Decentralised Soft-Security in Distributed Systems

Paul David Kiddie

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Electronic, Electrical and Computer Engineering
College of Engineering and Physical Sciences
The University of Birmingham
February 2011

ABSTRACT

Existing approaches to intrusion detection in imperfect wireless environments employ local monitoring, but are limited by their failure to reason about the imprecise monitoring within a radio environment that arises from unidirectional links and collisions. This compounds the challenge of detecting subtle behaviour or adds to uncertainty in the detection strategies employed.

A simulation platform was developed, based on the Jist/SWANS environment, adopting a robust methodology that employed Monte-Carlo sampling in order to evaluate intrusion detection systems (IDS). A framework for simulating adversaries was developed, which enabled wormholes, black holes, selfishness, flooding and data modification to be simulated as well as a random distribution thereof.

A game theoretic inspired IDS, sIDS, was developed, which applied reasoning between the detection and response components of a typical IDS, to apply more appropriate local responses. The implementation of sIDS is presented within the context of a generic IDS framework for MANET. Results showed a 5-15% reduction in false response rate compared to a baseline IDS over a number of attacking scenarios.

sIDS was extended with immune system inspired features, namely a response over multiple timescales, as employed by the innate and adaptive components of the immune system, and the recruitment of neighbouring agents to participate in a co-ordinated response to an intrusion. Results showed a true response rate of 95-100% for all simulated attack scenarios. For random misbehaviour and assisted black hole scenarios, PDR gains of up to 30% and 15% were observed respectively compared to the pure game theoretic approach, tracking the omniscient network performance in these scenarios.

In all, this study has shown that applying game theoretic reasoning to existing detection methods results in better discrimination of benign nodes from adversaries, which can be used to bias network operation towards the benign nodes. When fused with immune system inspired features, the resulting IDS maintained this discrimination whilst substantially reducing attack efficacy.

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor, Dr. Costas Constantinou, for all the support and advice he has given over the course of the PhD, the many thought-provoking discussions we've had and for reading through the thesis drafts.

I would also like to thank all the past and present members of the Distributed Systems and Networks Laboratory, especially David, Keita, Amalia, Hani, Debra and Yuri for those beer breaks at Staff House!

Then to my family: Mum and Dad, thanks for encouraging my passion in computing as a child and providing continuous love and encouragement, especially throughout this process. I wouldn't have made it this far without you. To Karl, Neil, Mark and Tara, thanks for always being there for me and taking an interest in my work, but also for taking my mind off it when I needed a recharge, whether it was the quick chat on the phone, or the quick online game. Thanks to Gwyn, John, Val, Helen, Andy, Jayne, Clive, Glyn, Pauline and Nan – you each played a part in inspiring me to get this far, and your continuous emotional support has been invaluable.

Special thanks go out to Sarah, James, Mark, Deborah and Dan, for being there when I needed my mind taken off the pressures of the PhD and thesis-writing.

Finally, I would like to thank my partner Rhiannon for supporting me at every step and for that I am eternally grateful. She also took on the (not insignificant) task of proof reading numerous drafts of my thesis.

Thanks to the EPSRC, whom this research was funded by, and to the team at the Centre for Learning, Innovation and Collaboration for providing employment in some of their exciting projects during the writing up process.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AIS	Artificial Immune System
AODV	Ad-Hoc On-Demand Distance Vector
API	Application Programmers Interface
CA	Certificate Authority
CBR	Constant Bit Rate
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DC	Dendritic Cell
DoS	Denial of service
DSR	Dynamic Source Routing
FSM	Finite State Machine
GSM	Global System for Mobile communications
HIS	Human Immune System
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPS	Intrusion Prevention System
Jist	Java In Simulation Time
LCP	Linear Complementarity Problem
LH	Lemke Howson
LP	Linear Program

MAC	Medium Access Control
MANET	Mobile Ad-Hoc Network
NE	Nash Equillibrium
P2P	Peer-to-peer
PAMP	Pathogen Associated Molecular Pattern
PD	Prisoners Dilemma
PDR	Packet Delivery Ratio
PPAD	Polynomial Parity Arguments on Directed graphs
RERR	Route Error
RFC	Request For Comments
RNG	Random Number Generator
RREP	Route Reply
RREQ	Route Request
sIDS	soft-Intrusion Detection System
SNS	Self Non Self
SWANS	Scalable Wireless Network Simulator
TCP	Transmission Control Protocol
TFT	Tit For Tat
TLR	Toll Like Receptor
TTL	Time-To-Live
VANET	Vehicular Ad-Hoc Network
WEP	Wired Equivalent Privacy
WPA	Wireless Protected Access

CHAPTER 1

INTRODUCTION

A wireless peer to peer (P2P) network is a wireless network that allows members to communicate without requiring any existing infrastructure. Members use each other as routers to in order to reach hosts that are not in direct communication range. The earliest such networks were Packet Radio networks (PRNET) (Jubin & Tornow, 1987), a project concerned with developing them for military applications.

The interest in wireless P2P networks has increased greatly in recent years, brought about by the ubiquity of mobile computing and commoditisation of wireless technologies, such as 802.11 and Bluetooth, which means that most mobile devices are now equipped with at least one short-range wireless technology. Furthermore, with their independence from any form of infrastructure, P2P networks provide flexibility, which means they are well suited to a variety of applications, from large networks covering a battlefield to smaller, spontaneous networks set up for sharing data at business meetings. Practical applications of P2P networks are emerging, including wireless sensor networks for use in smart home-/office environments for monitoring energy consumption, and the dissemination of safety information to road users by vehicle to vehicle (V2V) or vehicle to roadside infrastructure (V2I) communications in Vehicular Networks (VANET). Wireless mesh networks have also been deployed in a number of cities worldwide to provide Internet access to its citizens.

The key feature of each of these examples of P2P network is the ability to *self-organise,* which is fundamental for their operation and is provided by a number of *decentralised* services that are responsible for routing and addressing, as well as other critical services. These services are the source of much research activity and pose a large number of research challenges. This thesis is concerned with **Mobile Ad-hoc Network (MANET) security**, an active area of research that poses significant challenges since these networks rely on the co-operation of all participating nodes in order to survive.

1.1 The MANET security problem

MANET security is a significant problem due to the relative ease that an intruder can eavesdrop on any wireless traffic in its neighbourhood. This is in direct conflict to the security attributes that are a must for most network applications: *availability*, *confidentiality*, *integrity*, *authentication* and *non-repudiation* (Zhou & Haas, 1999). Whilst access control techniques such as Wired Equivalent Privacy (WEP) or Wireless Protected Access (WPA) can prevent casual eavesdropping by outsiders, there is as much of a threat of insider attacks from compromised hosts. Insider attacks are a risk in hostile environments where a lack of physical protection means hosts are susceptible to capture and re-purposing by the adversary. Intrusion detection can assist in identifying the sources of insider attacks and responding appropriately. In these security-critical applications, multiple layers of security should be employed to reduce the overall attack surface.

Typical security approaches suited for infrastructure networks such as firewalls and centralised certification authorities (CAs) are not appropriate for MANET as they do not scale with a MANET's dynamic membership and they present a single point of failure that adversaries are likely to target in order to succeed in disabling the network. Thus, the services that run on ad-hoc networks should be fully distributed to promote high *availability* (Zhou & Haas, 1999). Tamper-proof hardware (Buttyan & Hubaux, 2003) or the use of a trusted platform module (TPM) (Shi & Perrig, 2004; Hao *et al.*, 2006; Yan, 2006) have been suggested as ways to prevent reprogramming or enable neighbours to verify remotely that a node has not been reprogrammed, but Intrusion Detection Systems (IDS) should always be considered as a precautionary measure in a security-critical environment. Numerous distributed approaches to the problem of insider security have been proposed in the literature, including secure routing, (Hu *et al.*, 2005; Sanzgiri *et al.*, 2002a; Patwardhan *et al.*, 2005; Acs *et al.*, 2004) misbehaviour detection, (Huang & Lee, 2003; Mishra *et al.*, 2004) cooperation enforcement (Buttyan & Hubaux, 2000, 2003; Michiardi & Molva, 2002; Buchegger & Le Boudec, 2002b) and the use of incentives (Buttyan & Hubaux, 2003). Such schemes employ local monitoring to confirm neighbours behave as expected but they are limited by their failure to reason about the imperfect monitoring possible within a radio environment that arises from unidirectional links and collisions. This often results in false detections and reduces the quality of the consequent responses, leading to retaliations to benign nodes.

1.2 Research Questions

This thesis studies the challenges of intrusion detection in imperfect wireless environments. The sources of uncertainty in wireless environments that are of direct relevance to intrusion detection are investigated. A set of local and distributed attacks of varying subtlety is implemented and run in fading environments to demonstrate the impact an imperfect wireless environment can have on detector output.

The main question posed in this thesis is if *game theory and immune system inspired approaches can be applied to improve the quality of intrusion detection in MANET.* In particular, we are interested in whether these techniques be used to reduce the number of responses toward benign nodes and improve the detection rates of subtle misbehaviour. Locally gathered information is used to build up a historical behavioural profile of neighbours that represents their ability to perform a given function (e.g. forwarding or routing) and is used within both approaches. Previous work in misbehaviour detection (Huang & Lee, 2003; Mishra *et al.*, 2004) gathers local information but fails to reason about the uncertainties in this information, either because the misbehaviour is subtle, it cannot be detected by local detection alone, or relies on detection schemes that are inherently imperfect. Another issue this thesis addresses is *if end-to-end network performance can be improved by employing these decentralised approaches.* If game theory and immune system inspired approaches can deliver improvements in intrusion detection, can this be translated into an advantage in terms of common metrics such as Packet Delivery Ratio (PDR) that benefits the community of benign (or, well behaved) nodes?

1.3 Contributions

The first contribution of this thesis is the development of a framework for simulating a diverse range of misbehaviour that targets Ad-Hoc On-Demand Distance Vector (AODV), a MANET routing protocol. Noteworthy is a number of interesting variations of the classical black hole attack that employs route invasion techniques to make significant gains in attack efficacy for low ratios of adversaries in the network.

Next, a general, modular intrusion detection system for MANET is presented, which provides a framework to formulate the baseline and game theoretic inspired IDS. The generic IDS presented can be extended to detect other attacks by simply adding to the set of detectors to provide continual improvements to the measure of a node's behaviour.

A key contribution of this thesis is the development of *sIDS*, a game theoretic inspired IDS that applies reasoning between the detection and response components of a typical IDS. The additional reasoning enables the IDS to determine whether to respond given the evidence presented whilst being aware of the imperfections implicit in the detection strategies used. This approach delivers improvements in true response rates for most studied scenarios.

The final contribution is *sIDS+AIS*, an extension of the game theoretic approach with a number of immune system inspired features to initiate co-ordinated responses over multiple timescales. This improves the end-to-end performance of benign nodes whilst keeping the earlier advantage of a high base response rate.

1.4 Organisation of thesis

The thesis is organised as follows:

- Chapter 2 provides a background to MANET and ad-hoc routing protocols, identifying the attack surface exposed by them and motivating the need for intrusion detection systems. It introduces the fundamental concepts of game theory and the Human Immune System (HIS) and reviews the existing approaches to intrusion detection in the literature, focussing on those that apply game theory and immune system analogies to the area of intrusion detection.

- Chapter 3 provides the simulation methodology employed, including a brief evaluation and validation of Jist/SWANS versus a number of other network simulators. It also provides the performance metrics upon which the discussion of further chapters has been based.

- In Chapter 4, a number of attacks on ad-hoc networks are described and an adversary framework is formulated suitable for incorporation into Jist/SWANS. A small case study is presented on a number of route invasion techniques that increase attack efficacy.

- In Chapter 5 a general MANET IDS is described, which forms the basis of successive IDS implementations, detailing the common services and detection techniques used to detect the attacks described in Chapter 4. This chapter also presents the transformation of detector notifications into metrics suitable for incorporation into a compact measure of a neighbour's behaviour.

- Chapter 6 presents *sIDS*, a MANET IDS that employs game theoretic reasoning as a means to improve the response characteristics of a typical MANET IDS. The performance of *sIDS* is evaluated in simulations over a diverse range of scenarios employing local and distributed misbehaviour.

- Chapter 7 presents *sIDS+AIS*, a multi-layered IDS employing both game theoretic reasoning and immune system inspired mechanisms, and evaluates its performance in the same scenarios as for sIDS.

- Finally, in Chapter 8, the thesis is concluded with a discussion of the main findings and possible avenues for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter introduces the properties of MANETs and provides a detailed description of the operation of distributed routing protocols that run on member nodes, identifying a large attack surface and motivating the need for intrusion detection systems (IDS) for reliable and secure dissemination of data across the network. To help the following discussion, basic intrusion detection system terminology is introduced. Finally, current approaches to mitigate misbehaviour in MANET is reviewed, and the steps, if any, that these approaches take when reasoning and minimising uncertainties.

2.2 Background

2.2.1 Mobile Ad-Hoc networks

A Mobile Ad-Hoc Network (MANET) is a collection of wireless nodes that co-operate to form a multi-hop network in which members maintain connectivity to one another without the support of infrastructure (Figure 2.1). This connectivity is maintained when nodes move, roam in and out, or when environmental factors cause individual wireless link breaks or errors (Perkins & Bhagwat, 1994). Data flows are often routed over several hops to their final destination (Perkins & Bhagwat, 1994; Perkins, 1997). The flexibility that a multi-hop network offers means these networks are well suited to applications that require instant deployment such as disaster relief and battlefield communications (Broch *et al.*, 1998).

Ad-hoc routing protocols provide end-to-end connectivity between nodes and are commonly implemented as a decentralised service operated by each node in the network. They are designed to provide best effort connectivity even in the presence of mobility and an unreliable medium with a high error rate (Royer & Toh, 1999).

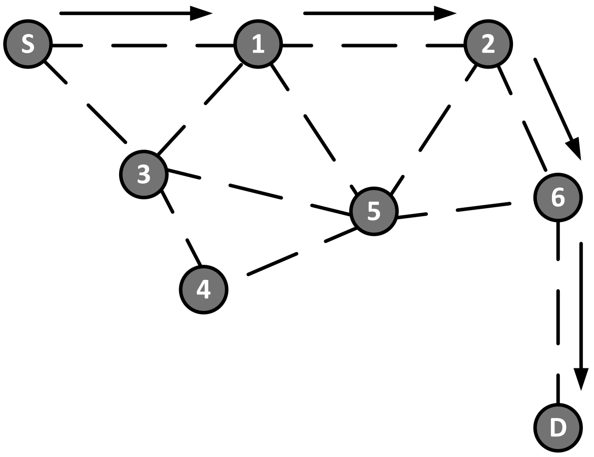


Figure 2.1: A collection of nodes forming a Mobile Ad-Hoc Network (MANET). The dashed lines show connectivity to immediate neighbours, and the arrows show the selected path of data flow between source S and destination D.

2.2.2 Ad-hoc routing protocols

Ad-hoc routing protocols are often categorised as *proactive*, *reactive* or *hybrid*. *Proactive* routing protocols (also known as *table-driven*) maintain a route table with entries to each node in the network and propagate changes to topology regularly to ensure each node maintains a complete and consistent view of the network topology (Royer & Toh, 1999), resulting in a "chatty" protocol with large routing tables. *Reactive* protocols (also known as *on-demand*) create, maintain and keep in memory only those routes currently in use, which minimises storage and computational requirements (Abolhasan *et al.*, 2004) but at the cost additional latency in data dissemination whilst routes are discovered. *Hybrid* routing protocols employ techniques from both proactive and reactive protocols. Abolhasan *et al.* (2004) and Perkins *et al.* (2001) illustrate the difference these properties can have on network performance over a range of topologies and traffic patterns.

Ad-Hoc On-Demand Distance Vector (AODV) (Perkins, 1997) and Dynamic Source Routing (DSR) (Johnson & Maltz, 1996) are two common *reactive* ad-hoc routing protocols which are under consideration for ratification by the Internet Engineering Task Force (IETF).

2.2.3 Reactive ad-hoc routing protocols

Reactive routing protocols such as AODV and DSR operate by running two procedures concurrently: *route discovery* and *route maintenance*. A source invokes *route discovery* when it wants to send data to a destination, creating a Route Request (RREQ) that is disseminated network-wide by neighbouring nodes. If the destination or a neighbour with a valid route to the destination receives the request, a Route Reply (RREP) is sent back to the source. When the source receives the reply, a route table entry is added and data packets for the destination are serviced and sent onwards to the appropriate next hop (Johnson & Maltz, 1996; Perkins, 1997). If the route discovery procedure times out and fails to discover a route, a suitable unreachable host message is sent to the application, and the packets for the corresponding unreachable destination are dropped.

The second procedure, *route maintenance*, monitors local connectivity to ensure routes are preserved by detecting link breaks between pairs of nodes (Johnson & Maltz, 1996; Perkins, 1997). Two methods are commonly employed in order to detect link breaks, which are periodic HELLO messages or from feedback directly from the link layer. HELLO messages are broadcast packets that are sent periodically to advertise a node of its presence. If a HELLO message is received by a neighbour, it assumes bidirectional communication to the sender. Link layer feedback is provided by the MAC protocol after it deems a frame as undeliverable, usually if multiple retransmissions of the frame are unsuccessful. The latter is more reactive to changes in topology and introduces no further overheads (Chakeres & Belding-Royer, 2005) but it requires specific hardware and driver support. If a link break is detected, *local repair* is often used as a first attempt to restore connectivity locally. This is where a request is sent with a small Time To Live (TTL) in order to find an alternative neighbour and is often transparent to the source. If local repair is unsuccessful, a Route Error (RERR) is sent to the source causing it to rediscover a route to the destination. Route maintenance is also involved in removing inactive routing entries to minimise the propagation of stale routing entries and to keep the size of routing tables down.

DSR and AODV use similar routing messages that have similar roles but there are important differences in the operation of each protocol. Firstly, the forwarding process is very different. DSR is a *source routing* protocol where the sender appends the full path to the destination onto the packet header that contains a list of intermediate nodes the packet should traverse (Johnson & Maltz, 1996). Intermediate nodes forward to the next hop by looking at the next hop entry in the path and sending the data onward. In contrast, AODV does not manipulate data packets, being a *distance vector* protocol based on the Bellman-Ford routing algorithm. Instead, the next hop decision is made at each intermediate node (Perkins *et al.*, 2001). Secondly, DSR caches multiple routes to a destination, permitting more redundancy during data transfer over AODV, which caches one route per destination. Lastly, DSR specifically states the use of link layer feedback (Johnson & Maltz, 1996) whilst AODV routers can be configured to send periodic HELLO messages or can use link layer feedback.

