList == 0) {
            perror("new");
            exit(1);
      }

      D1 = new u_int32_t[NODES * NODES];
      if(D1 == 0) {
            perror("new");
            exit(1);
      }
      memset(D1, '\xff', sizeof(u_int32_t) * NODES * NODES);

      D2 = new u_int32_t[NODES * NODES];
      if(D2 == 0) {
            perror("new");
            exit(1);
      }
      memset(D2, '\xff', sizeof(u_int32_t) * NODES * NODES);
}



extern "C" char *optarg;

int
main(int argc, char **argv)
{
      char ch;

      while ((ch = getopt(argc, argv, "v:n:s:m:M:t:P:p:x:y:w:i:o:")) != EOF) {

            switch (ch) {

            case 'v':
              VERSION = atoi(optarg);
              break;

            case 'n':
              NODES = atoi(optarg);
              break;

            case 's':
                  SPEEDTYPE = atoi(optarg);
                  break;

            case 'm':
                  MINSPEED = atof(optarg);
                  break;

            case 'M':
                  MAXSPEED = atof(optarg);
                  break;

            case 't':
                  MAXTIME = atof(optarg);
                  break;

            case 'P':
                  PAUSETYPE = atoi(optarg);
                  break;

            case 'p':
                PAUSE = atof(optarg);
                break;

          case 'x':
                MAXX = atof(optarg);
                break;

          case 'y':
                MAXY = atof(optarg);
                break;

          case 'w':
                wnodes = atof(optarg);
                break;


          case 'i':
                wids = atof(optarg);
                break;


          default:
                usage(argv);
                exit(1);
          }
     }

     if(MAXX == 0.0 || MAXY == 0.0 || NODES == 0 || MAXTIME == 0.0) {
          usage(argv);
          exit(1);
     }

     /* specify the version */
     if (VERSION != 1 && VERSION != 2) {
       printf("Please specify the setdest version you want to use. For original
1999 CMU version use 1; For modified 2003 U.Michigan version use 2\n");
       exit(1);
     }

     if (VERSION == 2 && MINSPEED <= 0) {
       usage(argv);
       exit(1);
     } else if (VERSION == 1 && MINSPEED > 0) {
       usage(argv);
       exit(1);
     }


     // The more portable solution for random number generation
     rng = new RNG;
     rng->set_seed(RNG::HEURISTIC_SEED_SOURCE);

/***********************************************************************

Steady-state avg speed and distribution depending on the initial distributions
given.
***********************************************************************/


     /* original setdest */
     if (VERSION == 1) {
            fprintf(stdout, "#\n# nodes: %d, pause: %.2f, max speed: %.2f, max x:
%.2f, max y: %.2f, wnodes: %d, wids: %d \n#\n", NODES, PAUSE, MAXSPEED, MAXX,
MAXY, wnodes, wids);
        }

        /* modified version */
        else if (VERSION == 2) {
            /* compute the expectation of travel distance in a rectangle */
            compute_EXP_R();

            /* uniform speed from min to max */
            if (SPEEDTYPE == 1) {
                EXP_1_V = log(MAXSPEED/MINSPEED) / (MAXSPEED - MINSPEED);   //
E[1/V]
                SS_AVGSPEED = EXP_R / (EXP_1_V*EXP_R + PAUSE);
        // steady-state average speed
                PDFMAX = 1/MINSPEED*EXP_R / (EXP_1_V*EXP_R + PAUSE) / (MAXSPEED-
MINSPEED);  // max of pdf for rejection technique
            }

            /* normal speed clipped from min to max */
            else if (SPEEDTYPE == 2) {
                int bin_no = 10000;
            // the number of bins for summation
                double delta = (MAXSPEED - MINSPEED)/bin_no; // width of each
bin
                int i;
                double acc_k, acc_e, square, temp_v;

                MEAN = (MAXSPEED + MINSPEED)/2.0;    // means for normal dist.
                SIGMA = (MAXSPEED - MINSPEED)/4.0;   // std for normal dist.
                /* computing a normalizing constant KAPPA, E[1/V], and pdf max
*/
                KAPPA = 0.0;
                EXP_1_V = 0.0;
                PDFMAX = 0.0;

                /* numerical integrals */
                for (i=0; i<bin_no; ++i) {
                    temp_v = MINSPEED + i*delta;  // ith v from min speed
                    square = (temp_v - MEAN)*(temp_v - MEAN)/SIGMA/SIGMA;

                    acc_k = 1.0/sqrt(2.0*PI*SIGMA*SIGMA)*exp(-0.5*square);
                    KAPPA += (acc_k*delta); // summing up the area of
rectangle

                    acc_e = 1.0/temp_v/sqrt(2.0*PI*SIGMA*SIGMA)*exp(-
0.5*square);
                    EXP_1_V += (acc_e*delta);//summing up for denominator of
pdf

                    /* find a max of pdf */
                    if (PDFMAX < acc_e) PDFMAX = acc_e;
                }
                EXP_1_V /= KAPPA;                // normalizing
                SS_AVGSPEED = EXP_R / (EXP_1_V*EXP_R + PAUSE);
                    // steady-state average speed
                PDFMAX = EXP_R*PDFMAX/KAPPA / (EXP_1_V*EXP_R + PAUSE);
                    // max of pdf for rejection technique
            }
            /* other types of speed for future use */
            else
                 ;

            fprintf(stdout, "#\n# nodes: %d, speed type: %d, min speed: %.2f, max
speed: %.2f\n# avg speed: %.2f, pause type: %d, pause: %.2f, max x: %.2f, max y:
%.2f\n#\n",
                    NODES , SPEEDTYPE, MINSPEED, MAXSPEED, SS_AVGSPEED, PAUSETYPE,
PAUSE, MAXX, MAXY);
       }


       init();


//********************** Sohail (start) ************************


            for( int i = fw; i < (wnodes * wids);  ){
               NodeList[i].Setnoch( PARENT, wids, i );
       //Node_type = PARENT, No. of IDS per Whitewasher, ID of First Whitewasher (=
0)
               i = i + wids;
            }


            u_int32_t i;

            for(i = 0; i < NODES; i++) {

            if ( NodeList[i].node_type == NONE)
                 NodeList[i].PrintPosition(i);
            if ( NodeList[i].node_type == PARENT){
            for (int j=0; j < NodeList[i].noch; j++){
                 NodeList[i].PrintPosition(i+j);
            }
                 //i = i + NodeList[i].noch;
            }

            if ( NodeList[i].node_type == CHILD) { }
                 //return;

            }

//********************** Sohail (End) ************************


       while(TIME <= MAXTIME) {
            double nexttime = 0.0;
            u_int32_t i;

            for(i = 0; i < NODES; i++) {

                 NodeList[i].Update();
            }

            for(i = 0; i < NODES; i++) {
                 Node *n = &NodeList[i];

                 if(n->time_transition > 0.0) {
                      if(nexttime == 0.0)
                           nexttime = n->time_transition;
```

```
                        else
                             nexttime = min(nexttime, n->time_transition);
                   }

                   if(n->time_arrival > 0.0) {
                        if(nexttime == 0.0)
                             nexttime = n->time_arrival;
                        else
                             nexttime = min(nexttime, n->time_arrival);
                   }

              }


              assert(nexttime > TIME + ROUND_ERROR);
              TIME = nexttime;
         }

         //show_counters();

         int of;
         if ((of = open(".rand_state",O_WRONLY | O_TRUNC | O_CREAT, 0777)) < 0) {
           fprintf(stderr, "open rand state\n");
           exit(-1);
           }
         for (unsigned int i = 0; i < sizeof(random_state); i++)
             random_state[i] = 0xff & (int) (uniform() * 256);
         if (write(of,random_state, sizeof(random_state)) < 0) {
           fprintf(stderr, "writing rand state\n");
           exit(-1);
           }
         close(of);
}


/* ===============================================================
   Node Class Functions
   =============================================================== */

u_int32_t Node::NodeIndex = 0;

Node::Node()
{
         //u_int32_t i;

         index = NodeIndex++;

         //if(index == 0)
         //    return;

         route_changes = 0;
       link_changes = 0;

//************************* Sohail *******************

         node_type = NONE;
         //children = false;
         //child = NULL;
```

```
/**********************************************************************

Determine if the first trip is a pause or a move with the steady-state pdf
    **********************************************************************/

      /* original version */
      if (VERSION == 1) {
              time_arrival = TIME + PAUSE;              // constant pause
      }

      /* modified version */
      else if (VERSION == 2) {
            /* probability that the first trip would be a pause */
            double prob_pause = PAUSE / (EXP_1_V*EXP_R + PAUSE);