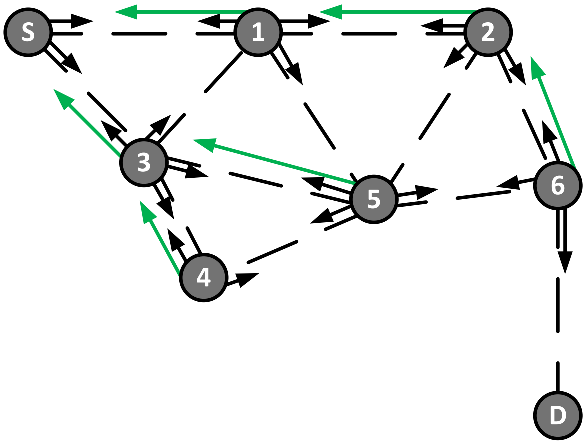


Figure 2.2: The dissemination of a route request from S (black arrows). Reverse route pointers (in green) are set up as the route request is broadcast, used when forwarding a route reply from D.

2.2.4 Ad-hoc On Demand Distance Vector (AODV)

This project employs the most recent Request For Comments (RFC) of AODV (July 2003), a *distance vector* protocol for MANETs. In order to understand the vulnerabilities exposed by AODV it is critical to understand the *route discovery* and *route maintenance* procedures thoroughly.

2.2.4.1 Route discovery procedure

The *route discovery* procedure for AODV is initiated when a source broadcasts a RREQ for a destination that it does not currently have a route for. This message is consequently broadcast by its neighbours, and is identified by a RREQ ID and originator IP address that prevents re-broadcasting and mitigates the effects of broadcast storms (Ni *et al.*, 1999), which can severely affect network performance.

As the RREQ is disseminated, each node adds a reverse route entry to the source via the neighbour it received the RREQ as shown in Figure 2.2, used to forward the Route Reply (RREP) generated either by the destination, or intermediate node with a “fresh enough” route to the destination. Each intermediate node that receives the RREP adds an entry to its routing table to the destination that is used during data forwarding. When the source receives the RREP, data packets are serviced for that destination by sending them onto the appropriate next hop.

The *routing table entry* (Table 2.1) lists the destination address, the next hop address, the number of hops to the destination and a route lifetime used to ensure that inactive routes are removed. A destination sequence number is also maintained for each destination, used to ensure routes are “fresh” and to guarantee loop-free operation of the protocol

Table 2.1: Excerpt of route table for S, showing a route entry to D via next hop 1

destination	nextHop	numHops	destSeqNum	lifetime (ms)
D	1	4	25	4000
2	1	2	20	8000

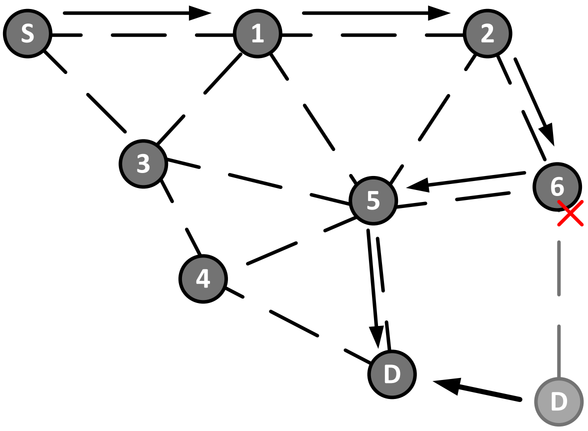


Figure 2.3: Route maintenance procedure. Destination D has moved out of range of node 6 and has become a neighbour of nodes 4 and 5. 6 invokes local repair and discovers route to D via 5, and data takes the new path highlighted.

(Perkins, 1997). Each node maintains their own sequence number allowing stale routing information to be discarded.

2.2.4.2 Route maintenance procedure

The *route maintenance* procedure is invoked if a node detects a link break between itself and a next hop which is actively being used to serve a route. Link breaks can be determined by keeping a count of broadcast messages received from a neighbour within a given time interval or through the use of a link layer notification (such as that provided by 802.11).

A node that detects a link break between itself and a next hop generates a RERR listing the unreachable destination(s). This message is disseminated to appropriate neighbours, either to invoke local repair of the affected part of the route, or as a last resort to force the source node to re-route. Once an appropriate route is discovered, queued data packets are once again serviced and sent via the new route (Figure 2.3).

2.2.5 MANET security challenges

MANETs expose a number of security vulnerabilities that span each layer of the protocol stack. An attacker can target the physical layer, as in the case of jamming (Xu *et al.*,

Table 2.2: A number of attacks MANET are susceptible to, with protocol layer they target and the original author who investigated it.

Attack	Directed at layer	Investigated in
Jamming	Physical	Xu *et al.* (2005)
Flooding	Network	Stajano & Anderson (2000)
Sleep deprivation	Routing	
Black hole	Routing	Wang *et al.* (1997)
Grey hole	Routing	Riedel (2003)
Wormhole	Routing	Hu *et al.* (2003a)
Rushing	Routing	Hu *et al.* (2003b)
Sybil	MAC/Network	Douceur (2002)
Spoofing	Routing	Vigna *et al.* (2004)
Route table overflow	Routing	Wang *et al.* (1997)
Selfishness	Routing	Michiardi & R. (2001)
	MAC	Kyasanur & Vaidya (2003)

2005), or the upper layers, by delaying or dropping packets to cause an an application that requires a high Packet Delivery Ratio (PDR), such as Voice over IP (VoIP) (Caro *et al.*, 2008) to fail. A selection of attacks on MANET are presented in Table 2.2. A large number of attacks these are directed at the routing protocol, targeting the unique characteristics of a MANET, such as the use of distributed algorithms and a shared, open wireless medium (Deng *et al.*, 2002). Member nodes often implicitly trust one another (true of classical MANET routing protocols), and attacks can occur by *outsiders* and *insiders* alike. An *outsider attack* occurs by an adversary that does not belong to the MANET, whilst an *insider attack* occurs by adversaries that belong to the network (Helsinki & Kärpijoki, 2000). A jamming attack is an effective but broad form of *outsider attack*; *insider attacks* are often more targeted and can be used to target specific nodes and network services.

To become an insider, the adversary could attack physical layer security protocols such as Wired Equivalent Privacy (WEP) or Wireless Protected Access (WPA) by a brute-force attack. But often, MANET nodes are left unattended with little or no physical protection, thus a preferred method is to physically capture member nodes, in the process acquiring the cryptography keys used to communicate to other members.

Attacks can be further classified as *active* or *passive*. An *active attack* is one launched outside the normal operation of the protocol by injecting false routing information or causing artificial error conditions on routes. In contrast, *passive attacks* are protocol compliant, completed as part of normal routing procedures. Additionally, *atomic misuses* are attacks encapsulated in a single routing message whilst *compound misuses* are a sequence of *atomic misuses* that may include protocol compliant routing messages (Ning & Sun,

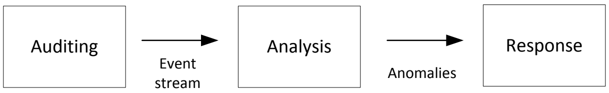


Figure 2.4: Interaction between modules of a network IDS

2003).

2.2.6 Network Intrusion Detection Systems (IDS)

Network IDS traditionally consist of several modules; an audit module, a detection module and a response module. The interplay between each module is shown in Figure 2.4. Network IDS can be:

- *Host-based*, in which each node performs intrusion detection and can be further *decentralised* if they communicate their detection outcomes to one another.
- *Hierarchical*, often employed in heterogeneous networking environments where device classes have computational and memory constraints which imposes a limit on their role in the overall IDS.
- *Fully centralised*, where a single device is responsible for providing IDS services for the entire network.

2.2.6.1 Auditing

Auditing is the initial process employed by an IDS in which an event stream is acquired from several sources. For a network based IDS, sources may include promiscuous notifications or a sampling of received and sent packets at the network layer. Network Security Monitor (NSM) (Heberlein *et al.*, 1990) first employed such notifications in wired networks and enabled clients to eavesdrop on the transmissions of neighbouring clients. In a MANET, this eavesdropping is trivial as member nodes share the wireless medium. When a network interface is set to promiscuous mode, nodes receive all messages in their reception range including those that are not directly addressed to them. This was exploited for IDS in MANET by Marti *et al.* (2000) in Watchdog and Pathrater, which uses promiscuous notifications to verify that a packet sent onward to a neighbour for forwarding was actually forwarded by that neighbour. Alternative approaches used to determine the packet forwarding ability of nodes are acknowledgments (Balakrishnan *et al.*, 2005; Liu *et al.*, 2007) and probing (Just *et al.*, 2003). Two-hop acknowledgments can be used to verify that data was both forwarded and received without error. Probing can verify a node is behaving as expected by the protocol (e.g. forwarding capability) by sending a control message and checking those generated in response are in line with expectations, but this approach assumes the target node should not be able to discriminate between a normal control message and a probing control message. In both cases, intrusion events can only be generated from those neighbours the monitoring node currently has a relationship with (*direct monitoring*), and both increase the control data load leading to additional energy expenditure.

Promiscuous notifications enable nodes to monitor the forwarding action of neighbours on flows that the monitoring node is both participating in and those it is not. In contrast, a node using an acknowledgment scheme can only monitor the flows it is participating in. Most MANET IDS including the Watchdog/Pathrater (Marti *et al.*, 2000) use eavesdropping as the basis for collecting audit data (Paul & Westhoff, 2002).

2.2.6.2 Analysis

Analysis is the process of detecting intrusions in audit data by the use of *misuse* detectors, *anomaly* detectors, or a combination of both. *Misuse detectors* use prior knowledge of known attack signatures to construct a finite set of rules, which are added to a rule base. Audit data is searched for signatures that correspond to one or more of these rules, and a match confirms an attack is in progress. Well defined misuse detectors are extremely accurate and are characterised by having a low false positive rate, but the rules are hard-coded and requires the IDS is constantly updated to deal with emerging threats. A simple misuse detector for a MANET could capture protocol violations, for example if a node originates an abnormal number of route requests in a given interval. Finite State Machines (FSMs) and Coloured Petri Nets are examples of a misuse detection IDS used within MANETs.

Anomaly detection uses statistical approaches to detect attacks and detectors are typically trained offline on audit data that corresponds to a normal profile. Once online, any deviation from the normal profile is then flagged as an anomaly. Anomaly detectors are more flexible than their misuse counterparts and can detect previously unknown attacks (Kemmerer & Vigna, 2002), but they generate a significant number of false detections depending on the specificity of the detectors and how representative the training set is of normalcy. Cross-Feature Analysis (Huang *et al.*, 2003), Genetic Algorithms (GA) (Sundararajan & Shanmugam, 2009), Neural Networks (Abraham *et al.*, 2007) and Artificial Immune Systems (AIS) (§2.4) are examples of anomaly detection schemes employed in MANETs.

2.2.6.3 Response

A *response* is the final action an IDS performs. Typical responses employed by MANET IDS may include elevating a suspicion level, disseminating evidence to a selection of trusted neighbours, or co-ordinating a response to isolate the suspect. Responses can occur locally, or over an extended area of the network. Majority voting techniques (Zhang & Lee, 2000; Paul & Westhoff, 2002) may be used to aggregate intrusion outcomes over an extended network area where there is uncertainty about whether a particular node is intruding based solely on local detection. The technique is susceptible to leading if the majority of reporting nodes are co-ordinated adversaries, possibly causing responses being taken against benign nodes.

2.3 Current approaches to providing security assurances in MANET

Table 2.2 showed a cross section of the attacks MANET are susceptible to. These have resulted in a corresponding large number of approaches that detect, mitigate or prevent such attacks from adversaries in MANET. These approaches are often categorised by five common security metrics; *availability*, *authentication*, *integrity*, *confidentiality* and *non-repudiation* and have been redefined by Zhou & Haas (1999) in the context of ad-hoc networks.

Typical fully centralised network IPS and IDS schemes for wired networks do not scale but instead introduce further weaknesses if directly applied to a MANET (Deng *et al.*, 2002). Centralised security services present a single point of failure (Dhillon *et al.*, 2004) likely to be targeted by the adversary. If the service is critical to the functioning of the network, say by authenticating nodes or certifying data and control packets, then disabling the security service will disable network communications entirely. If the service is distributed between a number of nodes the security mechanism is more resilient and the adversary has to target multiple agents, but distributing a security protocol is often challenging; it may require nodes to be synchronised and share state by passing intrusion reports, which itself could be targeted by the adversary. To fully meet the security needs of a MANET, a *defence-in-depth* solution (Figure 2.5) spanning multiple layers of the MANET protocol stack is often required (Yang *et al.*, 2004). A number of existing MANET security schemes which could be part of this *defence-in-depth* solution are now reviewed, focusing on the network and routing layer.

2.3.1 Secure Routing

Secure routing protocols such as aridane (Hu *et al.*, 2005), ARAN (Sanzgiri *et al.*, 2002b,a), SecAODV (Patwardhan *et al.*, 2005) and endairA (Acs *et al.*, 2004) are proactive MANET security schemes that secure the ad-hoc network from manipulation, replay or eavesdropping of routing messages by unauthorised nodes. Secure routing protocols employ a number of techniques to achieve this including hop-by-hop authentication of routing messages, Certification Authorities (CAs), frequent key revocation and encryption of routing packet headers, to assure *authentication*, *integrity* and *non-repudiation* of the routing protocol.

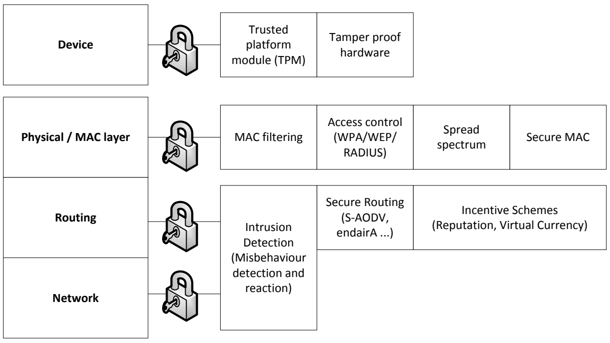

Device
Trusted platform module (TPM)
Tamper proof hardware
Physical / MAC layer
MAC filtering
Access control (WPA/WEP/ RADIUS)
Spread spectrum
Secure MAC
Routing
Network
Intrusion Detection (Misbehaviour detection and reaction)
Secure Routing (S-AODV, endairA ...)
Incentive Schemes (Reputation, Virtual Currency)

Figure 2.5: *Defence-in-depth* for MANET.

A CA may introduce a weak point in the ad-hoc network unless it is distributed as in the case of threshold cryptography (Luo *et al.*, 2005; Basile *et al.*, 2005), which allows the network to tolerate partial signatures originating from compromised CA nodes. However, like majority voting, there is a limit on the number of adversaries a threshold cryptography CA tolerates before the number of partial signatures generated by adversaries outweigh those generated by honest nodes.

2.3.2 Incentive Schemes

A common approach to promote *availability* is to use incentives to promote co-operation in MANETs. Incentives are artificial mechanisms that reward a node for co-operating with the protocol to stimulate the desired behaviour. They are typically used in environments where members belong to a number of competing authorities, each with conflicting objectives, or where each node is resource constrained, such as in energy or available throughput. Virtual currency and reputation schemes are commonly employed to stimulate co-operation when routing.

Virtual currency schemes (Buttyan & Hubaux, 2000, 2003) provide economic incentives to stimulate cooperation during MANET routing. Nodes maintain a budget that is used when they send their own packets and is replenished when they service data for others, hence rewarding contributing nodes. These schemes require each node be equipped with tamper-proof hardware to store the budget (Carruthers & Nikolaidis, 2005). Furthermore, routing messages are used directly when decrementing and replenishing a nodes budget, which could be manipulated by the intelligent adversary. The maximum budget a node can attain is based on the number of routes it serves, thus the schemes favour well-connected nodes.

For malicious adversaries, reputation schemes such as CORE (Michiardi & Molva, 2002) and CONFIDANT (Buchegger & Le Boudec, 2002b) can be employed. In these schemes, each node assigns a reputation to each neighbour it has communicated with. This reputation depends on observations made locally through the use of eavesdropping and in some cases, second-hand reports acquired from trusted neighbours. Reputation can be used to identify trustworthy neighbours to send data onward, or to make a decision whether to carry a neighbour's traffic. Neighbours with consistently low reputation are excluded from the network, which lessens the impact of the misbehaviour and should motivate the adversary to behave. CORE defines *functional reputation,* which allows a node to identify those neighbours with the highest reputation for a given network function, for example, packet forwarding. CORE allows excluded nodes (as a result of prior misbehaviour) to be gradually reintroduced should they be cooperative over a sustained period of time. Neither CORE or CONFIDANT studies the effect of imperfect observations on the computation of reputation and its effect on the overall fairness of the security protocol.

2.3.3 Watchdog/Pathrater

Watchdog/Pathrater (Marti *et al.*, 2000) is a host-based IDS which promotes network *availability* by allowing member nodes to verify the packet forwarding behaviour of neighbours by the use of promiscuous notifications. Each node keeps a rating per observed neighbour, which is increased if the chosen neighbour forwards data when it is asked to and the observing node receives a notification and is reduced otherwise. The Pathrater component uses these ratings to determine "stable" routes, preferring nodes with a higher rating. Marti *et al.* (2000) identify that collisions and asymmetric links may cause missed observations and corresponding false detections but in their scenario it seems to have minimal impact on network throughput. Pathrater does not exclude nodes with consistently low ratings from using the network and is not appropriate for providing incentives for selfish nodes (Buchegger & Le Boudec, 2005).

2.3.4 Finite State Machines (FSMs)

AODVSTAT (Vigna *et al.*, 2004) is a host-based IDS that ensures *availability* and *integrity* in MANET. Each node runs STAT (Vigna *et al.*, 2003), a framework that allows the modelling of attack scenarios and is preloaded with FSMs designed for the detection of misbehaviour in MANET. Each FSM detects a single type of attack, corresponding to a form of *misuse detection.* The FSMs transition from an initial idle state to a number of intermediate states which represent a snapshot of the security state, based on input audit data from promiscuous monitoring. Each FSM has a final ATTK state which signals an alert to invoke an appropriate response.

2.4 Artificial Immune Systems

2.4.1 Background

Artificial immune systems (AIS) are an Artificial Intelligence (AI) paradigm that includes Genetic Algorithms (GAs) and Neural Networks that are inspired by biological models. Specifically, AIS are inspired by the processes and interactions between the various cells of the immune system. The Human Immune System (HIS) is a multi-faceted, multi-layered intrusion prevention and intrusion detection system with three main lines of defence. The first is the epithelial surfaces (skin and mucosal linings) and the bodily fluids such as stomach acid, which provide *perimeter security* that prevent most pathogens from entering the body (Janeway *et al.*, 2004, p. 40). Should the pathogens pass this initial barrier, the *innate*, and later, the *adaptive* immune response takes over. The cells involved in the innate immune response are:

- *Innate leukocytes* that include natural killer cells (NK cells), mast cells, eosinophils and basophils.

- *Phagocytes* that include macrophages, neutrophils and dendritic cells.

These immune cells recognise pathogens through the activation of toll-like receptors (TLRs) on the immune cells surface that recognise pathogen associated molecular patterns (PAMPs), present only on pathogens (Janeway *et al.*, 2004, p. 51). Each immune cell is equipped with a subset of between ten and fifteen types of TLR (Janeway *et al.*, 2004; Doan *et al.*, 2007). The activation of one or more TLRs by a pathogen leads to the host immune cell taking one of the following immediate actions (Janeway *et al.*, 2004):

- if the host cell is *phagocytic* it engulfs and digests the recognised pathogen. Dendritic cells present the antigen of a digested pathogen as part of the preparation for an adaptive immune response, if one is required.

- secrete cytokines and chemokines that have a number of effects that shape the immune response, from triggering inflammation to recruiting and activating neighbouring immune cells.

- release cytokines that directly act on the pathogen. *Interferons* are antiviral agents that prevent the spread of viruses to uninfected cells (Janeway *et al.*, 2004, p. 87). *Tumour Necrosis Factor* (TNF) can cause the target cell to undergo apoptosis, or 'programmed cell death'.

Another important part of the innate immune system is *complement*, which is a system of plasma proteins that can mark a pathogen for destruction by phagocytes (Janeway *et al.*, 2004, p. 55) or otherwise enhance the immune response to the pathogen.

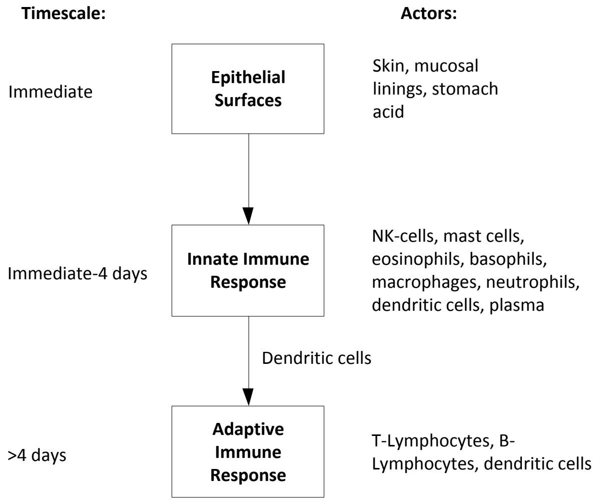

Timescale:
Actors:
Immediate
Epithelial Surfaces
Skin, mucosal linings, stomach acid
Immediate-4 days
Innate Immune Response
NK-cells, mast cells, eosinophils, basophils, macrophages, neutrophils, dendritic cells, plasma
Dendritic cells
>4 days
Adaptive Immune Response
T-Lymphocytes, B-Lymphocytes, dendritic cells

Figure 2.6: The layers of protection in the vertebrate immune system, illustrating the actors and timescales associated with each layer. Epithelial surfaces provide the first line of defence. If breached, the innate immune system is invoked immediately, whilst the adaptive immune response takes the order of a few days for effector cells to mature.

The adaptive immune response is induced when a pathogen is ingested by an immature dendritic cell and, upon activation, presents pathogenic antigens to T-lymphocytes in a nearby lymph node. These lymphocytes undergo a period of maturation called *negative selection.* This period of maturation explains the lag after the original infection (Janeway *et al.*, 2004, p. 12) before the adaptive immune response is invoked. Thus, dendritic cells, with their ability to sample the local environment and present non-self antigen to maturing lymphocytes, act at the interface of innate and adaptive immunity. This process is shown in Figure 2.6.

There are many examples in immunology literature that demonstrate the imperfect nature of the immune system and the evolution of pathogens to the threat of detection, each with shared analogues in network environments. Allergies can lead to systemic responses to common, harmless, non-self antigens that emerge as conditions such as asthma or hayfever (Janeway *et al.*, 2004). Autoimmunity, the process by which the immune system responds to host cells is rare, but can cause substantial damage to the host organism. Generally it is kept under control by a number of mechanisms, but if one or more of these is compromised by infection, genetic defects or environmental conditions, self-reactivity can occur. Autoimmunity is always a risk because of the subtle difference between *self* and *non-self* (Janeway *et al.*, 2004). Finally, pathogens have evolved ways to evade the immune response or to attack immune cells directly (Chapel *et al.*, 2006). For example, Herpes enters a state of latency and hides in sensory neurons in which it is undetectable, whilst Tuberculosis uses macrophages as its host and is immune to the bactericidal actions of the cell.

2.4.2 AIS models

Forrest & Hofmeyr (2001) was pivotal to the development of the AIS field and artificialised *negative selection*, the process by which lymphocytes mature and are checked for their auto-reactive properties. In Forrest & Hofmeyr (2001), a set of immature detectors are created and compared to a set of triples corresponding to network connections of a normal profile. If any immature detectors match those triples they deleted and replaced with another in the same way a lymphocyte would be deleted from the repertoire in the body. Any surviving detectors have a finite lifespan and are used to detect anomalies in the network traffic, raising an alarm if found. This is typical of the traditional view of the immune system proposed by Janeway *et al.*, 2004, which is that the immune system distinguishes foreign bodies (non-self) from host cells (self) by *self non-self recognition* (SNS).

A number of improvements to the original negative selection algorithm have been proposed. Dasgupta & Gonzalez (2002) proposes the use of GAs or neural networks to produce the immature detectors for providing more even coverage of self and non-self space, whilst Gonzalez & Dasgupta (2004) suggests the use of fuzzy detectors to provide overlap between self and non-self and be more analogous with nature. The weaknesses of the negative selection algorithm are discussed in Aickelin *et al.* (2003), who highlight concerns over the scalability of the algorithm for large detector sets, required to ensure few holes in detector coverage and minimise false negatives.

A more recent model is the Dendritic Cell Algorithm (DCA) (Greensmith *et al.*, 2006) inspired by the role of the dendritic cell in sampling and as a mediator of the adaptive immune system. In the DCA, dendritic cells are signal processing elements that take multiple input signals in the form of safe signals, danger signals and PAMPs (Aickelin & Cayzer, 2002) and produce a single output. This output is used to activate or suppress a lymphocyte and their ability to do so is based on their maturity just like their natural counterparts (Janeway *et al.*, 2004). Safe signals are defined to be an indicator of normal behaviour, danger signals are measures which increase in response to abnormal behaviour and a PAMP is a signature of abnormal behaviour. DCA is based on earlier work which highlights suitable analogies from Danger Theory (DT) (Matzinger, 2002), an alternative view on role of the immune system, which is that it is concerned with the recognition of and response to *dangerous* non-self. Harmless non-self does not cause a reaction.

2.4.3 AIS applied to ad-hoc networks

AIS approaches to IDS in ad-hoc networks tend to use one of the above approaches at their core. Balachandran *et al.* (2006) is a host-based IDS for nodes running DSR. Negative selection is used to train a detector set within a network environment free of misbehaviour, and the surviving detector set is used within the IDS of a live network. The initial training conflicts with the instant deployment ideal of MANET and it is also unrealistic to assume that the training set is completely free of all forms of misbehaviour, or that it is completely representative of normalcy. The effect of imperfectly observing the routing protocol event sequence on the training of detectors or the effect on detection is also not considered. SASHA (Bokareva *et al.*, 2005) also employs negative selection but uses neural networks to create the immature detector set. It is a hierarchical IDS that uses high powered monitoring sensors and PC class devices which collaborate to detect misbehaviour within a sensor network environment. Whilst this has the advantage of freeing sensor nodes from computationally demanding IDS duties, the hierarchical nature means the system is partially centralised. The privileged links between the monitoring sensors and PCs are likely to be the target of the more intelligent adversary.

Kim *et al.* (2006) presents a host-based IDS that applies the DCA (Greensmith *et al.*, 2006) to detect the Interest Cache Poisoning Attack in sensor networks. This is an attack on the Directed Diffusion routing protocol where an adversary disrupts data paths by injecting fake interest packets to replace entries in the interest cache of target nodes. The interest cache is similar to the routing table of typical MANET routing protocols. The signals are mapped onto appropriate observables, for example, the danger signal is mapped to the interest cache insertion rate. There is no consideration of the impact that an imperfect observation has on the calculation of the signals nor the effect this has on the detection statistics.

Lastly, there are hybrid approaches that employ both *negative selection* and analogies from Danger Theory to detect misbehaviour. Sarafijanovic & Le Boudec (2005) applies such an approach to detect misbehaviour in MANET running DSR. Events are generated from routing packets received or sent and are compressed into an event stream that lists the type of routing packet and its direction (sent or received). This is further compressed into a gene which corresponds to the number of times given events occur in sequence in a given interval, before being converted into a bit stream suitable for use in negative selection, in a format similar to Kim & Bentley (2001). The novelty is that detectors can be trained during MANET runtime and a virtual thymus allows the self set (a representation of the desired DSR behaviour) to change over time. A signal corresponding to packet losses is used to suppress or activate an individual response but as stated earlier, packet losses may arise from a number of network conditions that are not the result of misbehaviour (§2.2.6.1).

2.4.4 Discussion

The immune system is a multi-layered approach to security and incorporates a number of effective measures to eradicate infection; from physical barriers to innate and adaptive immunity. The autonomy of immune cells and co-ordination of a response through chemical signals provide much inspiration for a MANET IDS. Of course, the immune system is not subject to the same limitations as MANET nodes but nevertheless the high level features and processes are suitable for application to a MANET IDS.

2.5 Game Theory

2.5.1 Background

The field of game theory is considered to have begun in the 1940s with the publication of *The Theory of Games and Economic Behaviour* (Von Neumann & Morgenstern, 1970). This provided the mathematical basis for a diverse range of economics publications to follow that provided tools to construct and model numerous scenarios to study the interactions between a number of conflicting players (or agents), termed *non-cooperative* game theory.

In game theory, each player has a number of actions and a utility function which determines their preference for a given outcome. Through the application of *solution concepts* it can be determined what strategies players are likely to adopt in the game. The most widely used *solution concept* is the Nash Equilibrium (NE) (Nash, 1950). If each player plays to maximise his own payoffs then the player is *rational.*

The main primitive modelled in algorithmic game theory is the *game*, G, which defines the players, utility functions and set of actions for each player. A game can be represented in two forms. The first is a game in *normal form* and is commonly used where players choose actions simultaneously, or, alternatively, where players do not observe the moves made by the other players (Rasmusen, 2001). Such a game is represented as an n-dimensional matrix, where n corresponds to the number of players. Table 2.3 illustrates the normal form of a two-player Prisoners Dilemma, a classical and well-studied game in game theoretic literature.

Table 2.3: An illustration of a normal form game: The Prisoners Dilemma.

		Column	
		Deny	Confess
Row	Deny	-1,-1	-10,0
	Confess	0,-10	-8,-8

Formally, an n-person normal-form game is a tuple $G = (N, A, u)$ (Shoham & Leyton-Brown, 2009) where:

- N is the set of n players
- $A = A_1 \times ... \times A_n$, where A_i is a finite set of actions available to player i, and each vector $a = (a_1, ..., a_n) \in A$ is a strategy profile.
- $u = (u_1, ..., u_n)$ where $u_i : A \mapsto \Re$ is a real-valued utility (or payoff) function for player i.

The one-shot Prisoners Dilemma (PD) bears some interesting properties; immediately evident is that the equilibrium outcome $(Confess, Confess)$ may initially be seen to be an undesirable one, as it is intuitively worse for both players than $(Deny, Deny)$. By enumerating the process of finding the dominant-strategy equilibrium to find the players' best responses we find there is in fact no better outcome. If $Column$ were to choose *Deny* then the best response, leading to the greatest payoff for *Row*, would be to choose $Confess$, as 0 is greater than -1. Knowing this, $Column$, given *Row*'s choice of $Confess$ yields a choice of -10 for *Deny* and -8 for $Confess$, so it also chooses $Confess$. This leads to the equilibrium outcome (also a Nash Equilibrium) of $(Confess, Confess)$.

A game can also be represented in *extensive form*, similar to a game tree. Each node on the tree represents a decision to be made by one of the players and the edges from that node represent the action set available to the player at that node. As the game progresses, signals may be revealed that are observed by the opposing players. The leaf nodes represent outcomes to which players are awarded payoffs. In an *extensive form*, a player's strategy requires a decision at each of his choice nodes, regardless of whether or not it is possible to reach that node during the game (Shoham & Leyton-Brown, 2009).

Formally, an n-person extensive-form game is the tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ (Shoham & Leyton-Brown, 2009) where:

- N is the set of players
- A is the set of actions
- H is the set of nonterminal choice nodes
- Z is a set of terminal nodes, disjoint from H
- $\chi : H \mapsto 2^A$ is the action function, which assigns to each choice node a set of possible actions
- $\rho : H \mapsto N$ is the player function, which assigns to each non-terminal node a player $i \in N$ who chooses an action at that node

- $\sigma : H \times A \mapsto H \cup Z$ is the successor function, which maps a choice node and an action to a new choice node or terminal node
- $u = (u_1, ..., u_n)$, where $u_i : Z \mapsto R$ is a real-valued utility function for player i on the terminal nodes Z

A number of *solution concepts* exist that determine appropriate *strategy profiles* for the game and describe how it should be played. Basic solution concepts include Minimax (Von Neumann, 1928) or dominant strategy equilibria, which are straightforward to compute but only produce a solution for a subset of game types. For example, minimax only applies to *zero-sum* games (games whose payoffs add up to zero in each strategy combination) that are not typically found in realistic applications (Rasmusen, 2001). A *strategy profile* is the set of strategies for each player of the game, i.e. $S = s_i^*, ..., s_N^*$, and may be in terms of *pure* or *mixed* strategies. A pure strategy states that a player should play the given strategy with certainty, whilst a mixed strategy states the mixing probabilities that players should play their pure strategies by. A strategy for player i, s_i, is a rule which tells the player which action to choose at which each of his moves in the game.

Nash Equilibrium is a robust and widespread *solution concept*, and for any game at least one exists in pure or mixed strategies (Shoham & Leyton-Brown, 2009). NE tests that a player's strategy s_i is a best response to the other player's strategies s_{i-1}. If a player cannot benefit by changing his or her strategy whilst the strategies of other players are fixed then the proposed strategy profile $s^* = (s_i^* ... s_{i-1}^*)$ is a NE. This may be Pareto sub-optimal in that there may exist a profile where both players can do better; such a strategy profile is *Pareto optimal.* In the Prisoners Dilemma, the resulting NE $(Confess, Confess)$ is not Pareto optimal, but the Pareto optimal strategy profile $(Deny, Deny)$ is in incompatible with the NE; hence players are likely to deviate away from the Pareto optimal profile and toward the NE. Incentives or an uncertainty over when interactions will terminate can aid to curtail deviation and promote the desired behaviour. If multiple NE exist, *Pareto optimality* can be used as an *equilibrium refinement* to remove less desirable equilibria.

2.5.2 Game Types

Several game types exist which allow realistic scenarios to be captured in a game theoretic setting. *Repeated games* model frequent interactions or relationships between players over time and are represented as a given game being played multiple times (or in multiple stages) by the same set of players. Players observe the outcome of the game in the previous stages, which is stored in their history profile and is used to compute their *strategy profile* in the current stage. A player has *infinite horizon* if they can recall the history of all prior stages, but if this is bounded, the player has *finite horizon.*

A game can be repeated finitely or infinitely. Infinitely repeated games can be used if there is uncertainty over the interaction duration between players (Buttyan & Hubaux, 2007). Finitely repeated games are used when the duration is *a priori* known and can suffer from an 'unravelling' effect, where players use backward induction to determine dominant strategies that may be non-cooperative strategies, leading to undesirable outcomes from the modeller's perspective (Nachbar, 1992). This unravelling does not occur in infinitely repeated situations as it is not clear where the terminal payoffs are. In these situations, payoffs cannot be attached to terminal nodes so a *discount factor* is applied to future payoffs instead, representing the probability that the game will be stopped in that round (Shoham & Leyton-Brown, 2009). Thus, infinitely repeated games can lead to the emergence of co-operation which does not exist in finitely repeated games (Buttyan & Hubaux, 2007).

Bayesian games model situations where players are unsure of the rules of the game, for example, the precise game being played or the utility function of the other player. The utility function employed by the player may be different for each type of the player. In this case, the other players have a belief that the player is of a given type which is re-evaluated based on signals or observations made through game play.

2.5.3 Game Theory applied to ad-hoc networks

Common applications of game theory in ad-hoc networks have explained the rationale of selfishness that plagues decentralised algorithms in general (Feigenbaum & Shenker, 2002), demonstrating analogues between selfishness and the Prisoners Dilemma (PD). In the PD, both parties confess in the absence of binding commitments or side-payments and arrive at a solution which has a worse payoff than if they had cooperated. Similarly, a selfish node prefers to drop rather than forward a packet if there are no reprisals for doing as it benefits by conserving finite resources (such as energy). This is problematic if all nodes adopt this strategy, when packet forwarding ceases network-wide and the network becomes disconnected. Thus, the main issue is to stimulate cooperation by promoting the desirable, co-operative, *Pareto Optimal* outcome. Srinivasan *et al.* (2003) illustrates the emergence of co-operative behaviour by modelling selfishness as an infinitely repeated PD where players adopt the Generous Tit For Tat (GTFT) strategy, showing that the stable, *Pareto-optimal* NE corresponding to co-operation can be promoted without the use of incentives. GTFT is similar to Tit For Tat (TFT), a common and highly effective strategy used in game theory, but which gives a benefit of doubt to a player who confesses on occasion. Their network model is simple and abstracts the underlying topology away. In addition, they assume that all nodes know perfectly how many energy classes there are in the system and that selfish nodes send a positive or negative acknowledgment of their decision to forward to the requestor, which may not be the case in realistic scenarios. Buttyan & Hubaux (2007) also provides evidence that that by infinitely repeating their

Forwarders Dilemma (a game with the same payoff structure as the PD) an NE with cooperative properties emerges, even though a single stage game has defect as its only NE. Felegyhazi *et al.* (2003) investigates the effect of mobility on promoting co-operation but unlike Srinivasan *et al.* (2003), does not use energy as a cost, instead the source derives a benefit from the arrival of a packet at the destination. In static configurations, the conditions necessary for co-operation rarely exist, whilst in a more mobile network, the conditions become more feasible, and is further improved by increasing 'generosity' in their model. Jaramillo & Srikant (2007) uses the repeated PD to model selfishness and assesses the use of common and derived game theoretic strategies to promote co-operation in ad-hoc networks. TFT requires a perfect signal that a given packet was relayed, which is not possible due to collisions on the medium. GTFT requires a perfect estimate of the probability that a packet that was forwarded was not overheard by the originating node. Contrite TFT (CTFT) integrates reputation when determining whether to cooperate or defect. Nodes all begin in good standing and stay that way if they co-operate when CTFT specifies they should, but it also provides an opportunity for nodes in a bad standing to correct their error by cooperating in a single stage.