            /* the first trip is a pause */
            if (prob_pause > uniform()) {
                  /* constant pause */
                  if (PAUSETYPE == 1) {
                        time_arrival = TIME + PAUSE;        // constant pause
                  }
                  /* uniform pause */
                  else if (PAUSETYPE == 2) {
                        time_arrival = TIME + 2*PAUSE*uniform();  // uniform pause
[0, 2*PAUSE]
                  }

                  first_trip = 0;   // indicating the first trip is a pause
            }
            /* the first trip is a move based on the steady-state pdf */
            else {
                  time_arrival = TIME;
                  first_trip = 1;         // indicating the first trip is a move
            }
      }


      time_update = TIME;
      time_transition = 0.0;

      position.X = position.Y = position.Z = 0.0;
      destination.X = destination.Y = destination.Z = 0.0;
      direction.X = direction.Y = direction.Z = 0.0;
      speed = 0.0;


      RandomPosition();


      //fprintf(stdout, NODE_FORMAT3, index, 'X', position.X);
      //fprintf(stdout, NODE_FORMAT3, index, 'Y', position.Y);
      //fprintf(stdout, NODE_FORMAT3, index, 'Z', position.Z);


 }
      //**************** Sohail ***************

 void
 Node::Setnoch(NODE_T nodetype, int nofch, int cur )
 {
   NodeList[cur].node_type = nodetype;
```

```
  NodeList[cur].noch = nofch;
  for(int i = cur+1; i <= (cur + noch)-1; i++ )
      NodeList[i].node_type = CHILD;

}


void
Node::PrintPosition(u_int32_t nod)
{

      fprintf(stdout, NODE_FORMAT3, nod, 'X', position.X);
      fprintf(stdout, NODE_FORMAT3, nod, 'Y', position.Y);
      fprintf(stdout, NODE_FORMAT3, nod, 'Z', position.Z);

}


void
Node::RandomPosition()
{
    position.X = uniform() * MAXX;
    position.Y = uniform() * MAXY;
      position.Z = 0.0;
}


void
Node::RandomDestination()
{
      destination.X = uniform() * MAXX;
      destination.Y = uniform() * MAXY;
      destination.Z = 0.0;
      assert(destination != position);
}



/*****************************************************************
 * Speeds are chosen based on the given type and distribution
 *****************************************************************/

void
Node::RandomSpeed()
{
      /* original version */
      if (VERSION == 1) {
            speed = uniform() * MAXSPEED;
            assert(speed != 0.0);
      }

      /* modified version */
      else if (VERSION == 2) {
            /* uniform speed */
            if (SPEEDTYPE == 1) {
                  /* using steady-state distribution for the first trip */
                  if (first_trip == 1) {
                        /* pick a speed by rejection technique */
                        double temp_v, temp_fv;

                        do {
```

```
                        temp_v = uniform() * (MAXSPEED - MINSPEED) + MINSPEED;
                             temp_fv = uniform() * PDFMAX;
                        } while (temp_fv > 1/temp_v*EXP_R / (EXP_1_V*EXP_R +
PAUSE) / (MAXSPEED-MINSPEED));

                        speed = temp_v;
                        first_trip = 0;         // reset first_trip flag
                   }
                   /* using the original distribution from the second trip on */
                   else {
                   speed = uniform() * (MAXSPEED - MINSPEED) + MINSPEED;
                        assert(speed != 0.0);
                   }
              }
              /* normal speed */
              else if (SPEEDTYPE == 2) {
                   /* using steady-state distribution for the first trip */
                   if (first_trip == 1) {
                        double temp_v, temp_fv, square, fv;

                        /* rejection technique */
                        do {
                        temp_v = uniform() * (MAXSPEED - MINSPEED) + MINSPEED;
                             temp_fv = uniform() * PDFMAX;
                             square = (temp_v - MEAN)*(temp_v -
MEAN)/SIGMA/SIGMA;
                             fv = 1/KAPPA/sqrt(2.0*PI*SIGMA*SIGMA) * exp(-
0.5*square);
                        } while (temp_fv > 1.0/temp_v*fv*EXP_R / (EXP_1_V*EXP_R +
PAUSE));

                        speed = temp_v;
                        first_trip = 0;
                   }
                   /* using the original distribution from the second trip on */
                   else {
                        double temp_v, temp_fv, square;
                        double max_normal = 1.0/KAPPA/sqrt(2.0*PI*SIGMA*SIGMA);
                             // max of normal distribution

                        /* rejection technique */
                        do {
                        temp_v = uniform() * (MAXSPEED - MINSPEED) + MINSPEED;
                             temp_fv = uniform() * max_normal;
                             square = (temp_v - MEAN)*(temp_v -
MEAN)/SIGMA/SIGMA;
                        } while (temp_fv > max_normal * exp(-0.5*square));

                        speed = temp_v;
                        assert(speed != 0.0);
                   }
              }
              /* other types of speed for future use */
              else
                   ;
         }
}


void
Node::Update()
```

```
{
      position += (speed * (TIME - time_update)) * direction;

      if(TIME == time_arrival) {
            vector v;

            if(speed == 0.0 || PAUSE == 0.0) {

            RandomDestination();
                  RandomSpeed();

                  v = destination - position;
                  direction = v / v.length();
                  time_arrival = TIME + v.length() / speed;
            }
            else {

                  destination = position;
                  speed = 0.0;

                  /* original version */
                  if (VERSION == 1) {
                        time_arrival = TIME + PAUSE;
                  }
                  /* modified version */
                  else if (VERSION == 2) {
                        /* constant pause */
                        if (PAUSETYPE == 1) {
                              time_arrival = TIME + PAUSE;
                        }
                        /* uniform pause */
                        else if (PAUSETYPE == 2) {
                              time_arrival = TIME + 2*PAUSE*uniform();
                        }
                  }
            }

      //**************** Sohail ***************


            if ( NodeList[index].node_type == NONE) {
            fprintf(stdout, NODE_FORMAT,
                  TIME, index, destination.X, destination.Y, speed);

            }

            else
             if ( NodeList[index].node_type == PARENT){
                  for (int i=0; i < NodeList[index].noch; i++)
                  fprintf(stdout, NODE_FORMAT,
                  TIME, (index + i), destination.X, destination.Y, speed);
                  }
            else
                return;


      }

      time_update = TIME;
      time_transition = 0.0;
}
```

Source Code of Traffic Generator Script

The purpose of this script is to generate random traffic patterns (connections) for nodes in the simulation. We modify this script (shown below) for our attack model implementation.

```
# ======================================================================
# Default Script Options
# ======================================================================
set opt(nn)        0             ;# Number of Nodes
set opt(seed)            0.0
set opt(mc)        0
set opt(pktsize)   64

set opt(rate)            0
set opt(interval)  0.0           ;# inverse of rate
set opt(type)            ""

set opt(wn)        0             ;# Whitewasher nodes  3
set opt(wids)            0            ;# Ids per w-node  5
# ======================================================================

proc usage {} {
    global argv0

    puts "\nusage: $argv0 \[-type cbr|tcp\] \[-nn nodes\] \[-seed seed\] \[-mc
connections\] \[-rate rate\] \[-wn wnodes\] \[-wids wids\]\n"
}

proc getopt {argc argv} {
      global opt
      lappend optlist nn seed mc rate type wn wids

      for {set i 0} {$i < $argc} {incr i} {
            set arg [lindex $argv $i]
            if {[string range $arg 0 0] != "-"} continue

            set name [string range $arg 1 end]
            set opt($name) [lindex $argv [expr $i+1]]
      }
}


proc pre-print { src dst stime } {

      global opt

# wn is the number of Whitewashers, wids is the number of IDs used per
Whitewasher, UP is the uper bound.

      set UP [expr {$opt(wids) * $opt(wn) - 1}]

      if {($src > $UP) && ($dst > $UP)} {
          set stop 0
          print $src $dst $stime $stop
       }

       if {($src <= $UP) && ($dst > $UP)} {
          for {set i $src} {$i < [expr { ($src + $opt(wids)) - ($src % $opt(wids))
}] } {incr i} {
          set stop [expr { (($i % $opt(wids)) + 1) * 100 }]
          if {$stop > 400} {
             set stop 0
           }
          print $i $dst $stime $stop
          set stime [expr {$stop + 1}]
          }
       }


       if {($src > $UP) && ($dst <= $UP)} {
          for {set i $dst} {$i < [expr { ($dst + $opt(wids)) - ($dst % $opt(wids))
}] } {incr i} {
          set stop [expr { (($i % $opt(wids)) + 1) * 100 }]
          if {$stop > 400} {
             set stop 0
           }
          print $src $i $stime $stop
          set stime [expr {$stop + 1}]
          }
       }


       if {($src <= $UP) && ($dst <= $UP)} {
          for {set i $dst} {$i < [expr { ($dst + $opt(wids)) - ($dst % $opt(wids))
}] } {incr i} {
          set stop [expr { (($i % $opt(wids)) + 1) * 100 }]
          if {$stop > 400} {
             set stop 0
           }
          print $src $i $stime $stop
          incr src
          set stime [expr {$stop + 1}]
          }
       }
}



proc adjust_id { stime wid } {

       global opt

       if { $stime <= 100 } {
          set ID [expr {$wid * $opt(wids)}]
       } elseif { [expr {($stime > 100) && ($stime <= 400)}] } {

          set temp [expr { ($stime - ($stime % 100)) / 100 }]
          set ID [expr {($wid * $opt(wids)) + $temp}]
       } elseif { [expr {($stime > 400)}] } {

          set ID [expr {($wid * $opt(wids)) + ($opt(wids) - 1)}]
       }
       return $ID
```