Bayesian games have also been used to model selfishness, (Urpi *et al.*, 2003) capturing the lossy nature of wireless communications. Each node belongs to one of a set of secret energy types, and payoffs are computed based upon a number of factors: a type dependent evaluation of their energy importance, the measure of energy spent with success and the ratio of sent packets compared to those it wanted to send. Typical of Bayesian games, nodes maintain a belief that a neighbour is of a given energy type, which provides the weights to the Bayes Nash Equilibrium (BNE). Other incentive schemes (described in §2.3.2) are applied to the model, and it is shown that co-operation can emerge provided that enough members of the network agree to do so and that members do not forward more traffic than they generate (introducing fairness).

A small number of proposals use game theory in the context of malicious behaviour. Alpcan & Basar (2003) develop a distributed IDS implemented on a network of sensors and an underlying cooperative N-player coalitional game that allows the IDS to switch between a number of security levels during runtime, which increases or decreases the sensitivity of security measures. Each sensor reports a security risk value for a given intrusion type and the Shapley value, a method used to analyse the output of coalitions (Shapley, 1953), aggregates the risk values and updates the security level should the output exceed a predefined threshold. Alpcan & Basar (2003) also develop an extensive game with incomplete information to model the behaviour, intent and target of adversaries and the response of the IDS. IDS nodes know the adversary has made a move but do not know the action; whether it attacked or did not. A mixed-strategy NE emerges in which the probability of the adversary attacking decreases as the false alarm cost to the IDS also decreases. How the adversary estimates the false alarm cost is uncertain, but this is an intuitive result as in the absence of a false alarm cost, the IDS given a choice of either detecting or not would always detect, and the adversary would always be caught in an ideal IDS.

Bayesian games are also used to model the interactions between malicious adversaries and IDS. Patcha & Park (2004) model these interactions in a repeated, zero-sum, 2-player Bayesian game based on a classic signalling game. The sender is either a *regular* or *attacker* type, and is private information. In both types the sender can *attack* or *not attack*. The receiver player is the IDS, has a single type and two actions: *detect* or *miss*. Beliefs are maintained for each interacting node and are updated after each game stage. Intuitively, the beliefs should converge over the stages of the game reducing the false alarm rates and increasing the IDS payoff. Like Alpcan & Basar (2003), if the false alarm cost is sufficiently low, then the IDS will always respond. Liu *et al.* (2006) use the same player definitions as Patcha & Park (2004), but players have additional terms representing energy cost for monitoring and attacking in their payoff function. The updated beliefs switch the IDS between lightweight and heavyweight monitoring systems to conserve energy when heavyweight monitoring is unnecessary, for example, when the majority of nodes in the network are deemed to be benign.

2.5.4 Discussion

Game theory provides a mathematical framework for studying the interactions of players under conflict. There are an abundance of game models and types that can accurately model common scenarios within MANET including repeated interactions between nodes, uncertainty about the behaviour of other players and imperfect information revelation and signalling. Most current applications deal with the problem of stimulating co-operation in the presence of selfish agents; few deal with the use of game theory to stimulate the correct behaviour in the presence of malicious adversaries. Game theory and mechanism design offers the possibility to stimulate the desired behaviour as defined by the protocol designer given the right incentives, enforcement of a given strategy, or simply by providing uncertainty in the duration of the game. Game theory is complimentary to AIS: the former provides a framework to reason whilst the latter provides techniques to detect and respond to attacks.

2.6 Summary

This chapter reviewed a number of approaches which proactively mitigate misbehaviour in ad-hoc networks as well as those which react to the changing MANET environment in terms of the security attributes they aim to meet (§2.3). Whilst the threats from adversaries constantly evolve (Table 2.2), they arise from the presence of a selfish, malicious or faulty adversary. At a basic level most threats severely impact the *availability* of the network, whether indirectly by decreasing the network lifetime or by the introduction of routing instabilities. Many of the reviewed mechanisms (§2.3) do not consider detection ambiguities within a MANET environment through their use of promiscuous notifications, which often result in a large false detection rate. Furthermore, many do not consider appropriate responses. The two emerging fields of AIS and game theory have been chosen in this study as they possess desirable properties in a MANET environment. AIS and the natural immune system it is based upon provides a decentralised, reactive, defence-in-depth solution to detecting and eradicating infection. It is not sufficient to rely on the AIS itself as there is the imperfect signalling to contend with in a wireless environment. Game theory is proposed as a way to deal with this, by reasoning about detection events during IDS activities. Chapter 4 identifies and develops adversary models for MANET. Chapter 6 details the use of a general IDS framework to with game theoretic reasoning. Chapter 7 enhances the game theoretic reasoning with an AIS component.

CHAPTER 3

SIMULATION METHODOLOGY

3.1 Introduction

SWANS is the wireless network simulation component built on top of the Jist simulation engine, a Java-based, discrete time event simulator (Barr, 2004b,a). As of v1.0.6, development of the simulator has been discontinued, but there is a growing community of users that continue to provide bug fixes and extensions to the Jist/SWANS kernel. A measure of this is the number of releases using Jist/SWANS at their core; these include SIDnet-SWANS (Ghica *et al.*, 2008) optimised for simulating sensor networks and SWANS++ (Otto & Choffnes, 2007) for simulating vehicular ad-hoc networks.

This chapter describes briefly the evaluation of Jist/SWANS and a number of competing simulators, introduces the design of simulations in the simulation environment and details the development work of a number of extensions that were added to enhance the simulator. A number of validation studies are also presented that highlighted bugs in several of the Jist/SWANS protocol implementations. The chapter presents the fixes applied to stabilise the metrics achieved from these validation scenarios. Finally, a number of performance metrics are defined to measure the efficacy of an attack carried out by an adversary and measure the response effectiveness of an IDS from output network traces.

3.2 Simulation environment

Jist/SWANS (Barr, 2004b,a), Ns2 (ns2, 2010) and Opnet (OPNET Technologies Inc, 2010) were each evaluated for this study, and Jist/SWANS was adopted for a number of reasons. The author was familiar with the Java language and Barr *et al.* (2005); Schoch *et al.* (2008) showed that the simulator performed and scaled well in simulations containing large numbers of nodes. A brief evaluation of Ns2 revealed a programming model that used a combination of a C++ for simulation models backed by a tcl scripting engine, resulting in complex simulation code. Opnet was discounted as the kernel source code was not readily available. Additionally, the Jist/SWANS Application Programmers Interface

(API) was well documented and facilitated rapid prototyping and was easy to understand as the project was organised into a set of packages which corresponded roughly to the classical OSI model (Figure 3.1). The code pattern enforced strict simulation communication patterns, for example preventing MAC frame headers being made available to the network layer. It also came with a large number of radio channel models, propagation models, mobility models and routing protocols by default. The availability of propagation models was especially important to determine the effects these had on intrusion detection performance.

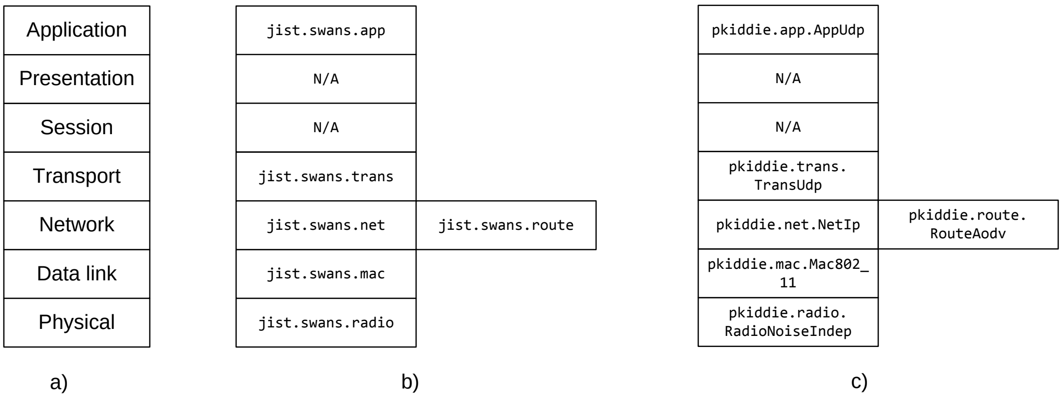


Figure 3.1: a) OSI Model, b) corresponding SWANS package structure, c) Example node definition

As the project progressed, the use of Jist/SWANS presented a number of challenges. Simulated entities were buggy and incomplete, which led to poor network performance during initial validation studies that was not in line with expectations. In addition, several vital components were lacking, including trace support and a user interface to view the simulation execution, necessary for an effective simulation study. Thus, a trace structure was defined, file logging was implemented, and a simulation user interface developed. These extensions are presented in Appendix C for the interested reader. Also, contributions from the Jist/SWANS community were applied in the course of this work, namely Schoch (2009) who provided patches to the kernel and Kliot (2009), who provided bug fixes for several protocol implementations that were used in this study.

3.3 Metrics

A number of metrics were defined for use to compare the effectiveness of the developed IDS's. Goodput, packet delivery ratio and end-to-end delay are performance metrics and provide an indication of the efficacy of responses invoked by the IDS. Response time and response rate were IDS specific metrics derived to compare the performance of individual IDS's. Each metric was implemented as a script that post-processed the trace file generated by the network simulation. These scripts can be found on the web as described in Appendix D.

3.3.1 Goodput

Goodput, G, is defined as the application layer throughput, which is the number of *useful* data bytes sent by flow source s that was received by a flow destination d, excluding any retransmissions or routing overheads. This was measured by considering only the application layer events for s and d within the trace file. The metric was implemented in `goodput.pl`. Flow goodput is given by:

$$G(s,d) = \frac{bytes\,received(d)}{t(last\,packet\,sent(s)) - t(first\,packet\,sent(s))} \tag{3.1}$$

3.3.2 Packet Delivery Ratio

Packet Delivery Ratio, PDR, is the ratio of data packets sent by s that were received d. Like the goodput, this can be measured by considering the received and sent application layer events for pairs of s and d only. The PDR for a flow is given by:

$$PDR(s,d) = \frac{data\,packets\,received(d)}{data\,packets\,sent(s)} \tag{3.2}$$

This was implemented in `packetstats.pl`, which additionally outputted losses from all possible sources, including queue, mac, route, ttl, misbehaviour or ids.

3.3.3 Average end-to-end delay

End to end delay for a packet p, τ_p, is defined as the interval between an application layer packet p being sent by s and received by d's application layer, and is given by:

$$\tau_p(s,d) = t_p(send(s)) - t_p(receive(d)) \tag{3.3}$$

This metric was implemented in `delay.pl`.

3.3.4 Control Overheads

Control overheads are computed as the ratio of routing and IDS transmissions to data transmissions in a simulation and is based upon the definition by Marti *et al.* (2000) where a transmission is defined as the sending or forwarding of a packet. Broadcasts (e.g. route requests) are counted as a single transmission. This measure was used instead of *normalised routing load* (NRL), defined to be the ratio of the sum of transmitted control messages to the sum of data packets **delivered** to a destination (Perkins *et al.*, 2001), since NRL is sensitive to misbehaviour in which packets are dropped as they are forwarded towards the destination. This metric was implemented in `routingoverhead_marti.pl`.

3.3.5 Response rate

The response rate is computed as the ratio of responses taken against IDS nodes and adversaries to the total number of responses in the simulation, and provides insight into the ability of the IDS to discriminate between benign and malicious behaviour. This metric was implemented in `response_count.pl`.

3.3.6 Average recovery time

Recovery time measures the time difference when an active attack reduces a metric to its minimum and the IDS restores the metric to the pre-intrusion average. To measure this, a time averaged metric was processed by `peakdet` (Billauer, 2011), to compute the set of maxima and minima within the time averaged data that have a given peak threshold. The recovery time is calculated from the difference of the time when the first minimum metric after misbehaviour starts was located and the time which the metric exceeds the pre-intrusion average.

In most cases, the misbehaviour results in data loss so goodput (§3.3.1) provided an appropriate metric to determine recovery time. For other cases where misbehaviour did not adversely affect goodput, other metrics, such as intercepted bytes (implemented in `window_intercepted_packets.pl`) was used.

To determine recovery time, the Matlab function `avg_recoverytime` was developed and the base metric time averaged over 20 samples. The peak threshold was set to 5% of the pre-intrusion average metric, which resulted in good discrimination of extrema without being susceptible to noise.

3.4 Monte-Carlo framework

Network simulations are the product of a large number of random variables that result in the topology, flow placement, traffic generation, packet and frame losses and other features of a simulation run. These properties fluctuate between runs, thus sampling a number of repeats is vital to ensure that the resulting network metrics are credible. A Monte-Carlo approach was adopted that repeated simulations for only the necessary number of times until the running average of a studied metric converged to within a given threshold. In Jist/SWANS, a Monte-Carlo simulation runner was developed as an intermediate layer between the driver and Jist/SWANS runtime, and is shown in Figure 3.2.

The runner is supplied with the chosen post-processing script (§3.3) and some convergence threshold. The script post-processes the trace file once the simulation has completed and produces a metric. The runner adds this to a set of samples and computes a running average. If the difference between the old and new running average is less than the convergence threshold, the simulation is terminated, else the simulation is repeated once more. When the runner terminates, an average, variance and standard deviation of the metric is output. This process is illustrated in Figure 3.3. `DescriptiveStatistics`, part of the Commons Math Library (Apache Commons, 2010), was used to store the set of samples in the runner and compute the average, variance and standard deviation. The distribution of samples was also output as a histogram to check the resulting distribution was unimodal and ensure the mean metric was useful as an end result of the simulation.

To further automate metric collection, the runner was developed so multiple Monte-Carlo sessions could be queued for possible combinations of variables. For example, one variable could be a choice of router from the set $\{defenceless, omniscient\}$, whilst the second could be the ratio of adversaries present on the field, taking a minimum value of 0, maximum value of 0.5 and step size of 0.1. The Monte-Carlo runner enumerates through all the possible values, collecting metrics for a given simulation with the following parameters:

- defenceless, ratio of adversaries = 0.0
- omniscient, ratio of adversaries = 0.0
- defenceless, ratio of adversaries = 0.1
- omniscient, ratio of adversaries = 0.1
- ...

In the above description, two variable definitions were demonstrated. A `range` variable, like the ratio of adversaries on the field, takes a minimum, maximum and step size. A `choice` variable, such as the choice of router, takes a comma separated list of possible values. These are illustrated in Table 3.1.

Table 3.1: Variable types supported by Runner.

Type	**Arguments**	**Example**	**Notes**

`range`	`type;`
	`name;`
	`min;`
	`max;`
	`step`
		`type=range;`
		`name=pcAdversaryNodes;`
		`min=0.0;`
		`max=0.6;`
		`step=0.1`
			Increments `min` by `step` each time, till `max`.

`choice`	`type;`
	`name;`
	`choices`
		`type=choices;`
		`name=router;`
		`choices=aodv,`
		`aodv-omniscient`
			Invokes simulation for each of the choices specified in `choices`.

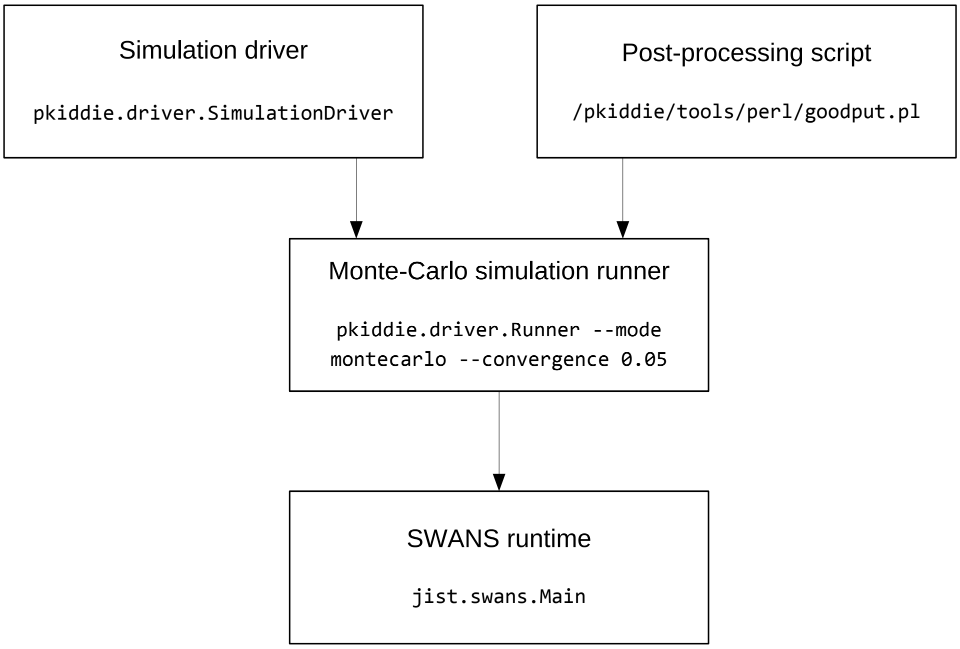

Simulation driver
pkiddie.driver.SimulationDriver
Post-processing script
/pkiddie/tools/perl/goodput.pl
Monte-Carlo simulation runner
pkiddie.driver.Runner --mode
montecarlo --convergence 0.05
SWANS runtime
jist.swans.Main

Figure 3.2: Monte-Carlo simulation runner acts as an intermediary between driver, post processing script and SWANS runtime.

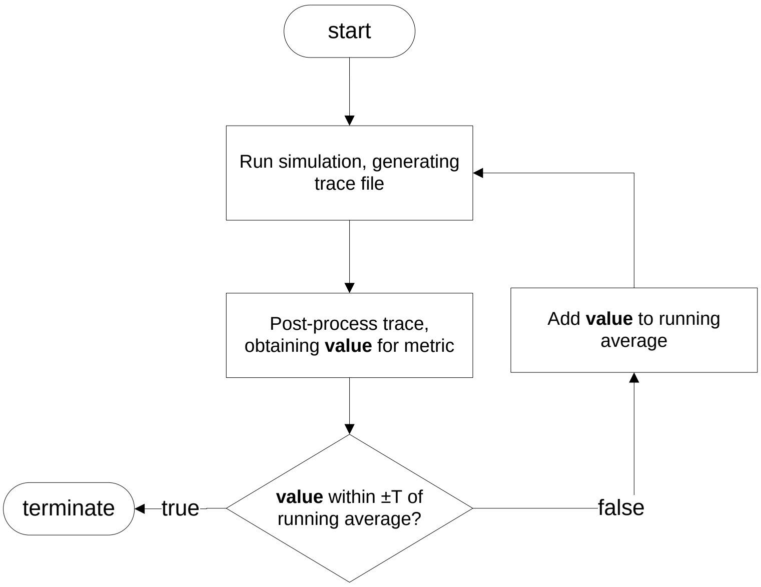

start
Run simulation, generating trace file
Post-process trace, obtaining value for metric
Add value to running average
terminate
true
value within ±T of running average?
false

Figure 3.3: Flow diagram for Monte-Carlo simulation runner

3.5 Validating Jist/SWANS

A significant amount of time was spent early on in development to ensure the simulation models supplied with Jist/SWANS produced network metrics comparable to other network simulators when simulating similar networks (including propagation environment and node models). Ns2 was used as the benchmark network simulator since its models were validated by the contributions of a large community of users.