```
}


proc create-cbr-connection { src dst stime } {

       global opt

       set UP [expr {($opt(wids) * $opt(wn)) - 1}]

       if {($src >= $opt(wn)) && ($dst >= $opt(wn))} {
          set src [expr {$src + $UP}]
          set dst [expr {$dst + $UP}]
          pre-print $src $dst $stime
       }

       if {($src < $opt(wn)) && ($dst >= $opt(wn))} {
          set src [adjust_id $stime $src]
          set dst [expr {$dst + $UP}]
          pre-print $src $dst $stime
       }

       if {($src >= $opt(wn)) && ($dst < $opt(wn))} {
          set src [expr {$src + $UP}]
          set dst [adjust_id $stime $dst]
          pre-print $src $dst $stime
       }


       if {($src < $opt(wn)) && ($dst < $opt(wn))} {
          set src [adjust_id $stime $src]
          set dst [adjust_id $stime $dst]
          pre-print $src $dst $stime
       }
}



proc print { src dst stime stoptime } {

       global rng cbr_cnt opt

       puts "#\n# $src connecting to $dst at time $stime\n#"

       ##puts "set cbr_($cbr_cnt) \[\$ns_ create-connection \
             ##CBR \$node_($src) CBR \$node_($dst) 0\]";
       puts "set udp_($cbr_cnt) \[new Agent/UDP\]"
       puts "\$ns_ attach-agent \$node_($src) \$udp_($cbr_cnt)"
       puts "set null_($cbr_cnt) \[new Agent/Null\]"
       puts "\$ns_ attach-agent \$node_($dst) \$null_($cbr_cnt)"
       puts "set cbr_($cbr_cnt) \[new Application/Traffic/CBR\]"
       puts "\$cbr_($cbr_cnt) set packetSize_ $opt(pktsize)"
       puts "\$cbr_($cbr_cnt) set interval_ $opt(interval)"
       puts "\$cbr_($cbr_cnt) set random_ 1"
       puts "\$cbr_($cbr_cnt) set maxpkts_ 10000"
       puts "\$cbr_($cbr_cnt) attach-agent \$udp_($cbr_cnt)"
       puts "\$ns_ connect \$udp_($cbr_cnt) \$null_($cbr_cnt)"

       puts "\$ns_ at $stime \"\$cbr_($cbr_cnt) start\""

       if { $stoptime != 0 } {
```

```
      puts "\$ns_ at [expr {$stoptime - 1} ] \"\$cbr_($cbr_cnt) stop\""
      }

      incr cbr_cnt
}

proc create-tcp-connection { src dst } {
      global rng cbr_cnt opt

      set stime [$rng uniform 0.0 180.0]

      puts "#\n# $src connecting to $dst at time $stime\n#"

      puts "set tcp_($cbr_cnt) \[\$ns_ create-connection \
            TCP \$node_($src) TCPSink \$node_($dst) 0\]";
      puts "\$tcp_($cbr_cnt) set window_ 32"
      puts "\$tcp_($cbr_cnt) set packetSize_ $opt(pktsize)"

      puts "set ftp_($cbr_cnt) \[\$tcp_($cbr_cnt) attach-source FTP\]"


      puts "\$ns_ at $stime \"\$ftp_($cbr_cnt) start\""

      incr cbr_cnt
}

# ======================================================================

getopt $argc $argv

if { $opt(type) == "" } {
    usage
    exit
} elseif { $opt(type) == "cbr" } {
    if { $opt(nn) == 0 || $opt(seed) == 0.0 || $opt(mc) == 0 || $opt(rate) == 0 }
{
      usage
      exit
    }

    set opt(interval) [expr 1 / $opt(rate)]
    if { $opt(interval {
      puts "\ninvalid sending rate $opt(rate)\n"
      exit
    }
}

puts "#\n# nodes: $opt(nn), max conn: $opt(mc), send rate: $opt(interval), seed:
$opt(seed)\n#"

set rng [new RNG]
$rng seed $opt(seed)

set stime [$rng uniform 10.0 280.0]
set stime [expr { int($stime) }]

set cbr_cnt 0
set src_cnt 0


set u2 [new RandomVariable/Uniform]
```

```
$u2 set min_ 0
$u2 set max_ [expr {$opt(nn) - 14}]
$u2 use-rng $rng

set u3 [new RandomVariable/Uniform]
$u3 set min_ 0
$u3 set max_ [expr {$opt(nn) - 15}]
$u3 use-rng $rng

for {set i 0} {$i < [expr {$opt(mc) }] } {incr i} {

      set s [$u2 value]
      set src [expr { int($s) }]

      set d [$u3 value]
      set dst [expr { int($d) }]

      if {$dst >= $src} {set dst [expr ($dst + 1)]}

      incr src_cnt

      create-cbr-connection $src $dst $stime

}

puts "#\n#Total sources/connections: $src_cnt/$cbr_cnt\n#"
```

Appendix B DSR Structure in NS2

The Dynamic Source Routing (DSR) protocol is one of the main routing protocols developed for mobile ad hoc networks; we discussed it in Section 2.6. Since most of our work was related to this protocol (for example, we incorporate our reputation-based scheme into DSR), it is important to discuss its main structure and functions. In this Appendix, we will discuss with the help of flowchart diagrams the DSR structure used in the NS-2 simulator [43-44]. The main functions will be explained, such as how a sender, a destination and a forwarder can perform their routing duties, using the diagrams B.1, B.2 and B.3 with the help of pseudo code. For clarity, each process or condition is marked with a number. The plus sign attached to a process indicates that this process is further extended in the next diagram.

Sender

When a sender node *S* has data to send to an unknown destination *D*, the following steps will be followed.

- Packets sent by the application agent or MAC layer agent will be received at the recv() function in the DSR agent, denoted by 1 in Figure B.1. We will refer these labels as boxes or processes interchangeably followed by there numbers, for example box 1.
- The packet will contain two parts: a data payload and a header (contains routing information). The packet header is checked to establish whether it contains a valid source route (SR). In the case where the packet has no established route, the decision path from box 2 in Figure B.1 will be No.
- In box 3, a check is made to establish whether it is a broadcast packet, in this case, it is not a broadcast packet, so the decision will be No.
- Packet is passed to a handler function of packets having no SR, denoted by box 4 and which is further extended in Figure B.2.
- In box 5, if the packet is not destined for the node that receives it (here node *S* will check it as well because it acts like a receiver from the upper layers or from other functions), then go to box 6.

- As the packet has no SR, *S* will check its route cache to find out if it has a route to the destination *D*; this is denoted by box 6. If there is no route found in the cache then go to box 7.
- Process 7 will buffer the data packet and try to find a route for it. In order to find the route to the destination, process 7 will make a route request packet in box 7a and then box 7b will pass it to the process 21 which is extended in Figure B.3. Process 21 will further broadcast the route request on the MAC layer following the order specified using processes 21-22-23-24-26.

We will show in the next section that when *S* receives a route reply as a destination node the node will store the SR and send the buffered data packet on that SR using processes 19-25.

Receiver

A node acting as a destination can receive data, route requests, route replies and route error packets, as explained below.

- The packet follows the path 1-2-8-9.
- At process 9, the route request follows the path 10-11; route error follows the path 10-12-13; route reply follows the path 10-12-14-15; and data packets follow the path 10-12-14-16. At process 16, any packet destined for the receiver (*i.e. D*) will be given to the upper layers (to the application layer via a demultiplexer or demux).

Forwarder

Nodes forward unicast (such as data and route reply) as well as broadcast (such as route request) packets.

- The unicast packets will follow the path 1-2-8-17-19-20-21-23-24-25, where process 19 is the main function for forwarding the packets.
- The broadcast packets will follow the path 1-2-8-17-18-21-22-23-24-26, where process 18 will investigate the drop reasons, for example the forwarder node drop the packet because the threshold for the number of re-transmissions is exceeded, etc.

The condition for checking whether the receiver is the destination at process 22 seems meaningless because it has already checked for it at process 8. The reason for this is that other functions also use the process 21, where it is necessary to check for the destination and if it turns out to be true, the packet is passed again to process 1, *i.e.* from process 28 to process 1.

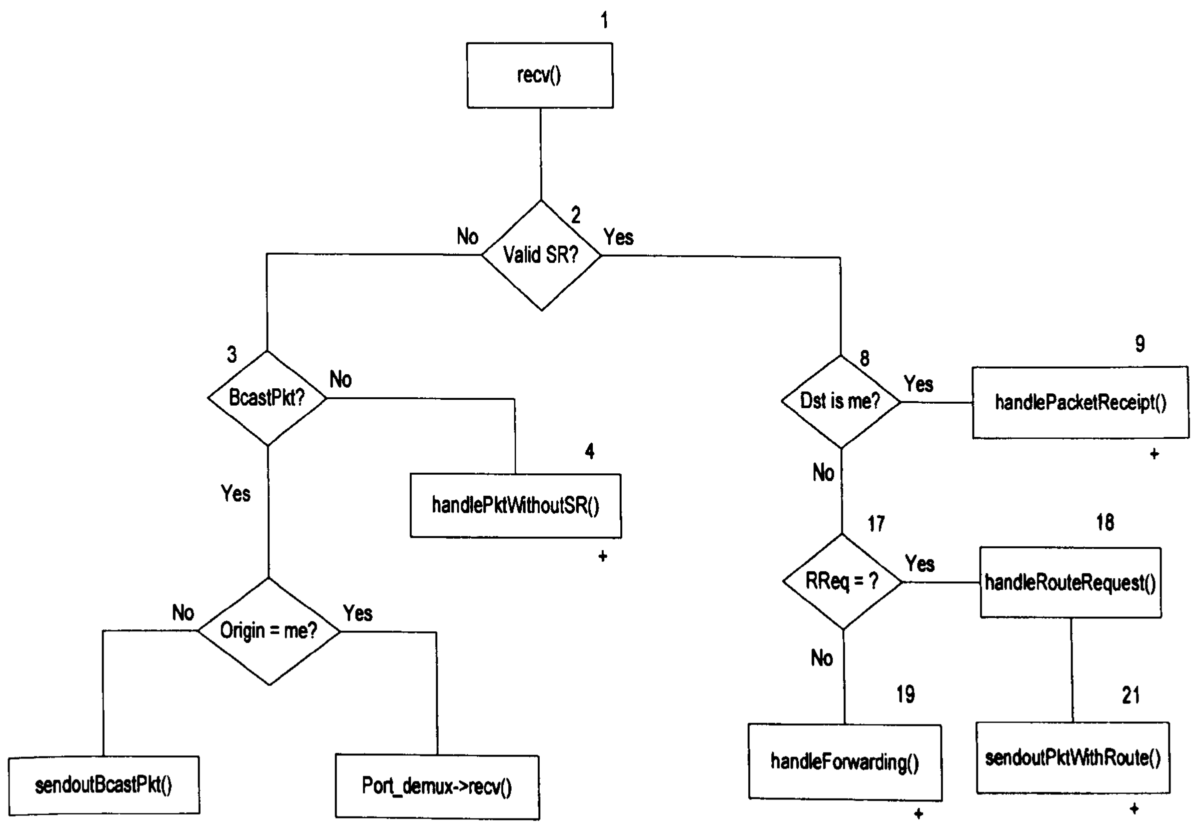

Figure B.1: Flowchart of DSR's recv() function in NS-2

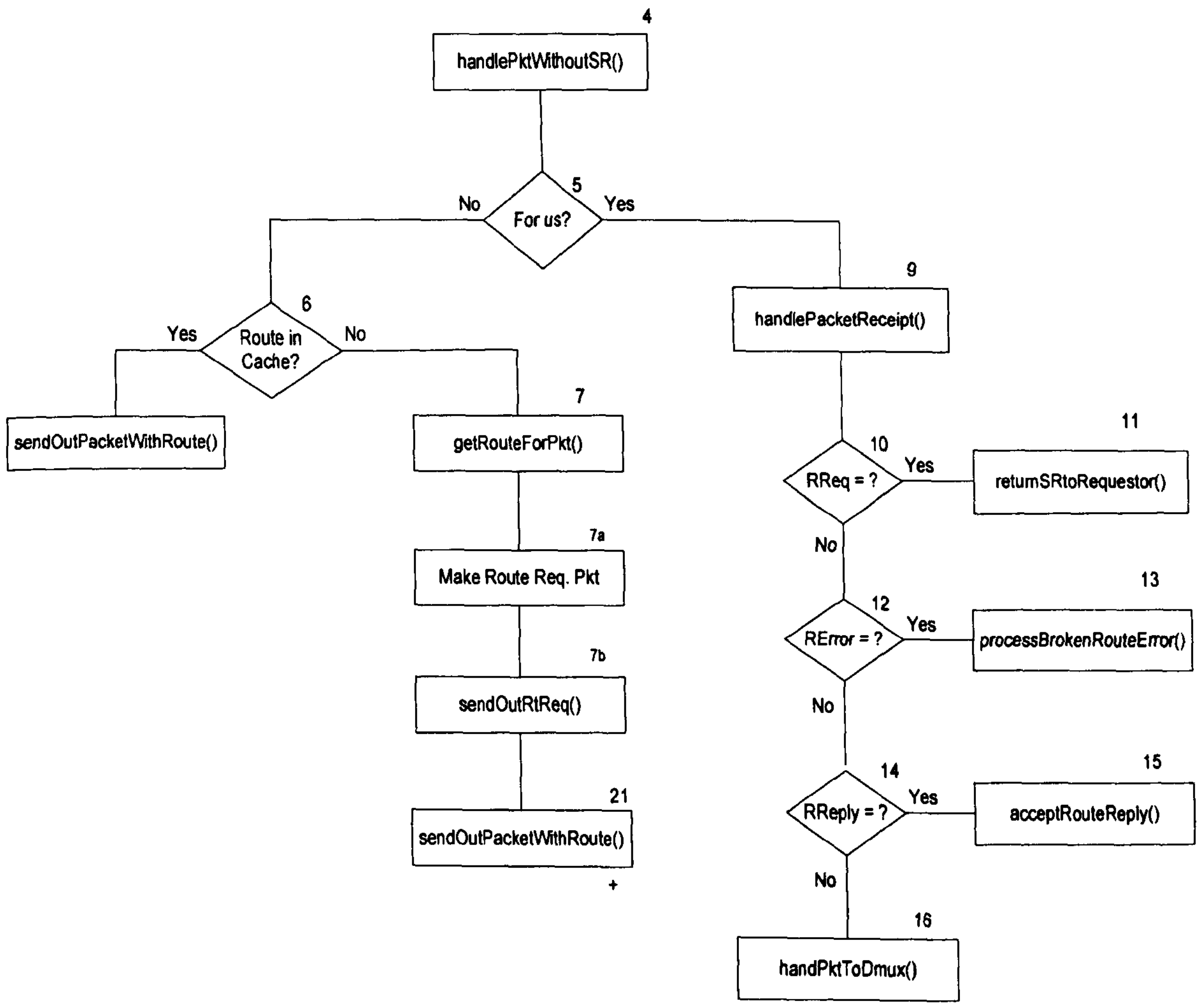

Figure B.2: Flowchart of DSR's handlePktWithoutSR() function in NS-2

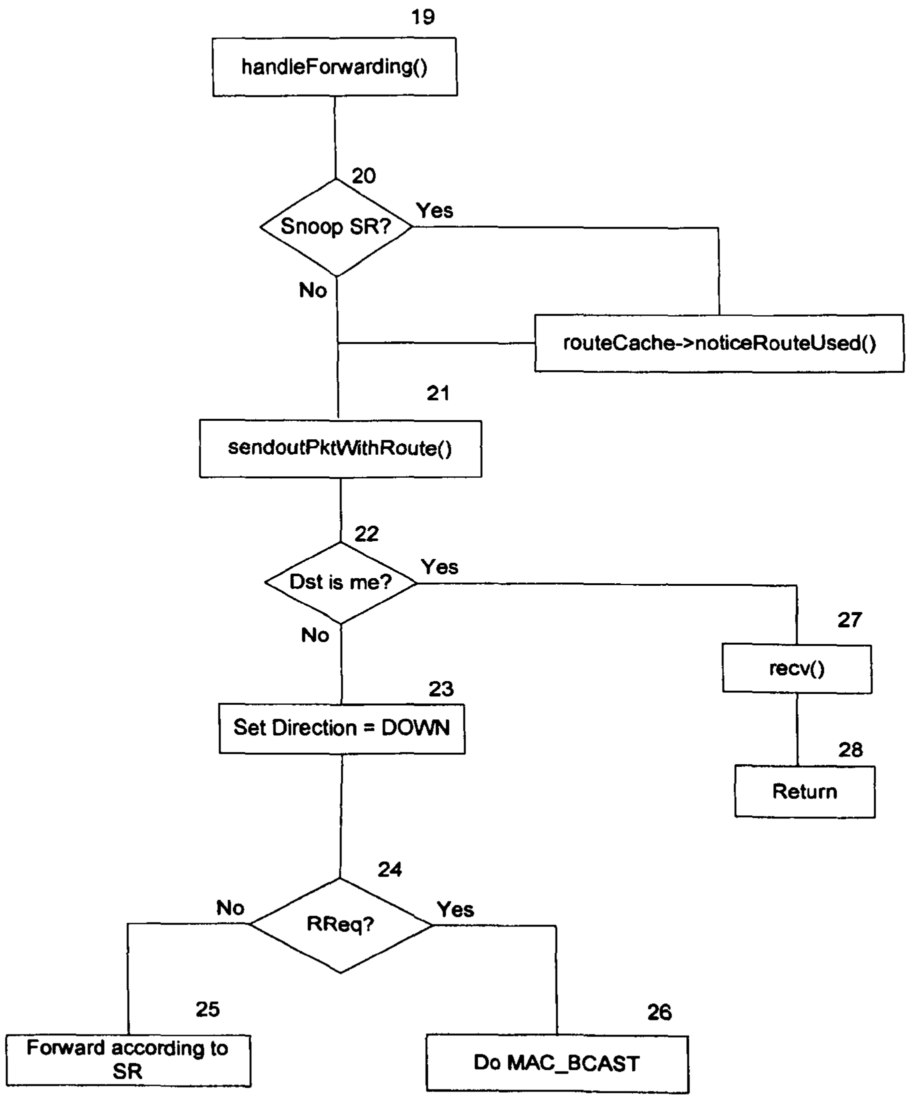

Figure B.3: Flowchart of DSR's handleForwarding function in NS-2

Appendix C Analysing RSS Fluctuation in Sun Spots

In order to analyse the fluctuation occur in the RSS readings of Sun Spot sensors, we conducted an experiment in which 1000 samples of RSS were collected at three different positions. The sensor was placed at 1, 15 and 33 feet distance from the base station. We found out that 33 feet (approximately equal to 10 meters) is the boundary of the radio range of the base station. The descriptive statistics and data distribution (using SPSS) of each position is given below.

Position 1:

Table C-1: Descriptive statistics for 1 foot distance

Descriptive Statistics

	N	Mean		Std. Deviation	Variance
	Statistic	Statistic	Std. Error	Statistic	Statistic
RSS_P_1	1000	-16.9180	.02408	.76148	.580
Valid N (listwise)	1000				

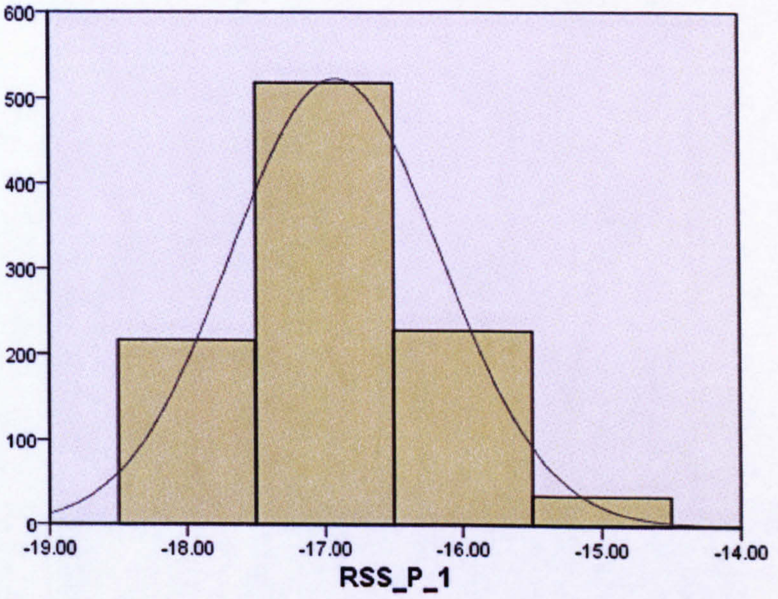

Figure C-1: Data distribution for 1 foot distance

Position 2:

Table C-2: Descriptive statistics for 15 feet distance

Descriptive Statistics

	N	Mean		Std. Deviation	Variance
	Statistic	Statistic	Std. Error	Statistic	Statistic
RSS_P_2	1000	-32.5970	.02571	.81317	.661
Valid N (listwise)	1000				

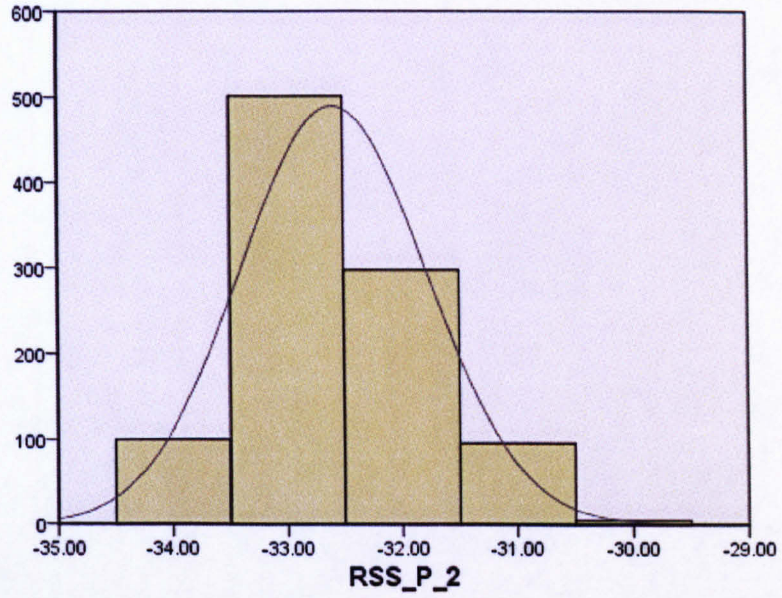

Figure C-2: Data distribution for 15 feet distance

Position 3:

Table C-3: Descriptive statistics for 33 feet distance

Descriptive Statistics

	N	Mean		Std. Deviation	Variance