3.5.1 Methodology

An Ns2 script, `parse_jistswans.tcl`, was created to enable the user to drive a simulation from a Jist/SWANS trace file with the same topology, traffic distribution and mobility pattern of the original simulation. The script extracted `scenario`, `add_node`, `move_node` and `createflow` events from the trace file and mapped them to the relevant Ns2 commands. For example the `move_node` event in Jist/SWANS was mapped to the `setdest` in Ns2.

Additionally, the node models used by the Ns2 simulations were hard coded into the script and matched those employed in the Jist/SWANS simulation. In particular, the Ns2 radio model was configured to match the default Jist/SWANS radio configuration. To this end, `RXThresh_` was configured appropriately by compiling and running `/ns-2.34/indep-utils/propagation/threshold.cc` and specifying a 2.4Ghz frequency.

Tables 3.2, 3.3 and F.2 present the runtime environment used for Jist/SWANS and Ns2 respectively to ensure the study satisfied the basic repeatability criteria of Kurkowski *et al.* (2005b,a). The scripts used can be found on the web as described in Appendix D.

Table 3.2: Runtime environment for Jist/SWANS simulator

Simulator	Jist/SWANS	
Version	based on v1.0.6	Modified simulation kernel from Schoch (2009) and own amendments. Full list of changes in Appendix B
Operating System	Windows 7 RTM (7600), 64bit	

Java Runtime Environment (JRE)	1.5.0_22, 32bit	Jist/SWANS currently incompatible with Java 1.6 and higher due to a dependency on Bytecode Engineering Library (BCEL) (Apache Jakarta Project, 2010). Development on BCEL stopped at Java 1.5 and is buggy when rewriting the bytecode of Java 1.6 class files.
Random Number Generator (RNG)	`java.util.Random` pseudo random number generator (PRNG)	linear congruential type; Use of global simulation PRNG, `random`, specified in `jist.swans.Constants`. Seed used is overridden in simulation driver, with unique seed specified per simulation run.

Table 3.3: Runtime environment for Ns2 simulator

Simulator	Ns2	
Version	ns-allinone-v2.3.4	
Operating System	Ubuntu 8.04 i686 (32bit)	kernel: 2.6.24-23-generic
gcc	4.2.4	
Random Number Generator (RNG)	default Ns2 RNG	linear congruential type

Further differences in the defaults employed by both simulators' models were found and were normalised to ensure a consistency in the simulated network environments. For example, the AODV router in Ns2 supports both HELLO messages and link layer feedback as methods to determine neighbour connectivity. whilst the Jist/SWANS AODV router, `RouteAodv`, supported only HELLO messages. For a fair comparison, HELLO messages were used in Ns2 by commenting out `#define AODV_LINK_LAYER_DETECTION` in `/ns-2.34/aodv/aodv.h` and recompiling.

The Monte-Carlo framework introduced in §3.4 was applied to the Jist/SWANS scenario, which generated a number of Jist/SWANS trace files that drove Ns2. Packet delivery ratio and end-to-end delay metrics were produced for traces produced by both simulators simulations and they showed that `RouteAodv` suffered from a poor PDR and higher end-to-end delay than the Ns2 AODV router. On closer investigation, the *HELLO_INTERVAL* constant in `RouteAodv` and used during route maintenance, was found to be set too high, resulting in a large number of undeliverable (and consequentially dropped) packets that worsened in highly mobile scenarios. In the default implementation, it could take up to 90s (*HELLO_ALLOWED_LOSS*HELLO_INTERVAL* s) to detect a link break. The defaults in both simulators are shown in Table 3.5.

Table 3.5: Default value of AODV constants in Jist/SWANS and Ns2

	Jist/SWANS (v1.06)	**Ns2 (v2.33)**
AODV_TIMEOUT	30s	N/A
HELLO_INTERVAL	30s	1s
HELLO_ALLOWED_LOSS	3	2

AODV_TIMEOUT is specific to `RouteAodv` and regulates calls to a `timeout` method every *AODV_TIMEOUT* seconds. This method performs periodic housekeeping, including checking the reachability of neighbours. To do so, it iterates through the set of outgoing nodes in the `outgoingSet`, checking if the number of periods a HELLO message or other broadcast message was not received has exceeded *HELLO_ALLOWED_LOSS*. The AODV specification (Perkins, 1997, §10) recommends the *HELLO_INTERVAL* constant be set to 1s, but in `RouteAodv`, this is meaningless unless *AODV_TIMEOUT* is also set to 1s. The results obtained before and after the constants were updated are shown in Figure 3.4 and illustrate the results obtained by `RouteAodvMultiInterface`, where *AODV_TIMEOUT* and *HELLO_INTERVAL* was set to 1s were much more comparable.

The scenario detailed in Table F.2 was then scaled up to include more flows and higher node densities, again driven primarily by Jist/SWANS and run in Ns2. As shown Figure 3.5, the PDR obtained in the Jist/SWANS simulation was significantly lower whilst the end-to-end delay was higher than the equivalent Ns2 scenario, requiring a further investigation into the cause of the losses. A subtle interaction between a bug in the `RadioNoiseAdditive` radio model and a bug in the `Mac802_11` MAC layer was identified: nodes entered backoff during congestion but never returned from this state. Schoch (2009) and Kliot (2009) had independently discovered these bugs and provided appropriate fixes, so these were added to produce a second version of `RouteAodvMultiInterface` resulting in more consistent network performance similar to that of Ns2 (Figure 3.5).

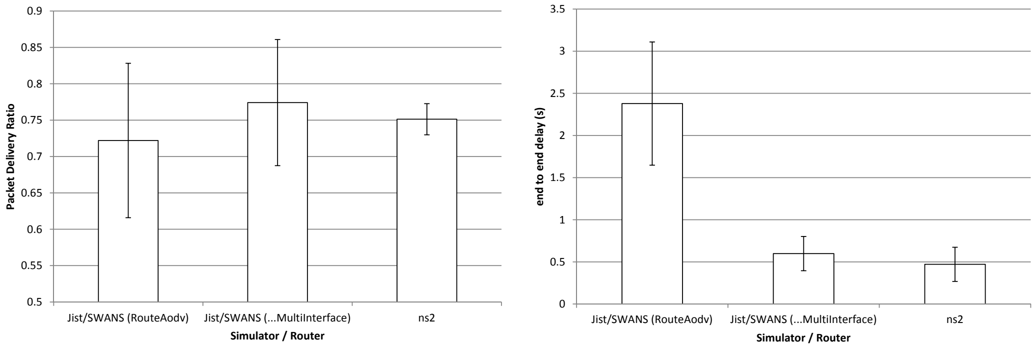


Figure 3.4: Packet delivery ratio and end to end delay achieved in `pkiddie.driver.Ns2ValidationSimple.java` from the use of the original and customised AODV routers in Jist/SWANS, and the Ns2 AODV router, set to use HELLO messages.

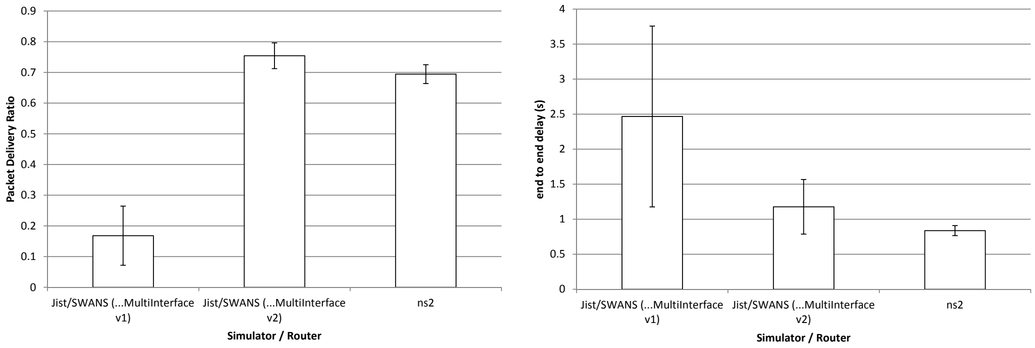


Figure 3.5: Packet delivery ratio and end-to-end delay achieved in scaled up scenario by Jist/SWANS with versions of `RouteAodvMultiInterface` and Ns2 AODV router.

Table 3.6: Jist/SWANS simulation parameters overridden in scaled up scenario and implemented in `pkiddie.driver.Ns2ValidationScaledUp.java`.

Simulation Property	**Jist/SWANS Class**	**Property**	**Value**
Route	`pkiddie.route.` `RouteAodvMultiInterface`		
flows	n/a	n/a	`20`
nodes	n/a	n/a	`100`

A final round of validation ensured the new AODV router produced results comparable to those in the literature. A number of scenarios presented in highly cited works were implemented in Jist/SWANS, including Perkins *et al.* (2001) and Takai *et al.* (2001) who evaluated AODV in the same scenario with employing several fading and path loss models. Both authors adopted Ns2 and used the default AODV router, achieving PDRs between 90-100% and 20-40% in each scenario respectively. The average PDRs obtained by Jist/SWANS (shown in §F.1 and §F.2) varied between 80-100% and 15-20%, correlating with these published results, which gave additional confidence in the robustness of the Jist/SWANS simulation models.

3.6 Summary

In this chapter Jist/SWANS was evaluated and chosen as a simulation platform. A number of components were added to ease simulation debugging, development and post-processing for metric collection. Significant validation work revealed bugs and incomplete implementations within several protocol layer implementations. Where possible, these bugs were fixed and the functionality added, which produced results comparable to published simulation studies for large scale MANETs in realistic propagation environments. An exhaustive list of all modifications made in this chapter are provided in Appendix B. This gives confidence in the results obtained by Jist/SWANS for future simulations. The development of a general framework for a MANET IDS is detailed in the next chapter.

CHAPTER 4

DEVELOPMENT OF ADVERSARY MODELS

In Chapter 2, a number of challenges of intrusion detection in MANETs were described. A significant challenge chosen as the subject of this work is to classify misbehaviour in the presence of a number of uncertainties, namely in detection. Many of current techniques we reviewed previously do not factor in these uncertainties which in turn leads to a high false detection rate.

This chapter describes the attacks that have been investigated in this project and provides implementations of these within AODV for the Jist/SWANS simulator, introduced in Chapter 3. The uncertainties in the detection of the each of these attacks have been identified. The capabilities of the simulated adversary are explored and described.

4.1 Introduction

The characteristics of MANETs (reviewed in Chapter 2) lead to security challenges that span every layer of the protocol stack. Attacks can be directed at the physical layer, as in the case of jamming (Xu *et al.*, 2005) or an adversary can drop or delay packets to target applications that require high Packet Delivery Ratio (PDR) and small delays, such as Voice over IP (VoIP) (Caro *et al.*, 2008). Many of these are classical networking attacks, but the unique characteristics of a MANET, such as the use of decentralised algorithms over which services such as routing are run, and a shared wireless medium (Deng *et al.*, 2002) lead to novel security challenges. Additionally, member nodes implicitly trust one another and attacks can occur by both *outsiders* and *insiders*. An *outsider attack* is caused by an adversary that does not belong to the MANET and is often brute-force (such as jamming communications) whilst an *insider attack* is often more focussed, targeting specific nodes or services, and emerges from captured nodes belonging to the network (Helsinki & Kärpijoki, 2000). To become an insider, an adversary could defeat physical layer security and cryptography exclusively through brute-force methods at considerable computational expense. An easier option in unattended MANET deployments is the capture of nodes, upon which the adversary captures all cryptographic keys, which can be used to access network services.

Routing in MANET is a distributed service run on each node and requires the full participation of each to function most efficiently. As is the case with any open distributed system, MANETs experience nodes with different preferences, objectives and utilities, or put simply, different *types*. Feigenbaum & Shenker (2002) term these types as selfish, malicious, faulty, or obedient. We redefine these agent types with respect to their objectives during routing.

A *selfish node* is strategic, conserving its finite, measurable resources (e.g. energy or own throughput) for its own activities and does not co-operate in the forwarding of data routing as is expected by others (Michiardi & Molva, 2002). To motivate a selfish node to co-operate, additional incentives are needed, such as the use of a strategy to service data for the requestor if it has been reciprocal in the past (Felegyhazi *et al.*, 2006). This strategy can bias the computation of the selfish node's utility function towards participation. Selfish nodes do not intentionally damage other nodes (Michiardi & Molva, 2002) and their behaviour should not cause inconsistencies (such as the propagation of routing loops) in the routing protocol of other nodes. However, the presence of selfish nodes can affect the availability of end-to-end routes if they connect network partitions, and may reduce the lifetime of the network as the load on honest neighbours increases.

A *malicious node* purposely inflicts damage by finding exploitable weaknesses in MANET services, but this behaviour *may* be bounded. If it is, the node is termed strategic, through the need to be in a good standing with neighbours to make use of the network in the future. This is usually measured in terms of reputation, for which security protocols such as CORE (Michiardi & Molva, 2002) or CONFIDANT (Buchegger & Le Boudec, 2002b) exist. The effects of malicious nodes are wide ranging and may lead to routing inconsistencies that may propagate throughout the network unless they are detected early, thus making attribution of the initial misbehaviour difficult. *Faulty nodes* do not follow the prescribed protocol as dictated by the service, due to hardware failure or a software related problem such as an incorrect implementation, buggy, crash-prone code, or poor software design. Finally, *obedient nodes* adhere to the operations prescribed by the protocol of the service in question.

4.1.1 Threats to MANET routing

Reactive routing protocols expose a range of vulnerabilities as they operate by discovering routes *on-demand*. The adversary can attack in real time with minimal set-up time (Wang *et al.*, 2003) and a range of mechanisms are available to the adversary by which to invade routes. The range of attacks is much more limited in a proactive routing protocol, in part due to routing information being exchanged well in advance requiring that attack must be fashioned in advance also (Wang *et al.*, 2003).

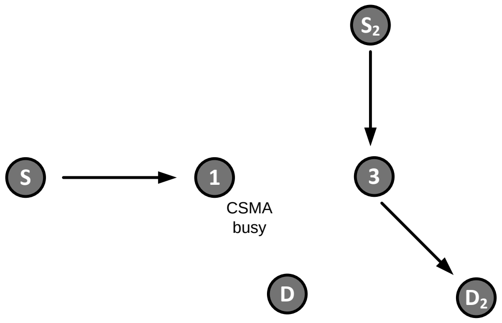


Figure 4.1: A topology with two established flows between S and D and S2 and D2. S would detect 1 to be dropping packets if S was analysing promiscuous notifications using a Watchdog technique. 1 is simply waiting for the medium to be free to send data to D.

A number of attacks can be directed at the routing layer of target nodes. Black hole, wormhole and grey hole attacks attract data from many sources to a small number of adversaries and may result in data loss, modification or information disclosure (Deng *et al.*, 2002). Flooding of routing packets, whether erroneous or not, depletes batteries, causing premature node expiry, and decreases network lifetime. A node can prevent itself from becoming part of a route by ignoring route requests and simultaneously making use of the network.

4.1.1.1 Selfishness

AODV route discovery specifies that if a node receives a request and it is not the destination, or an intermediate node with a fresh enough route to the destination, that it should forward the request (Perkins *et al.*, 2001). During route discovery, neighbours that forward requests set up a chain of reverse route entries which point to the originator of the request. This reverse route is used to forward a route advertisement to the originator should the route request be satisfied. A selfish adversary preserves resources by discarding requests. Thus, reverse routing entries never include the adversary, guaranteeing that the chosen route always excludes the adversary (Figure 4.2). By continually failing to forward requests, the selfish adversary can minimise the number of routes it serves. The net effect of this is that the routing load which would normally be satisfied by the selfish node is unfairly distributed on the remaining benign nodes, which leads to premature battery exhaustion and a reduction in network lifetime. Furthermore, the selfish adversary continues to use of the network to serve any of its own data flows.

Another common description of a selfish adversary is one that fully participates in route discovery (hence “agreeing” to carry the data flow), but switches off packet forwarding due to the associated energy cost (Felegyhazi *et al.*, 2006). This type of behaviour is fundamentally no different to a malicious adversary attacking with a black or grey hole,

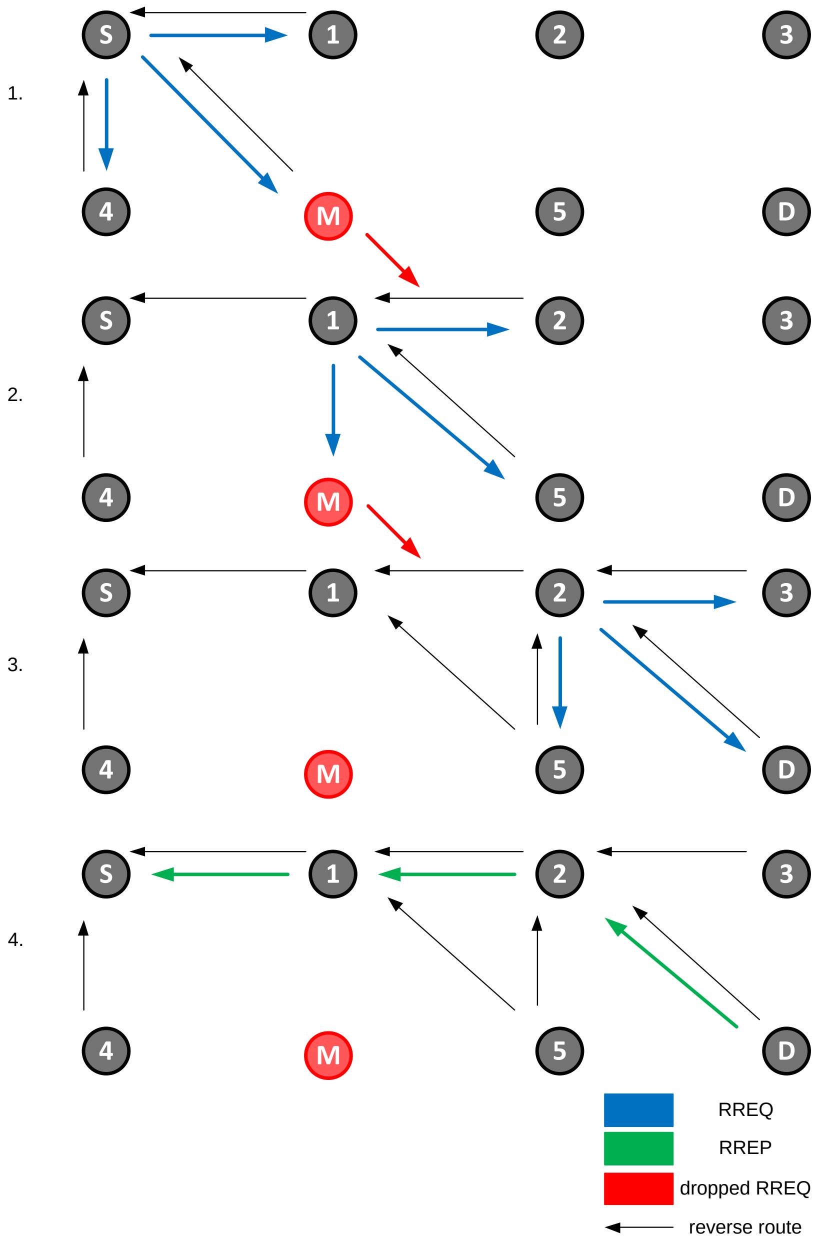


Figure 4.2: A selfish node (M) that does not forward route requests during route discovery. 1. S sends a route request for D. 2. M explicitly drops the RREQ whilst the other neighbours of S forward the route request to their neighbours. 3. The route request arrives at D. None of the reverse route entries point to adversary M. 4. The path the route advertisement from D takes. M has excluded itself from servicing a possible route.

thus we also classify this as malicious, rather than selfish behaviour.