	Statistic	Statistic	Std. Error	Statistic	Statistic
RSS_P_3	1000	-44.3110	.03560	1.12585	1.268
Valid N (listwise)	1000				

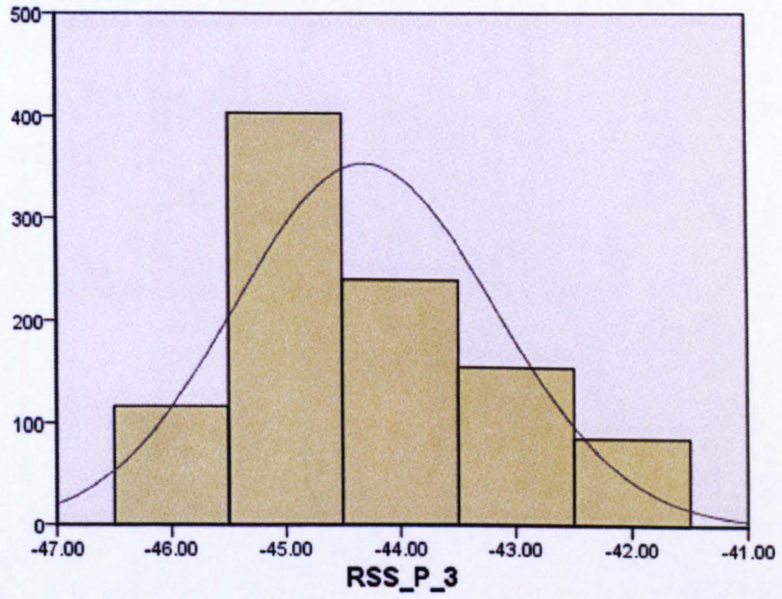

Figure C-3: Data distribution for 33 feet distance

Distance Produced from Fluctuations:

The power received at any distance *d* is inversely proportional to the *m*th power of the distance [102], *i.e.*

$$P_d \propto \frac{1}{d^m} \qquad \text{Eq. C-1}$$

Where P_d is the received power at distance *d* and in free space and in Line-Of-Sight (LOS) conditions, the path loss exponent *m* is 2. The received power at any distance can be calculated by using Eq. C-1 as below.

$$\frac{P_d}{P_{d_0}} = \frac{d_0^m}{d^m} \qquad \text{Eq. C-2}$$

Taking the logarithm of Eq. C-2, we can get the expression below for the received power at any distance *d* from the transmitter, *i.e.* P_d where $d > d_0$.

$$P_d\,(dbm) = P_{d_0}\,(dbm) - 10\,\log_{10}\left(\frac{d}{d_0}\right)^m \qquad \text{Eq. C-3}$$

Where d_0 is called the reference distance, we know from our Sun Spot experiment, Section 7.3, d_0 is 1 foot (0.30 meter) and P_{d_0} is approximately -17 (dbm) from Table C-1. We will use Eq. C-3 in order to find out the received signal strength at the boundary, *i.e.* where $d = 10$ meters, as shown below.

$$P_{10} = (-17) - \left[20 \log_{10}\left(\frac{10}{0.30}\right)\right]$$

$$P_{10} = (-17) - [20 \log_{10}(33.33)]$$

$$P_{10} = (-17) - [20 \times 1.52]$$

$$P_{10} = -47.4\, dbm$$

In our testbed experiment, the received power for 10 meters distance is -44.3 dbm, as shown in Table C-3. The reason for lower received signal strength in our indoor testbed experiment is that in real-world scenarios, there are other factors affecting the radio propagation, such as obstacles, etc. Now our aim is to find out the *distance* d_v resulted from 2.24 dbm variation in the signal strength, which is a two standard deviation in received signal strength at the boundary, shown in Table C-3. The received signal strength, when the 2.24 dbm variation or 2 Standard Deviation (SD) in the data exists, can be computed as

$$P_{d_v} - P_{10} = 2.24\, dbm$$

$$P_{d_v} = 2.24 - 47.4$$

$$P_{d_v} = -45.16\, dbm.$$

Whereas distance d_v can be computed using Eq. C-3 as the following.

$$d_v = 10^{\left(\frac{-(P_{d_v} - P_{d_0}) + 20 \log_{10}(d_0)}{20}\right)}$$

$$d_v = 10^{\left(\frac{-(-45.16 - 17) + 20 \log_{10}(0.30)}{20}\right)}$$

$$d_v = 10^{(0.91)}$$

$$d_v = 8.13\, m$$

Appendix D Ratio of RSSIs

The author [78] show that no sensor node can hide its location when it is monitored by four or more nodes as follows.

Suppose a node to be monitored transmits at the power P_t, then sensor a will receive this signal at power

$$P_{r(a)} = P_t K / d_a{}^{\alpha}. \qquad \text{Eq. D-1}$$

Where $P_{r(a)}$ is the received power at the sensor node a, K is constant, d_a is the distance between sensor node a and the monitored node, and α is the distance power gradient.

The ratio of the received signal at two different sensors, from a to b ($a \neq b$) is

$$\begin{aligned} P_{r(a)}/P_{r(b)} &= (P_t K/d_a{}^{\alpha})/(P_t K/d_b{}^{\alpha}) \qquad \text{Eq. D-2} \\ &= (d_b/d_a)^{\alpha}. \end{aligned}$$

$$d_b/d_a = \left(P_{r(a)}/P_{r(b)}\right)^{\frac{1}{\alpha}}. \qquad \text{Eq. D-3}$$

The beauty of Eq. D-3 is that it is independent of the transmit power P_t. Now assume that the location of the monitored node in two-dimensional Cartesian coordinates is (x, y), with sensor a, b, c, and d locations as (x_a, y_a), (x_b, y_b), (x_c, y_c), *and* (x_d, y_d) respectively. The location (x, y) can be determined by solving the following equation.

$$\begin{aligned} &((x - x_a) + (y - y_a))^2 \\ &= \left(P_{r(a)}/P_{r(b)}\right)^{\frac{1}{\alpha}} ((x - x_b) + (y - y_b))^2 \\ &= \left(P_{r(a)}/P_{r(c)}\right)^{\frac{1}{\alpha}} ((x - x_c) + (y - y_c))^2 \\ &= \left(P_{r(a)}/P_{r(d)}\right)^{\frac{1}{\alpha}} ((x - x_d) + (y - y_d))^2. \qquad \text{Eq. D-4} \end{aligned}$$

Appendix E Publications

E.1. S. Abbas, M. Merabti, and D.Llewellyn-Jones, "A Reputation Based Scheme to Deter Identity Based Attacks for Clustered MANETs," in *The 10th Annual Conference on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, 2009, pp. 243-248.

E.2. S. Abbas, M. Merabti, and D. Llewellyn-Jones, "Signal Strength Based Sybil Attack Detection in Wireless Ad Hoc Networks," in *Second International Conference on Developments in eSystems Engineering (DESE)*, 2009, pp. 190-195.

E.3. S. Abbas, M. Merabti, and D.Llewellyn-Jones, "A Survey of Reputation Based Schemes for MANET," in *The 11th Annual Conference on the Convergence of Telecommunications, Networking & Broadcasting (PGNet 2010)*, Liverpool, UK, 2010.

E.4. S. Abbas, M. Merabti, and D.Llewellyn-Jones, "Deterring Whitewashing Attacks in Reputation Based Schemes for Mobile Ad hoc Networks," in *The 3rd IFIP/IEEE Wireless Days*, Venice Italy, 2010.

E.5. S. Abbas, M. Merabti, and D.Llewellyn-Jones, "The Effect of Direct Interactions on Reputation Based Schemes in Mobile Ad hoc Networks," in *The 5th IEEE PerNets workshop 2011, in conjunction with IEEE CCNC conference,* Nevada USA, 2011.

E.6. S. Abbas, M. Merabti, and D.Llewellyn-Jones, "Identity-based Attacks against Reputation-based Systems in MANETs," in *The 12th Annual Conference on the Convergence of Telecommunications, Networking & Broadcasting (PGNet). Liverpool, UK,* 2011.

Bibliography

[1] C. K. Toh, "Future Application Scenarios for MANET-Based Intelligent Transportation Systems," in *Future Generation Communication and Networking (FGCN)*, 2007, pp. 414-417.

[2] Feldman, M., Papadimitriou, C., *et al.*, "Free-riding and whitewashing in peer-to-peer systems," *IEEE Journal on Selected Areas in Communications,* vol. 24, pp. 1010-1019, 2006.

[3] Karakaya, M., Korpeoglu, I., *et al.*, "Free Riding in Peer-to-Peer Networks," *IEEE Internet Computing,* vol. 13, pp. 92-98, 2009.

[4] R. Krishnan, M. D. Smith, Z. Tang, and R. Telang, "The Impact of Free-Riding on Peer-to-Peer Networks," in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004, pp. 70199c-70199c.

[5] L. Ramaswamy and L. Liu, "Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems," in *36th Annual Hawaii International Conference on System Sciences (HICSS'03)* 2003, pp. 220-220.

[6] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," presented at the *Proceedings of the 6th annual international conference on Mobile computing and networking, Boston, Massachusetts, United States*, 2000.

[7] Hao, Yang, Haiyun, Luo, *et al.*, "Security in Mobile Ad hoc Networks: Challenges and Solutions," *IEEE Wireless Communications,* vol. 11, pp. 38-47, 2004.

[8] H. Miranda and L. Rodrigues, "Friends and foes: preventing selfishness in open mobile ad hoc networks," in *23rd International Conference on Distributed Computing Systems Workshops* 2003, pp. 440-445.

[9] K. Balakrishnan, J. Deng, and V. K. Varshney, "TWOACK: preventing selfishness in mobile ad hoc networks," in *IEEE Wireless Communications and Networking Conference*, 2005, pp. 2137-2142 Vol. 4.

[10] Djenouri, D., Khelladi, L., *et al.*, "A Survey of Security Issues in Mobile Ad hoc and Sensor Networks," *IEEE Communications Surveys & Tutorials,* vol. 7, pp. 2-28, 2005.

[11] J. Mundinger and J.-Y. L. Boudec, "Reputation in self-organized communication systems and beyond," in *Proceedings from the workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems, Pisa, Italy*, 2006, p. 3.

[12] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: the dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad hoc networking (Ch 5)*, ed: Addison-Wesley Longman Publishing Co., 2001, pp. 139-172.

[13] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," presented at the *Second IEEE Workshop on Mobile Computing Systems and Applications, WMCSA* 1999.

[14] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *SIGCOMM Computer Communication Review,* vol. 24, pp. 234-244, 1994.

[15] Y. Yoo, S. Ahn, and D. P. Agrawal, "A credit-payment scheme for packet forwarding fairness in mobile ad hoc networks," presented at the *IEEE International Conference on Communications (ICC)*, 2005.

[16] L. Butty and J. P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *Mobile Network and Applications,* vol. 8, pp. 579-592, 2003.

[17] Y. Yoo and D. P. Agrawal, "Why does it pay to be selfish in a MANET?," *Wireless Communications, IEEE,* vol. 13, pp. 87-97, 2006.

[18] K. Mandalas, D. Flitzanis, G. F. Marias, and P. Georgiadis, "A survey of several cooperation enforcement schemes for MANETs," presented at the *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology* 2005.

[19] L. Shaohe, W. F. Xiaodong, Z. Xin, and Z. Xingming, "Detecting the Sybil Attack Cooperatively in Wireless Sensor Networks," presented at the *International Conference on Computational Intelligence and Security, CIS '08*, 2008.

[20] J. R. Douceur, "The Sybil Attack," presented at the *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

[21] K. Hoeper and G. Gong, "Bootstrapping Security in Mobile Ad Hoc Networks Using Identity-Based Schemes," in *Security in Distributed and Networking Systems,* , ed: book chapter in "Security in Distributed and Networking Systems", published by World Scientific Publishing Co., in the book series on "Computer and Network Security.", 2007.

[22] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil Attack In Sensor Networks: Analysis & Defences," presented at the *Third International Symposium on Information Processing in Sensor Networks (IPSN'04)* 2004.

[23] B. Parno and A. Perrig, "Challenges in securing vehicular networks," *Workshop on Hot Topics in Networks (HotNets-IV),* 2005.

[24] A. Tangpong, G. Kesidis, H. Hung-yuan, and A. Hurson, "Robust Sybil Detection for MANETs," in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks ICCCN* 2009, pp. 1-6.

[25] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol," presented at the *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, Lausanne, Switzerland,* 2002.

[26] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," in *Proceedings of P2PEcon, Harvard University, USA*, 2004.

[27] A. A. Pirzada, C. McDonald, and A. Datta, "Performance Comparison of Trust-Based Reactive Routing Protocols," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 695-710, 2006.

[28] U. Banerjee and A. Swaminathan, "Taxonomy of Attacks and Attackers in MANETs," *International Journal of Research and Reviews in Computer Science (IJRRCS)*, vol. 2 pp. 437-442, April 2011.

[29] P. Joshi, "Security Issues in Routing Protocols in MANETs at Network Layer," *Procedia Computer Science*, vol. 3, pp. 954-960, 2011.

[30] R. Sheikh, M. S. Chandel, and D. K. Mishra, "Security Issues in MANET: A review," in *Seventh International Conference On Wireless And Optical Communications Networks (WOCN)* 2010, pp. 1-4.

[31] M. Srivatsa, "Who is Listening? Security in Wireless Networks," presented at the *International Conference on Signal Processing, Communications and Networking, ICSCN India*, 2008.

[32] J. Merwe, D. Dawoud, and S. McDonald, "A survey on peer-to-peer key management for mobile ad hoc networks," *ACM Computing Surveys*, vol. 39, p. 1, 2007.

[33] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, *et al.*, "A performance comparison of multi-hop wireless ad hoc network routing protocols," presented at the *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, Texas, United States*, 1998.

[34] S. Mohseni, R. Hassan, A. Patel, and R. Razali, "Comparative Review Study of Reactive and Proactive Routing Protocols in MANETs," in *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST)* 2010, pp. 304-309.

[35] Z. J. Haas, M. R. Pearlman, and P. Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," *IETF Internet Draft, draft-ietf-manet-zone-zrp-04.txt*, July 2002.

[36] J. Kim and G. Tsudik., "SRDP: securing route discovery in DSR," in *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous 2005.* , 2005, pp. 247-258.