4.1.1.2 Black hole

The black hole is a well-studied attack on MANET routing protocols (Deng *et al.*, 2002; Hu *et al.*, 2005; Buchegger & Le Boudec, 2002a; Pirzada & McDonald, 2004) upon which an adversary drops the data it is meant to forward on behalf of a neighbour (Figure 4.3). To increase its chances of success, the adversary can intercept routes by continuously advertising itself as having a path to a requested destination. The adversary *short-circuits* the route table lookup and responds by issuing a fake route advertisement. The net effect is that a significant number of flows are attracted toward the black hole adversary and PDR drops substantially. An observation of dropped packets does not confirm the existence of a black hole since physical layer phenomena, such as a busy channel, collisions, fading caused by receiver mobility, and path loss, may all cause frames to be dropped, also leading to dropped packets.

4.1.1.3 Grey hole

The grey hole attack covers a number of variations of the black hole attack carried out probabilistically, usually for a list of targets. The probability used to determine whether to drop a packet or to send a fake advertisement can be unique to each target (Srinivasan *et al.*, 2008). By targeting a subset of nodes and dropping *some* data, the adversary introduces further uncertainty than it can use to manipulate a neighbour's perception of it and avoid detection.

4.1.1.4 Wormholes

Wormholes require at least a pair of co-operating adversaries (Yang *et al.*, 2004), and targets the selection of shortest paths between source-destination pairs in distance vector routing (Baras *et al.*, 2007). The first adversary receives packets at one point in the network and tunnels them over a private communication channel to the second adversary, who replays them at another point in the network (Hu *et al.*, 2002, 2005). The adversaries possess at least two network interfaces and are most successful at capturing data when the secondary (tunnel) interface has a higher throughput and lower latency than the primary network interface, and the tunnel distance is greater than the wireless transmission range of a single hop (Hu *et al.*, 2003a).

Wormholes are a form of a *rushing attack* (Hu *et al.*, 2003b), which targets the route discovery process, namely the fact that the destination, or an intermediate node with a route to the destination replies to the first route request received and discards requests with the same RREQ ID that arrive later. More generally, this behaviour corresponds to

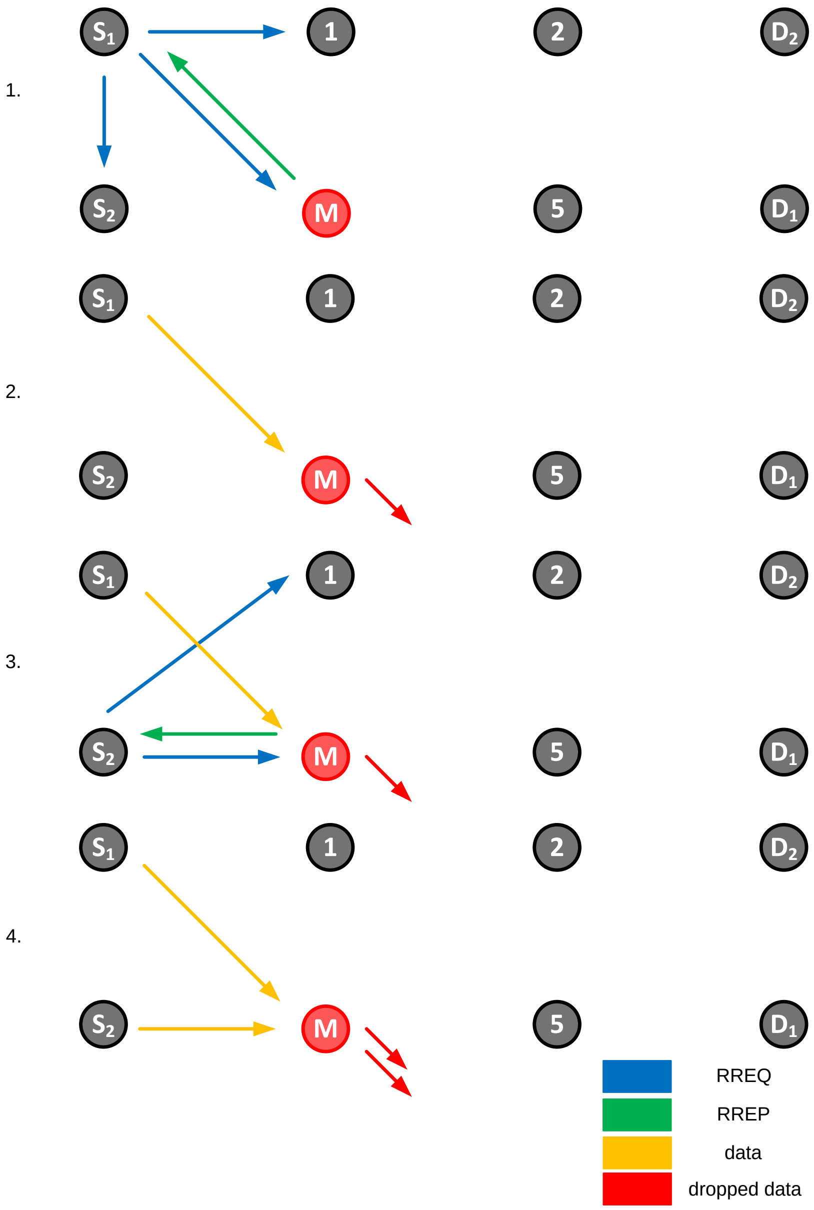


Figure 4.3: A black hole at M which attracts data flows from S1 destined to D1 and S2 to D2 respectively, and then drops the data. 1. S1 sends a route request for D1. M does not have a route to D1 but immediately sends a route advertisement to D1. 2. S1 uses M as the next hop to D1. Any data is dropped. 3. Another source, S2, sends a route request for D2. Again, the adversary responds immediately. 4. Two data flows have been attracted toward M.

a routing *race condition,* which arises when the routing service takes an action based on the first instance of a given message it receives, ignoring later instances of that message (Karlof & Wagner, 2003). Hence, if the adversary tunnels route requests through a higher throughput interface (implicitly rushing it), the adversaries are almost guaranteed to be part of the corresponding route and the flow will use the tunnel interface (Hu *et al.*, 2003b).

At this point the adversaries have not actually attacked the network in any way. In fact, they could be said to be offering a beneficial service as they are introducing other paths that data may flow, making the network more connected (Hu *et al.*, 2003a). Unfortunately, the existence of a wormhole puts the adversaries serving it in a powerful position as they will often be chosen to serve longer routes in the network. This position can be exploited by discarding, modifying or delaying packets, or disclosing information from the packets which should otherwise be forwarded over the tunnel.

4.1.1.5 Flooding Attack

Flooding attacks comprise any attack in which the adversary consumes resources by generating a number of repetitive messages and broadcasts them to its neighbours (Vigna *et al.*, 2004). The messages can optionally be constructed in such a way that they are processed by the neighbours of the adversary, disseminating them network wide and resulting in a collapse of network throughput in the worst case. In a RREQ flood attack, a malicious adversary continually originates route requests for a fictional destination, incrementing the RREQ ID so that neighbours process each request and forward them to their neighbours.

4.2 Development of adversary models for simulation

There are numerous published studies that identify and measure the effects of misbehaviour within a MANET environment, some of which were reviewed in §2.3. Unfortunately, few simulation models have emerged from this research and are often designed to support a particular scenario and/or topology. Thus, a number of general adversary models were developed to support the research in this thesis. The aim of developing these models were for each to be flexible enough to enable their application in any scenario and in any randomly generated topology. To develop the models, the actions necessary to successfully carry out each attack were decomposed. Often, these actions comprised of a period of data collection, the generation of appropriate routing packets and some confirmation of success before launching a successive attack. In this section, the actions required to carry out each attack are written as pseudo code, with minimum protocol detail necessary so that we can elaborate the attack processing. These attacks were implemented in Jist/SWANS v1.0.6 (Barr, 2004b,a), within `RouteAodvMultiInterfaceAdversary`, the adversarial router. It is worth noting that `receive` in this context means the routing layer is receiving a rou-

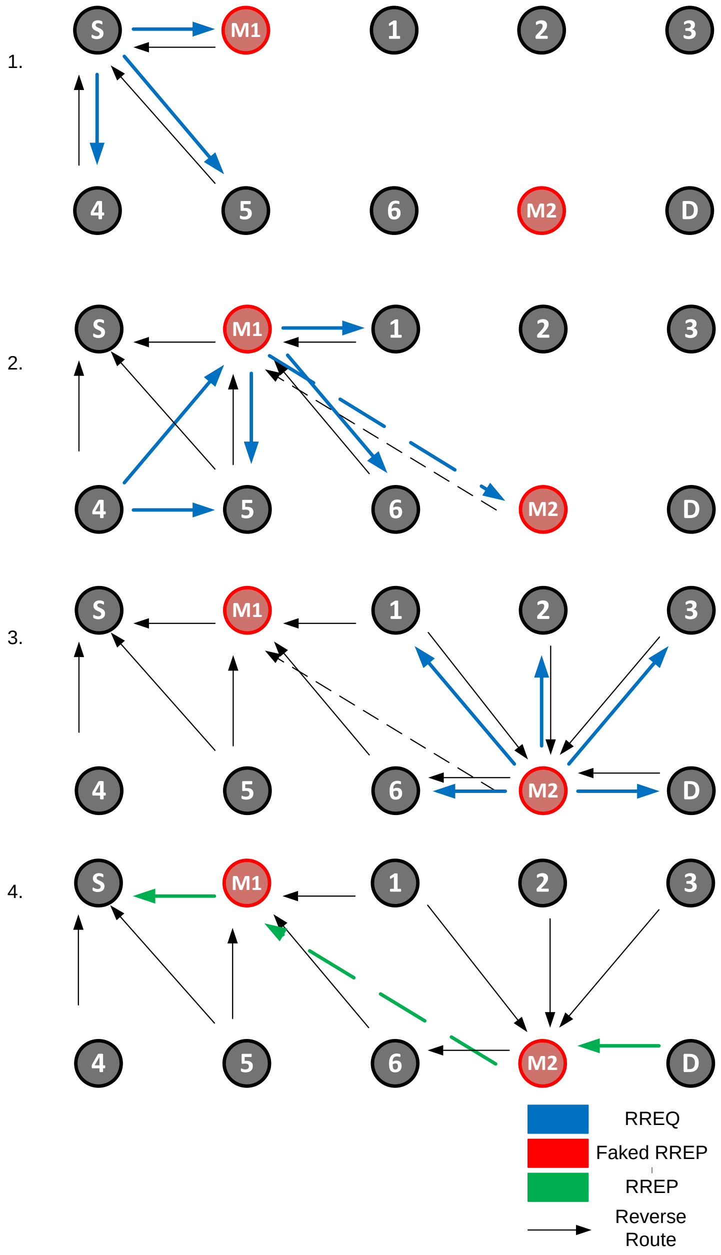


Figure 4.4: A wormhole attack in progress, with the tunnel between M1 and M2, who are colluding adversaries. 1. S broadcasts a route request to D. 2. M1 broadcasts the route request over all interfaces, including the high throughput tunnel between itself and M2. 3. M2 rebroadcasts the route request as normal, which reaches D before the third broadcast by the honest nodes has happened. 4. D replies as normal to the route request, which takes the route over the tunnel back to S. Any subsequent route request packets by the honest nodes are dropped by D.

ting message and send means the network layer is requesting the routing layer transmit a message that requires routing, which could be data, or routed control packets, in particular, route replies. Finally, netAddr and macAddr correspond to the network and MAC addresses of the node, used in most cases to check whether the packet should be targeted. This is used so an adversary does not target its own packets.

4.2.1 Selfishness

To implement selfishness within AODV, selfish adversaries drop any route requests received unless they are the requested destination (i.e. a node wants to start a data session with the selfish node). In RouteAodvMultiInterfaceAdversary, receiveRouteRequest() was overridden to drop the request received unless the requested destination was the selfish node. This is illustrated in the following pseudo-code, which illustrates the difference between route request processing by an honest and selfish node in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1 receiveRouteRequest for an honest node

```
if is destination then
  generate route reply
else
  if not in rreq buffer then
    decrement ttl
    if ttl greater than 0 then
      broadcast rreq
    end if
  end if
end if
```

Algorithm 2 receiveRouteRequest for a selfish node

```
if is destination then
  generate route reply
else
  drop
end if
```

4.2.2 Route invasion

Route invasion, in its various forms, is a precursor to the black hole, grey hole and wormhole attacks described in §4.1.1, and is of particular interest to an adversarial node that has just been captured to establish an attacking position within the MANET, whether routes are established and stable, or not. There are three classes of route invasion attacks: *passive*, in which routing messages are input or suppressed by the adversary during normal execution of the routing protocol, *active* attacks that cause premature error conditions on routes or inject unsolicited advertisements outside normal running of the protocol, and *hybrid* attacks, which combine both. For example, a false routing message sent during route discovery is *passive* as it is completed during an expected protocol procedure, in this case route discovery, whilst an unsolicited route advertisement is an *active* attack that is not specifically excluded by the routing protocol but is also not expected. From the adversary's perspective, route invasion should be controlled as the more routes an adversary successfully invades, the more loaded the adversary will become, and it is likely to be detected not for misbehaviour, but for excessive dropping instead if its interface queue length is regularly exceeded.

4.2.2.1 Route freshness

To understand route invasion fully, it becomes necessary to define a "fresh" or "more favourable" route advertisement, since route invasion targets the process by which reactive routing protocols select route advertisements and the method by which existing route table entries are removed. The AODV RFC (Perkins, 1997) specifies that upon receiving a route advertisement, the existing route table entry (if it exists) is updated in the following circumstances:

1. the sequence number in the routing table is marked as invalid in route table entry.
2. the Destination Sequence Number in the RREP is greater than the node's copy of the destination sequence number and the known value is valid, or
3. the sequence numbers are the same, but the route is marked as inactive, or
4. the sequence numbers are the same, and the New Hop Count is smaller than the hop count in route table entry.

The sequence number is generally used to ensure route freshness and guarantee loop-free operation in AODV, but it, and the Hop Count fields can be targeted by the adversary to cause a defenceless reactive routing protocol to select the adversary to serve a given route, even if the corresponding route through the adversary is not optimal.

4.2.2.2 Passive route invasion

Passive route invasion includes techniques that an adversary can invade a route during route discovery and route maintenance, and is suited for the invasion of new or stale routes. A couple of techniques were implemented in `RouteAodvMultiInterfaceAdversary` to achieve this.

The first and most simple technique causes the adversary to generate a more favourable advertisement for known destinations by adding a large constant to the destination sequence number (`dsn`) currently held in the route table entry, before it sends the advertisement to the target. This technique requires very little processing. When the target compares this with an existing route table entry, the route advertisement with the fake, higher `dsn` will overwrite any existing with a lower `dsn`. In general, advertisements with higher `dsn`'s are favoured over those with a lower hop count (Perkins, 1997, §6.7) . Thus, by implementing this technique, the adversary stands a good chance of being made part of the route.

The second technique involves the adversary creating a fake route advertisement to D from the receipt of a route request from a target source node, S (Figure 4.5), even when it does not have a route to D in its route cache. If the `dsn` is unknown, (S having had no prior interaction with D) the adversary guesses an appropriate sequence number based upon the maximum `dsn` of the current routes in its routing table. If, on the other hand, D was known to S in the past, the adversary can use the value of the `dsn` in the route request on which to base the `dsn` on in the faked advertisement. The adversary sends this fake route advertisement back to S. The two forms of this attack are for the adversary to generate a fake route advertisement as if it is from the destination and to generate the advertisement as if it is from an intermediate node with a route to the destination.

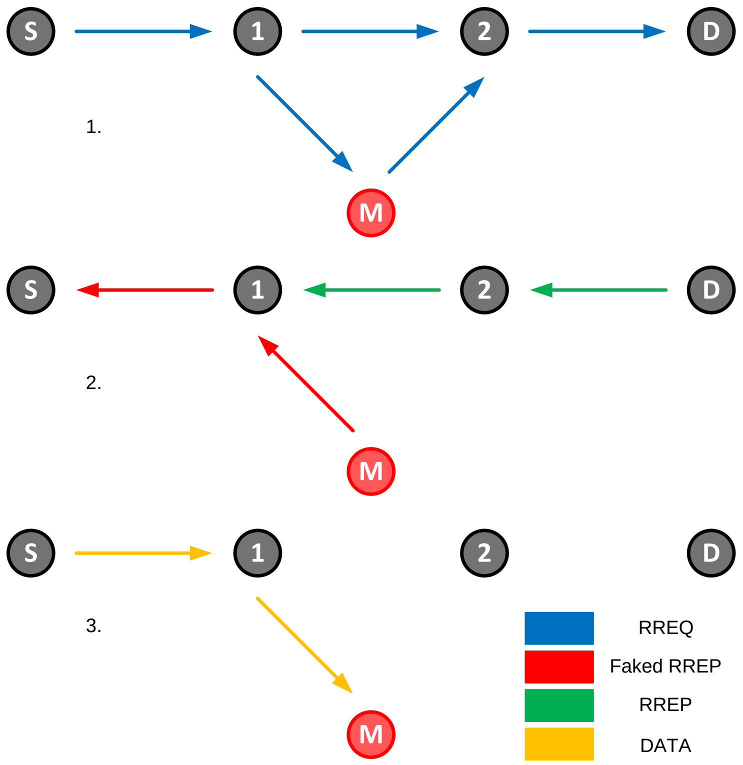


Figure 4.5: M invades route between S and D passively by creating and sending a fake route advertisement to S. 1. S broadcasts a route request for destination D. 2. M receives the route request and replies with a fake route advertisement. 3. Since M is closer to S than D is to S, the fake route advertisement from M is selected initially. This may be temporary based upon how attractive the real route advertisement from D is. M is now in a position to conduct further attacks as part of the flow.

Passive route invasion does not guarantee success for the adversary for a number of reasons. The attack is susceptible to mobility constraints, so D or S may be fewer hops than S is currently to M, limiting the success of the attack. Also, M may add only a conservative value to the `dsn` of the fake route advertisement, so the true route advertisement sent by D, or an intermediate node, may actually be more favourable. In this case, the fake route advertisement would either be ignored or added to the route cache and subsequently overwritten by the true advertisement, depending on the order that they are received. Thus, to increase its chances of being used to route data from S, the adversary can increase the value of the constant it adds to the `dsn`, or advertise a smaller hop count, to make the fake route advertisement more favourable. Large deviations of either from the norm are more likely to be detected.