[37] S. Kannan, T. Kalaikumaran, S. Karthik, and V. P. Arunachalam, "A Study on Various Attacks and Attack Detection Methods in Mobile Ad-Hoc Networks," *International Journal of Signal System Control and Engineering Application*, vol. 3, pp. 34-39, 2010.

[38] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, 2002.

[39] Y. Chaba, Y. Singh, and P. Rani, "Comparison of various passive distributed denial of service attack in mobile adhoc networks," in *Proceedings of the 9th WSEAS*

international conference on electronics, hardware, wireless and optical communications, *UK*, 2010, pp. 49-53.

[40] D. Djenouri, O. Mahmoudi, M. Bouamama, D. Llewellyn-Jones, *et al.*, "On Securing MANET Routing Protocol Against Control Packet Dropping," presented at the *IEEE International Conference on Pervasive Services* 2007.

[41] D. Djenouri and N. Badache, "On eliminating packet droppers in MANET: A modular solution," *Ad Hoc Networks*, vol. 7, pp. 1243-1258, 2009.

[42] L. Hogie, P. Bouvry, and F. Guinand, "An Overview of MANETs Simulation " *Electronic Notes in Theoretical Computer Science*, vol. 150, pp. 81-101, 2006.

[43] The Network Simulator. ns-2. http://www.isi.edu/nsnam/ns/index.html.

[44] The CMU Monarch Project's Wireless and Mobility Extensions to NS. http://www.monarch.cs.rice.edu/.

[45] Cygwin : http://www.cygwin.com/.

[46] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators," presented at the *Proceedings of the second ACM international workshop on Principles of mobile computing*, *Toulouse, France*, 2002.

[47] L. Butty and J. P. Hubaux, "Enforcing service availability in mobile ad-hoc WANs," presented at the *The 1st ACM international symposium on Mobile ad hoc networking \& computing*, *Boston, Massachusetts*, 2000.

[48] Y. Wang, F. Wang, C. Rong, and S. Wang, "SPM: A Security Policy Based on Micropayment in Ad Hoc Networks," presented at the *22nd International Conference on Advanced Information Networking and Applications - Workshops (AINAW)*, 2008.

[49] E. Carrara and G. Hogben, "Reputation-based Systems: A Security Analysis," *ENISA Position Paper No. 2, October* 2007.

[50] A. A. Pirzada and C. McDonald, "Establishing trust in pure ad-hoc networks," presented at the *Proceedings of the 27th Australasian conference on Computer science - Volume 26*, *Dunedin, New Zealand*, 2004.

[51] X. Li, M. R. Lyu, and J. Liu, "A trust model based routing protocol for secure ad hoc networks," presented at the *IEEE Proceedings of Aerospace Conference*, 2004.

[52] V. Vasantha and D. Manimegalai, "Mitigating Routing Misbehaviors Using Subjective Trust Model in Mobile Ad Hoc Networks," presented at the *International Conference onComputational Intelligence and Multimedia Applications*, 2007.

[53] E. Friedman, P. Resnick, and R. Sami, "Ch: 27- Manipulation-Resistant Reputation Systems," in *Algorithmic Game Theory*, N. Nisan, *et al.*, Eds., ed New York: Cambridge University Press, 2007, pp. 677-697.

[54] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehaviour in mobile ad hoc networks," presented at the *Proceedings of the 6th annual international*

conference on Mobile computing and networking, Boston, Massachusetts, United States, 2000.

[55] S. Buchegger and J.-Y. L. Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks," in *Proceedings of WiOpt `03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, France*, 2003.

[56] L. Kejun, D. Jing, K. V. Pramod, and B. Kashyap, "An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in MANETs," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 488-502, 2007.

[57] L. Zhao and J. G. Delgado-Frias, "MARS: Misbehavior Detection in Ad Hoc Networks," presented at the *IEEE Global Telecommunications Conference (GLOBECOM)* 2007.

[58] K. Graffi, P. S. Mogre, M. Hollick, and R. Steinmetz, "Detection of Colluding Misbehaving Nodes in Mobile Ad Hoc and Wireless Mesh Networks," presented at the *IEEE Global Telecommunications Conference (GLOBECOM)* 2007.

[59] S. Buchegger, C. Tissieres, and J.-Y. L. Boudec, "A Test-Bed for Misbehavior Detection in Mobile Ad-hoc Networks " How Much Can Watchdogs Really Do?," presented at the *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, 2004.

[60] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A Survey of Attack and Defense Techniques for Reputation Systems," *ACM Computing Surveys*, vol. 42, pp. 1-31, 2009.

[61] Y. Po-Wah and C. J. Mitchell, "Reputation methods for routing security for mobile ad hoc networks," presented at the *Joint First Workshop on Mobile Future and Symposium on Trends in Communications, 2003. SympoTIC '03.* , 2003.

[62] A. Rossi and S. Pierre, "Collusion-resistant reputation-based intrusion detection system for MANETs," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 9 pp. 8-14, November 2009.

[63] B. Qureshi, M. Geyong, and D. Kouvatsos, "Collusion Detection and Prevention with FIRE+ Trust and Reputation Model," in *10th IEEE International Conference on Computer and Information Technology (CIT)* 2010, pp. 2548-2555.

[64] M. Seredynski and P. Bouvry, "Trust management for collusion prevention in mobile ad hoc networks," in *IEEE GLOBECOM Workshops (GC Wkshps)*, 2010, pp. 508-513.

[65] D. McCoy, D. Sicker, and D. Grunwald, "A Mechanism for Detecting and Responding to Misbehaving Nodes in Wireless Networks," in *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks SECON.* , 2007, pp. 678-684.

[66] J. Hu and M. Burmester, "LARS: a locally aware reputation system for mobile ad hoc networks," presented at the *Proceedings of the 44th annual Southeast regional conference, Melbourne, Florida*, 2006.

[67] S. Bansal and M. Baker. Observation-based Cooperation Enforcement in Ad Hoc Networks, Stanford University Technical Report, July 2003. Available at:http://arxiv.org/pdf/cs.NI/0307012.

[68] E. Friedman and P. Resnick, "The Social Cost of Cheap Pseudonyms," *Journal of Economics & Management Strategy*, vol. 10, pp. 173-199, 2001.

[69] B. N. Levine, C. Shields, and N. B. Margolin, "A Survey of Solutions to the Sybil Attack," Technical Report 2006-052, University of Massachusetts Amherst, Amherst, MAOctober 2006.

[70] N. B. Margolin and B. N. Levine, "Quantifying Resistance to the Sybil Attack," presented at the *Financial Cryptography and Data Security*, 2008.

[71] V. A. Luis, B. Manuel, and L. John, "CAPTCHA: Using Hard AI Problems For Security," presented at the *Proceedings of Eurocrypt*, 2003.

[72] S. Capkun, J. P. Hubaux, and L. Buttyan, "Mobility helps peer-to-peer security," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 43-51, 2006.

[73] S. Hashmi and J. Brooke, "Authentication Mechanisms for Mobile Ad-Hoc Networks and Resistance to Sybil Attack," in *Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE* 2008, pp. 120-126.

[74] H. Luo, P. Zerfos, J. Kong, S. Lu, *et al.*, "Self-Securing Ad Hoc Wireless Networks," in *The 7th IEEE Symposium on Computers and Communications (ISCC'02)*, 2002, p. 567.

[75] Z. J. Haas and L. Zhou, "Securing ad hoc networks," *IEEE Network*, vol. 13, pp. 24-30, 1999.

[76] C. Srdjan, B. Levente, and H. Jean-Pierre, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, pp. 52-64, 2003.

[77] I. K. Eltahir, "The Impact of Different Radio Propagation Models for Mobile Ad hoc NETworks (MANET) in Urban Area Environment," in *The 2nd International Conference on Wireless Broadband and Ultra Wideband Communications* 2007, pp. 30-30.

[78] Z. Sheng, L. Li, L. Yanbin, and Y. Richard, "Privacy-Preserving Location based Services for Mobile Users in Wireless Networks," Department of Computer Science, Yale University, Technical Report ALEU/DCS/TR-1297, 2004.

[79] M. Demirbas and Y. Song, "An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks," presented at the *Proceedings of the International Symposium on World of Wireless, Mobile and Multimedia Networks*, 2006.

[80] J. Wang, G. Yang, Y. Sun, and S. Chen, "Sybil Attack Detection Based on RSSI for Wireless Sensor Network," presented at the *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom'07)*, 2007.

[81] Yunxin, Li, and X. Huang, "The Simulation of Independent Rayleigh Faders," *IEEE Transactions on Communications* vol. 50, pp. 1503-1514, 2002.

[82] T. Suen and A. Yasinsac, "Peer identification in wireless and sensor networks using signal properties," presented at the *Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems* 2005.

[83] H. Jeffrey and B. Gaetano, "Location Sensing Techniques," *UW CSE Technical Report*, 2001.

[84] B. Xiao, B. Yu, and C. Gao, "Detection and Localization of Sybil Nodes in VANETs," presented at the *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks, Los Angeles, CA, USA*, 2006.

[85] P. Resnick and R. Zeckhauser, "Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System," in *The Economics of the Internet and E-Commerce (Advances in Applied Microeconomics, Volume 11)*, Michael R. Baye ed: Emerald Group, 2002, pp. 127-157.

[86] F. Li and J. Wu, "Mobility Reduces Uncertainty in MANETs," in *26th IEEE International Conference on Computer Communications. INFOCOM*, 2007, pp. 1946-1954.

[87] Feng, Li, Jie, and Wu, "Uncertainty Modeling and Reduction in MANETs," *Mobile Computing, IEEE Transactions on*, vol. 9, pp. 1035-1048, 2010.

[88] M. Grossglauser and D. N. C. Tse, "Mobility Increases the Capacity of Ad hoc Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 477-486, 2002.

[89] J. Crowcroft, R. Gibbens, F. Kelly, and S. Östring, "Modelling Incentives for Collaboration in Mobile Ad hoc Networks," *Performance Evaluation*, vol. 57, pp. 427-439, 2004.

[90] D. R. Figueiredo, M. Garetto, and D. Towsley, "Exploiting Mobility in Ad-Hoc Wireless Networks with Incentives," Technical Report CMPSCI–04–66, University of Massachusetts, Computer Science Department, Amherst, Massachusetts, USA, July 2004.

[91] H. Yang, J. Shu, X. Meng, and S. Lu, "SCAN: Self-organized Network-layer Security in Mobile Ad hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 261-273, 2006.

[92] E. Altman, A. A. Kherani, P. Michiardi, and R. Molva, "Non-cooperative Forwarding in Ad-hoc Networks," INRIA, Sophia-Antipolis, France, Report no. 5116, February 2004.

[93] S. R. Zakhary and M. Radenkovic, "Reputation-based security protocol for MANETs in highly mobile disconnection-prone environments," in *Seventh International Conference on Wireless On-demand Network Systems and Services (WONS)* 2010, pp. 161-167.

[94] G. Resta and P. Santi, "An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks," presented at the *The second ACM international workshop on Principles of mobile computing, Toulouse, France*, 2002.

[95] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, pp. 257-269, 2003.

[96] T. Suen and A. Yasinsac, "Ad hoc network security: peer identification and authentication using signal properties," presented at the *Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop (IAW '05) New York*, 2005.

[97] M. S. Bouassida, G. Guette, M. Shawky, and a. B. Ducourthial, "Sybil Nodes Detection Based on Received Signal Strength Variations within VANET " *International Journal of Network Security*, vol. 8, pp. 322-333, May 2009.

[98] Sun SPOT (Sun; Small Programmable Object Technology). Available: http://www.sunspotworld.com/

[99] C. Chaudet, D. Dhoutaut, and I. G. Lassous, "Performance Issues With IEEE 802.11 in Ad hoc Networking," *IEEE Communications Magazine*, vol. 43, pp. 110-116, 2005.

[100] S. Kumar, V. S. Raghavan, and J. Deng, "Medium Access Control Protocols for Ad hoc Wireless Networks: A Survey," *Ad Hoc Networks*, vol. 4, pp. 326-358, 2006.

[101] A. Parameswaran, M. I. Husain, and S. Upadhyaya, "Is RSSI a Reliable Parameter in Sensor Localization Algorithms: An Experimental Study," in *Field Failure Data Analysis Workshop (F2DA'09), New York*, 2009.

[102] H. Bidgoli, "The Internet Encyclopedia, Volume 3," *John Wiley & Sons, Inc. p. 127.*, 2004.