If the adversary succeeds in invading the route and starts receiving data from S it is then in a position to carry out attacks. The adversary may not wish to arouse suspicion immediately so it could begin by forwarding data as normal. This may involve a route discovery for D if it short-circuited the original route discovery procedure. Alternatively, whilst it generated a fake route advertisement to D, it could have sent a proactive route request for D, used if successful in invading the route. In either case, whilst the adversary continues to be part of the route it can perform a number of data based attacks.

Sample pseudo code for passive route invasion is presented in Algorithm 3, describing the actions necessary to passively invade a route. In `RouteAodvMultiInterfaceAdversary`, `receiveRouteRequest()` is modified such that fake route replies are generated for any destination that is not the adversary. The ability for the adversary to generate a fake route reply either as the destination or as an intermediate node with a route to the destination is controlled by manipulating the RREP's `hopCount`. The maintenance of a *passive_invasion_cache* by the adversary is an optional procedure but ensures the adversary can regulate its rate of attack to reduce the likelihood of detection. Furthermore, it provides a notification of success so it can either attempt to invade the route again if unsuccessful or attack the successfully redirected flow (§4.1.1.2). As such, `receive()` is modified to check whether the data arriving at the adversary is from a targeted route in the *passive_invasion_cache*.

Algorithm 3 `receiveRouteRequest` method for a node attacking by passive route invasion

```
if is destination then
  send route reply
else
  rrep.origIp← rreq.origIp
  rrep.destIp← rreq.destIp
  rrep.hopCount← 0
  send fake route reply
  entry.origIp← rreq.origIp
  entry.destIp← rreq.destIp
  insert entry into passive_invasion_cache
  add timeout for entry
end if
```

Algorithm 4 `receive` method for a node attacking by passive route invasion

```
for all entries in passive_invasion_cache do
  if ip.src = entry.origIp and ip.dest = entry.destIp then
    //user defined function here
  end if
end for
```

4.2.2.3 Active route invasion

To intercept an existing flow an adversary can employ active route invasion. An adversary injects an unsolicited route advertisement for a destination D and sends it to the source S via a neighbour of the adversary that has been observed to be forwarding data for S. This is completed even whilst a route exists between S and D and is in use. Should the unsolicited route advertisement for the destination be more favourable than the existing entry in the route cache for intermediate hops back to S, M will succeed.

For the adversary, an initial stage of eavesdropping is required to collect data pertaining to target flows in the network, consisting of source-destination pairs and intermediate nodes that are currently forwarding data. Thus, active route invasion is more computationally expensive than passive route invasion, due to this additional requirement of state. When the adversary has identified a flow it is not part of (observed by only promiscuous packets being received for a given source-destination pair), it generates a fake route advertisement addressed to the flow source S and sends it to an intermediate node. Assuming that the targeted intermediate node runs a defenceless routing protocol, it modifies its next hop to use adversary M. Like passive route invasion, the adversary can regulate this behaviour by adjusting the interval by which active route invasion is attempted for observed flows. Active route invasion is shown in Figure 4.6, and implementation details are presented in Algorithm 5.

Algorithm 5 `receive` method for a node attacking by active route invasion

```
if ip message received promiscuously and not my ip message then
  rrep.origIp← ip.src
  rrep.destIp← ip.dest
  send fake route reply

  entry.origIp← rreq.origIp
  entry.destIp← rreq.destIp
  insert entry into active_invasion_cache
  add timeout for entry
else if ip message received not promiscuously then
  for all entries in active_invasion_cache do
    if ip.src = entry.origIp and ip.dest = entry.destIp then
      //user defined function here
    end if
  end for
end if
```

An interesting variation of the unsolicited RREP attack is where an adversary sends

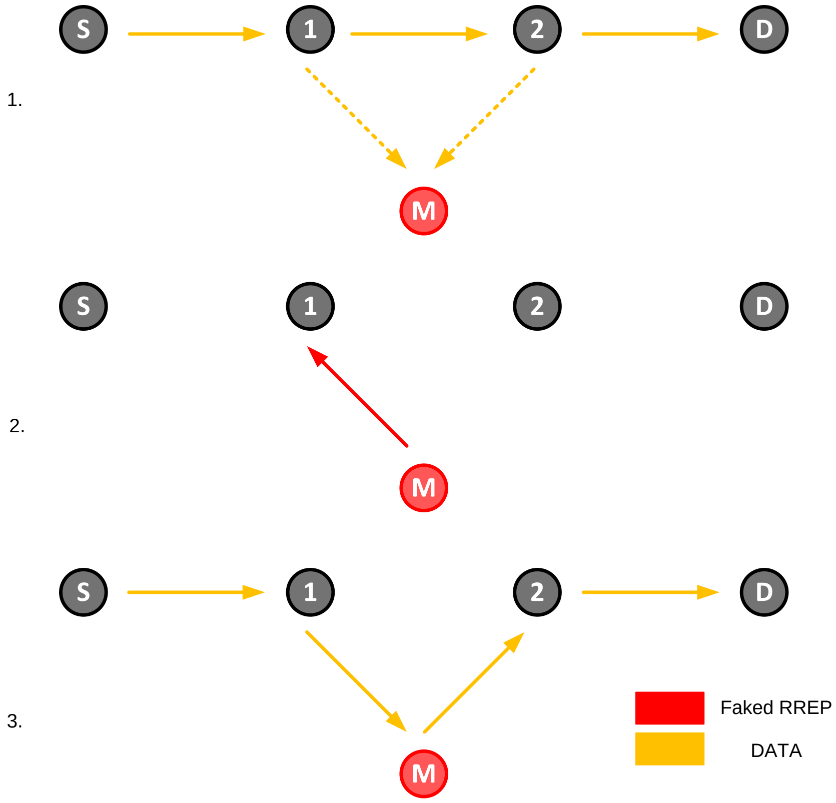


Figure 4.6: An existing data flow between S and D, which M wishes to invade through the use of active route invasion. 1. Adversary M caches nodes 1 and 2, and S and D as flow end-points by eavesdropping on relevant routing and data packets (dotted lines). 2. M sends a faked route advertisement about D to 1, and 1 updates the next hop to D accordingly. 3. M redirects the data flow between S and D, using this position to launch further attacks from.

a RREQ for the destination to retrieve its latest destination sequence number in order to base the sequence number used in the unsolicited advertisement, rather than choosing one arbitrarily. In addition, a number of other techniques are possible if the network has no protection from Sybil attacks (Douceur, 2002): an adversary can clone the identity of another node and create a route advertisement to redirect the route as if the advertisement was from a node participating in the route (Ning & Sun, 2003). It could also create an appropriate RERR as if the RERR was from a node participating in the route, forcing an upstream node to invoke route discovery that the adversary could subsequently target.

4.2.2.4 Hybrid route invasion

More elaborate schemes are possible in *hybrid* route invasion and are devised by combining both *active* and *passive* actions. If targets have no protection from Sybil attacks, a route error can be created and injected *actively* by the adversary. Unlike route replies, route errors are authenticated in AODV: they must be generated from neighbours upstream of the unreachable destination. The Sybil attacker sends a route error with a source IP address of a neighbour servicing the route with an unreachable destination of a later neighbour on the route. This is illustrated in Figure 4.7, whereby M sends a RERR as if from node 2 to node 1, and sends a fake route advertisement on the subsequent route discovery process.

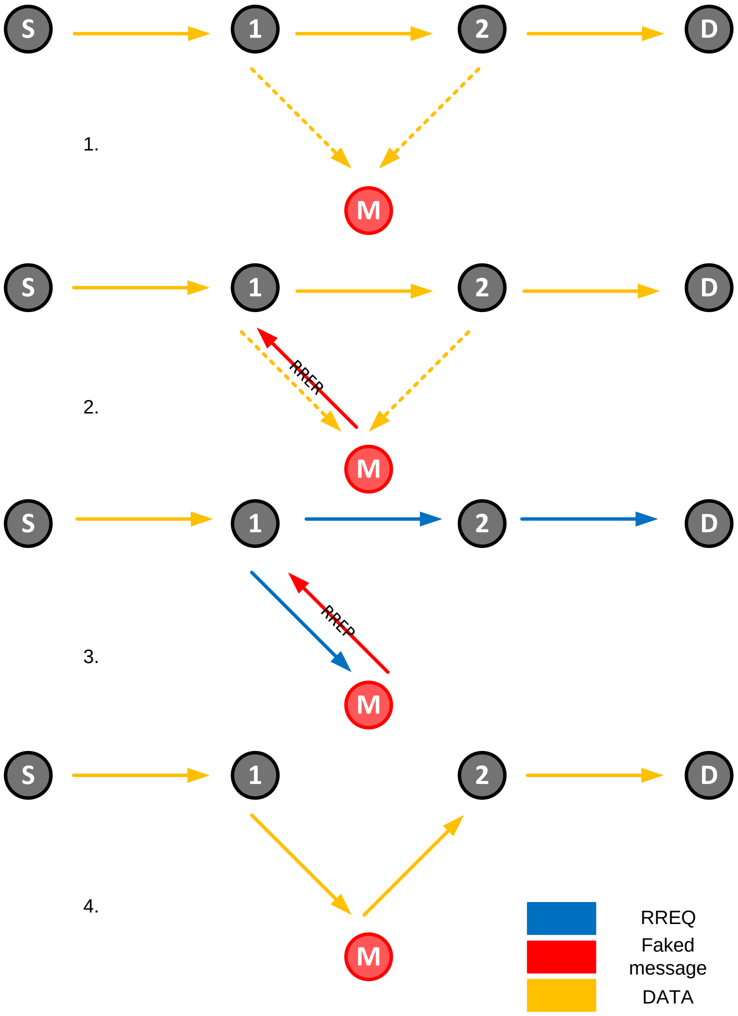


Figure 4.7: An existing data flow between S and D. 1. The adversary eavesdrops and caches the IP addresses/MAC addresses of intermediate nodes responsible for forwarding the flow data. 2. The adversary, having detected two intermediate nodes, creates a RERR from (2) about the route to D, and sends it to 1. 3. Node 1 queues packets from S whilst attempting to find a new route to D. M can reply passively with a forged route reply attack in order to invade the flow. 4. M has successfully redirected the flow through itself, through a combination of active and passive attacks.

As in active forms of route invasion, MAC-IP pairs of the target nodes need to be captured, in order to address the routing messages created by the adversary (§4.2.2.3). The net result is that the target invokes local repair to find an alternative route to the destination. If this is unsuccessful it often results in complete route rediscovery by S. The adversary now attacks *passively* by forging a more favourable route advertisement to the route requests generated from local repair (similar to the process involved in §4.2.2.2), and becomes part of the targeted route. These attacks, though interesting, are outside the scope of this thesis, so are not discussed any further.

4.2.2.5 The efficacy of route invasion attacks in MANET

Route invasion enables an adversary that has just captured a number of nodes to establish an attacking position within the MANET. Several mechanisms for achieving this have been discussed and the choice of an appropriate strategy is dependent on the current state of the targeted route that backs the flow, as shown in Table 4.1. Very often, a period of data collection is required to first identify the presence of flows, the set of neighbours serving those flows and an appropriate destination sequence number and hop count to use. The use of a realistic *dsn* and *hop count* could be important if the adversary fears it is going to be detected.

Table 4.1: Route invasion strategies to use depending on the state of the route.

Route state	**Route invasion strategy**
No route	Passive (§4.5)
Expired route	Passive (§4.5)
Route in use state	Active (§4.2.2.3)
	Hybrid (§4.2.2.4)

To determine the effectiveness of route invasion as a precursor to future attacks such as those in §4.2.3-4.2.4, simulations were run with increasing ratios of adversaries, with each employing one of a number of route invasion strategies. The simulations were configured as shown in Appendix A, with 10 CBR flows and 100 nodes. Where route invasion was used, the adversary proceeded to drop data that was not destined for it.

The impact on network PDR of benign nodes from adversaries employing various route invasion strategies is shown in Figure 4.8, illustrating that `fake-rrep`, a passive method for invading flows (§4.2.2.2) is the most effective approach an adversary can employ to reduce the overall network's PDR. Conversely, `fake-seqnum` does not result in any significant reduction of network PDR over the range of adversaries. The advantage of route invasion diminishes for a larger percentage of adversary nodes, which is explained by the fact that there is increased likelihood of encountering at least one adversary in the routes formed in simulations with a larger number of adversary nodes, in which route invasion becomes unnecessary.

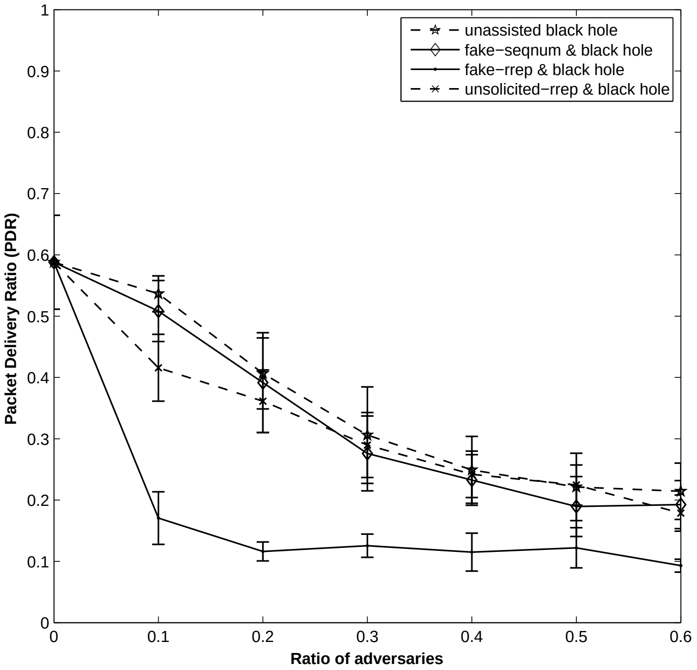


Figure 4.8: Packet Delivery Ratio achieved in simulations with varying numbers of adversaries employing one of a number of route invasion attacks.

4.2.3 Black hole attack

The black hole attack, in which flows are attracted towards it and dropped, was implemented in two stages. Firstly, the adversary awaits the arrival of a route request from a node. On receipt of a request, it elects to follow the route discovery procedure as normal or it replies with a fake route advertisement, using one of the route invasion techniques described in §4.2.2. By invading the route it reduces the time and increases the chance that the adversary will be selected to serve the route. Should the adversary start receiving data that requires forwarding from the requestor, the adversary simply drops the data. The description of the attack in Algorithm 6 assumes a route has been established from the source to destination, through the black hole.

Algorithm 6 `send` method for a black hole node

```
if if ip.src != netAddr then
    drop
else
    forward message: find next hop, or queue until next hop found.
end if
```

4.2.4 Grey hole attack

The grey hole attack is a variation of the black hole where the dropping behaviour of the adversary is more subtle. Variations of the attack implemented include:

1. Drop all data from a blacklist of target nodes.
2. Drop data probabilistically from a blacklist of target nodes.
3. Drop data probabilistically from all nodes.

In variations 1 and 2, the adversary only attempts to invade the route if the requesting node is in the blacklist. In variation 3, the adversary attempts to invade all routes. Once the adversary has captured the route, the adversary determines the probability by which it should drop the data packet from the blacklisted node, which determines whether the packet is dropped or not. In variation 1, the probability function is such that all data from all targets is dropped, should the grey hole node be required to forward it. In variations 2 and 3, a probability function determines when the adversary drops or forwards the data. The implementation of the grey hole is described in Algorithm 7.

Algorithm 7 `send` method for a grey hole node

```
if if ip.src != netAddr then
  check blacklist for ip.src and ip.dest
  if message is blacklisted then
    drop
  else
    forward message: find next hop, or queue until next hop found.
  end if
else
  forward message: find next hop, or queue until next hop found.
end if
```

4.2.5 Flooding Attack

A flooding attack involves any misbehaviour in which messages are fabricated by the adversary for the sole purpose of reducing throughput in the local area, or, if the messages are created in such a way that they are re-broadcast by neighbours, over a larger area. In RREQ flooding, the adversary generates messages for a fictional destination. The shape of the flooding attack is controlled by adjusting the burst, interval and pause times, which regulates the flooding behaviour. The dissemination area can be controlled through the use of an appropriate TTL. Each fabricated route request is created with an increasing RREQ ID which forces neighbours to process every RREQ received from the adversary, and broadcast it to their neighbours. Thus, based on the configuration of this misbehaviour the adversary can reduce network goodput by varying degrees.

4.2.6 Wormhole Attack

A wormhole attack requires adversaries to be equipped with multiple interfaces, where the second tunnel interface is of a higher throughput than the shared interface. The attack was implemented by extending the Jist/SWANS AODV router, `RouteAodv` (Lin, 2004), to be aware of multiple network interfaces. Route requests are forwarded over each interface that is not the loopback interface. As a result, routing table entries have an additional `interface_id` field which represents the interface to forward the data onto (Table 4.2). The underlying routing messages were not modified.

Table 4.2: A routing table entry for wormhole adversaries. The common fields of an entry are greyed out, whilst the `interface_id` field specific to multiple interface aware routing protocols (a requirement for wormhole adversaries) is emphasised.

Destination Address	Next Hop	Destination Sequence Number	Hop Count	Interface Id
D	M	200	2	1
...	...	...	...	...

In Figure 4.4, route requests arriving at adversary M1 (wormhole node) are broadcast over both interfaces; the shared interface and the tunnel to M2 (wormhole node). When a route reply is received (and in most cases, especially for larger hop counts, from M2), normal route reply processing occurs but now the interface that the reply was received is added to the corresponding route table entry. When M1 is requested to forward data, the `interface_id` and next hop address are used for onward data forwarding to the destination address. Algorithm 8 illustrates how we enumerate over each of the interfaces. This, in addition to the `interface_id` on route table entries, is all that is necessary to implement a multiple-interface aware AODV in Jist/SWANS.

4.3 Conclusions

In this chapter, the effect of selfishness, black holes, grey holes, wormholes, flooding and route invasion attacks in a MANET were investigated. As shown in §4.2.2.5, attack efficacy can be increased significantly for low percentages of malicious nodes by initially employing route invasion, to identify active flows or those in the process of being established, and to redirect them through the adversary. Most attacks required a number of pre-requisite

Algorithm 8 `receiveRouteRequest` method for a wormhole node

```
if is destination then
    generate route reply
else
    for all i in $interfaces$ do
        if $i \neq$ loopback then
            broadcast route request on interface i;
        end if
    end for
end if
```

conditions to be met, often done so by a period of eavesdropping to identify source-destination pairs and intermediate nodes. Implementation details of each attack were provided in the context of Jist/SWANS, and implemented in an adversarial router. The adversary's behaviour can be tuned at any point in simulation time, useful in this study as it allows intrusion detection systems developed for Jist/SWANS to be exercised over the full range of attacks. In Chapter 5, a number of detectors are described that can detect each of the attacks described here and are implemented to form the common services to be used in a novel approach in reasoning about detection events.

CHAPTER 5

COMMON SERVICES AND DETECTION STRATEGIES FOR A MANET INTRUSION DETECTION SYSTEM

5.1 Introduction

Typical intrusion detection systems employ a common architecture consisting of an audit stream, an analysis engine and a response component (Bace, 2000). Generally, information is acquired from a number of sources and compressed into a form suitable for analysis. The analyser generates detection events, which, based upon the rules and policies employed by the IDS, may retaliate.

The first vital requirement is to determine a *reliable* mechanism to gather audit data in order to observe misbehaviour. This chapter presents a number of audit sources in MANETs and details detection strategies based on the detailed description of misbehaviour described in Chapter 4. Each detector uses a number of common services to associate a detection event with a neighbour's identity. Detection strategies are distinguished by their use of *imperfect* or *perfect* audit data. Detectors using imperfect sources are susceptible to false positives whilst perfect schemes generate alerts only from the presence of misbehaviour in the network. The importance of differentiating these sources is motivated by a case study that compares the performance of a detector using imperfect audit data to one that is assisted by omniscient signalling. Lastly, a number of suitable responses are described.

5.2 Common Services

A number of data structures are defined and are maintained by the IDS, including a neighbour table and a list of known MAC-IP pairs, which are used by each detector introduced in this chapter.

5.2.1 Neighbour table

The IDS maintains a list of one-hop neighbours constructed from broadcast packets received. Each entry is keyed by a MAC addresses and a `timeout` that reflects the interval by which a neighbour is determined to be unreachable, computed in a similar manner to AODV, i.e. after *ALLOWED_HELLO_LOSS*HELLO_INTERVAL* Perkins (1997, §6.9). Entries that have timed out are removed and any registered detectors are informed of the change in order to update their state.

5.2.2 MAC-IP Pairs

The IDS maintains a list of the identities of network nodes by caching MAC-IP pairs from received packets. If a broadcast packet is received then the full identity of the sender is added to the MAC-IP table. If a unicast packet is received, each IP address listed within the packet is added to the table with a `null` MAC address. These partial entries are updated when any additional information becomes available (e.g. from receipt of broadcast packets). This allows the profile computed by the IDS on a partial identity to be re-assigned to the full identity when available. One of the common uses of this list is to enable detectors to resolve a MAC address with a known IP address, useful in the dropped packet detector as it prevents the raising of a false alert when a packet is being forwarded to the final hop, when the destination does not acknowledge receipt of the packet.

5.3 Audit source

Auditing produces an event stream that can be used in the analysis process. The IDS uses a combination of received packets, those received promiscuously and forwarded packets for analysis. Setting the MAC layer to promiscuous mode enables a node to receive all packets in its transmission range, including those not addressed to it, making an additional *imperfect* data source available. A number of alternative approaches to collecting audit data were evaluated, including the use of network layer acknowledgments (Balakrishnan *et al.*, 2005) and probing (Just *et al.*, 2003) but these introduce large overheads and are susceptible to targeted attacks by an intelligent adversary.

Promiscuous mode can be used to monitor the behaviour of neighbours *directly* and *indirectly*. In the latter case, this allows the IDS to monitor the behaviour of neighbours it has no association with. This can help to identify misbehaviour in denser networks where nodes have a large number of neighbours and it is likely the IDS could observe two or more forwarding notifications of a packet by neighbours wholly indirectly. Promiscuous monitoring has weaknesses though; it increases the energy expenditure and is imperfect in the sense that notifications will not always be received, due to ambiguous collisions, receiver collisions and asymmetric links (Marti *et al.*, 2000). An example of a collision is presented in Figure 5.2, which is a type of loss that should not be reacted to. Queuing delays and dropped packets resulting from an overloaded interface queue are outside the control of the monitored node but may result in a response being taken against it.

5.3.1 Generating an audit source in Jist/SWANS

An audit stream was generated in Jist/SWANS by passing packets from a number of contexts at the network layer directly to the IDS. Each node's MAC layer was initialised with the promiscuous flag (`promisc`) set to true, to enable all nodes to receive all packets in their transmission range, including those that are not explicitly addressed to them. The context of the packet was preserved by calling the `send` or `receive` procedure on the IDS, setting the `promisc` flag as appropriate and is passed to each detector in turn. A simplified figure showing this process is shown in Figure 5.3.

5.4 Detectors

Each audit data sample received from the network layer is passed into the IDS analysis engine containing a set of detectors, which immediately returns so the network layer can continue processing the packet as normal. Each detector then processes the packet immediately, caches the header or content for future use, or returns if it is of no interest. Each detector outputs notifications to the IDS asynchronously. Detection procedures are invoked outside the packet flow ensuring the IDS scales even for demanding network traffic. A block diagram of the analysis engine used which produces the notifications is presented in Figure 5.4.

The notifications are stored in a vector maintained by the IDS that keeps a count of each type for further processing at the end of an *interval*, upon which it is reset. The imperfect and perfect detection mechanisms were implemented in this thesis are presented in the subsections below.

5.4.1 Dropped packets detection

The detection of dropped packets is based on an imperfect detection mechanism introduced by Marti *et al.* (2000), which generates drop or forward notifications based on observations of the forwarding action of monitored nodes. Packets to be sent onwards are added to a buffer and packets overheard are used to determine if the packet in the buffer was actually forwarded by the monitored neighbour. This implementation was further developed to produce notifications based on whether the monitored packet is a control packet or data.

In *direct* monitoring (Figure 5.1a), a node maintains a time window of packets that it has recently forwarded to its neighbours. When a packet is to be forwarded, it is added to the window and assigned a timeout. If the timeout is exceeded without the monitoring

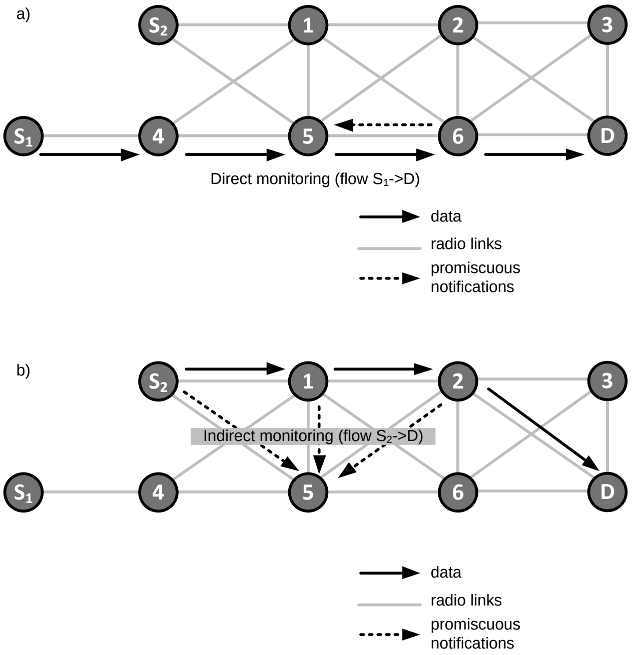

a)
S2
1
2
3
S1
4
5
6
D
Direct monitoring (flow S1->D)
data
radio links
promiscuous notifications
b)
S2
1
2
3
Indirect monitoring (flow S2->D)
S1
4
5
6
D
data
radio links
promiscuous notifications

Figure 5.1: a) Node 5 directly monitoring flow S1->D. b) Node 5 indirectly monitoring flow S2->D exclusively through promiscuous notifications.

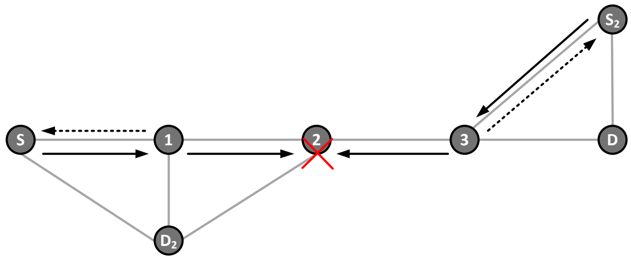


Figure 5.2: A collision at node 2, which is undetectable through promiscuous monitoring alone.

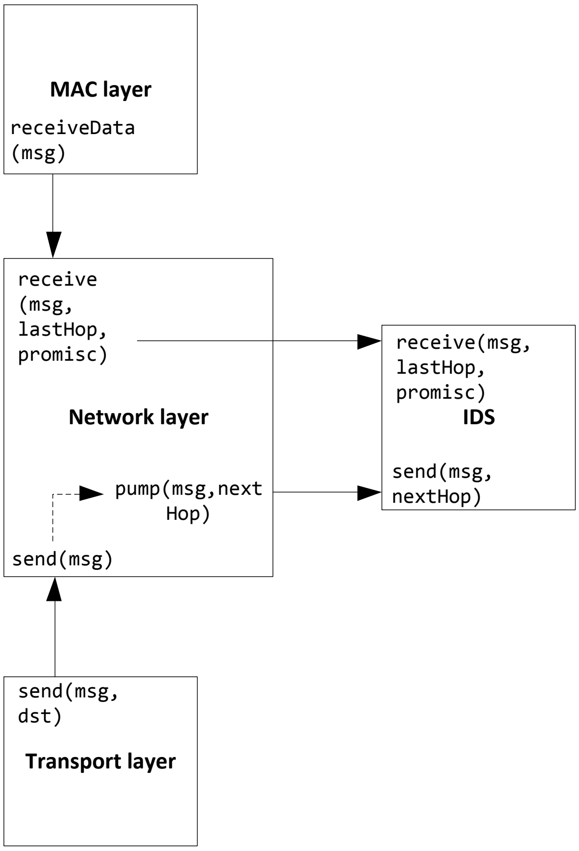


Figure 5.3: Simplified flow diagram showing audit data (complete with context) being sent to IDS.

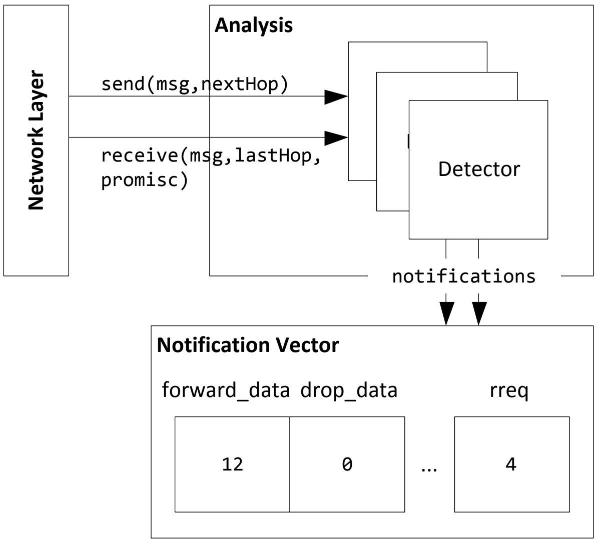


Figure 5.4: Analyser containing a set of detectors, operating on each audit data sample received by the IDS, and producing a set of notifications managed by a notification vector.

node having overheard the same packet being forwarded by the chosen next hop, the detector issues a drop notification. Otherwise, the monitored packet is removed, the timeout is cancelled and a forward notification is issued. The time window maintained by the dropped packet detector is illustrated in Figure 5.5. τ_n represents the duration between packet n being sent to a neighbour and it being consequentially overheard by the monitoring node as a result of the neighbour forwarding it onwards. Both routing and data packets are monitored in this way, thus the detector generates four possible notifications: `forward_data`, `drop_data`, `forward_routing` and `drop_routing`, thereby reporting positive and negative experiences with neighbours. The process by which notifications are generated from this detector is illustrated in Figure 5.6.

Indirect monitoring (Figure 5.1b), whereby promiscuously received packets are also added to the time window and the forwarding operation of the monitored node is verified is not possible using AODV since the required information is unavailable. It would be possible to do so with a source routing protocol by checking the forwarder corresponds to one of the nodes on the source route header of the forwarded data packet.

Algorithms 9–11 provide the core implementation of the dropped packet detector. Algorithm 9 illustrates the insertion of a packet into the time window. `nextHop` is the neighbour chosen to forward the packet, `pktEntry` is a data structure that maintains the state required to resolve a promiscuous notification consisting of a unique packet iden-

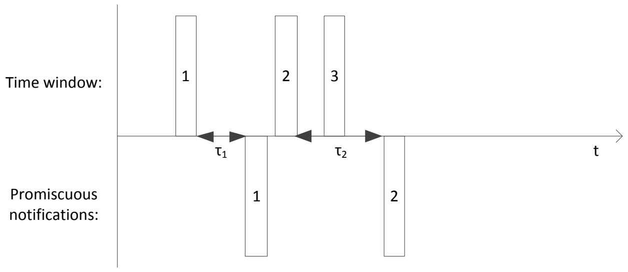


Figure 5.5: Time window maintained by the dropped packet detector, showing cached packet entries and promiscuous notifications received. τ_n represents the duration between packet n being sent and it being received promiscuously by the monitoring node as a result of the neighbour forwarding it onwards.

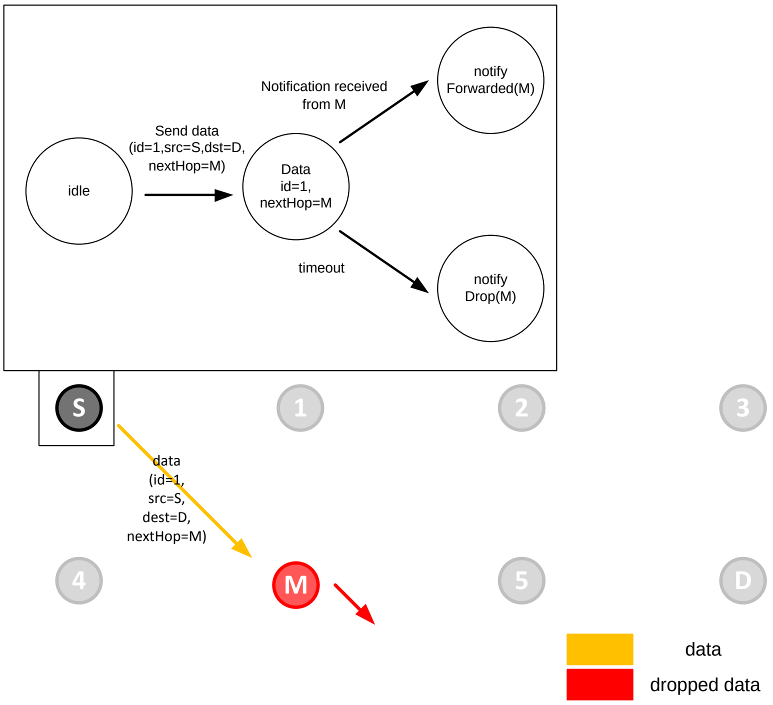

idle
Send data (id=1,src=S,dst=D, nextHop=M)
Data id=1, nextHop=M
Notification received from M
notify Forwarded(M)
timeout
notify Drop(M)
S
1
2
3
data (id=1, src=S, dest=D, nextHop=M)
4
M
5
D
data
dropped data

Figure 5.6: Detection of dropped packets from M by S, represented as a state transition diagram.

tifier, `id`, and the `nextHop`. `msg` is the packet to be sent. Algorithm 10 illustrates the actions that are taken when `msg` is overhead promiscuously and a forward notification is generated. Finally, Algorithm 11 provides the implementation of the timeout for each `pktEntry`, responsible for generating a drop notification should the packet not be overheard promiscuously.

Algorithm 9 `Watchdog.send(msg,intId,nextHop)` pseudo-code

```
if msg is unicast and nextHop is not msg destination then
    $pktEntry.id \leftarrow msg.id$
    $pktEntry.mac \leftarrow nextHop$
    $pktEntry.forwarded \leftarrow$ false
    insert packet entry $pktEntry$ into $window$
    assign timeout to $pktEntry$
end if
```

Algorithm 10 `Watchdog.receive(msg,lastHop,intId,promisc)` pseudo-code

```
if msg is unicast and msg received promiscuously then
    $size \leftarrow$ length($window$)
    for $i = 0$ to $size$ do
        $pktEntry \leftarrow window[i]$
        if $pktEntry.id = msg.id$ then
            $pktEntry.forwarded \leftarrow$ true
            $notifyForwarded()$
            return
        end if
    end for
end if
```

Algorithm 11 `Watchdog.packetTimeout(pktEntry)` pseudo-code

```
if pktEntry.forwarded = false then
    $notifyDrop()$
end if
delete($pktEntry$,$window$)
```

This detector differentiates between data and routing packets, which can assist the IDS in determining the type of misbehaviour and respond appropriately: a large number of dropped route replies may signify selfish behaviour, whilst a combination of dropped data and control packets may be the result of malicious behaviour. Additionally, the presence of wormholes can be determined by this detector, as if a wormhole node forwards data over a private wormhole interface, promiscuous notifications will not be received by the monitoring node, which would report a dropped packet. Lastly, to reduce false positives from undelivered packets, link layer feedback is used to disarm related entries in the time window, generated when the number of retries for a given frame transmission has been exceeded in 802.11.

5.4.2 Modified packets detection

A simple extension of dropped packet detection (§5.4.1) was developed to determine if monitored packets were forwarded without modification by the chosen neighbour. In this detection scheme, entries used in the time window for dropped packet detection have a hash of the IP packet and payload, generated from a computationally inexpensive hashing algorithm such as Message-Digest algorithm 5 (MD5) (Rivest, 1992). Should the monitor overhear the same packet from the monitored next hop, the hash of the packet in the time window is compared to the overheard packet. If both hashes are equal, the packet was sent without modification. If they are not equal, a `modify_data` or `modify_routing` notification is generated for monitored neighbours.

To ensure false positives are not generated, mutable fields such as the *TTL* (Time To Live) are excluded during the hashing operation. For routing packets, additional mutable fields are excluded that are updated during their processing by intermediate nodes (Perkins, 1997, §6.7), such as the *Number of Hops* for route replies.

5.4.3 Route Request flooding detection

AODV specifies an optional *RREQ_RATELIMIT* (Perkins, 1997, §6.9) constant which specifies the maximum number of RREQs a node should originate per second. Neither the default Ns2 or Jist/SWANS AODV routers rate limit RREQs, but simulations employing the basic scenario and up to 4 CBR flows (Table 3.6) showed that the number of RREQs originated per second by any node was considerably less than this limit. Thus, in flooding detection, each monitoring node keeps a count of the RREQs originated by each neighbour, and if the count for neighbour exceeds the maximum number of RREQs that can be originated over a given interval, a `flood_rreq` notification is generated.

5.4.4 Dropped Route Requests detection

The implementation of the dropped route request detector was motivated by selfish nodes, who purposely exclude themselves from serving routes in a number of ways. They could drop route replies indiscriminately or wait until the route is established and drop data, but this would be would be detected by the dropped packet detector (§5.4.1). Alternatively, they could indiscriminately drop RREQs.

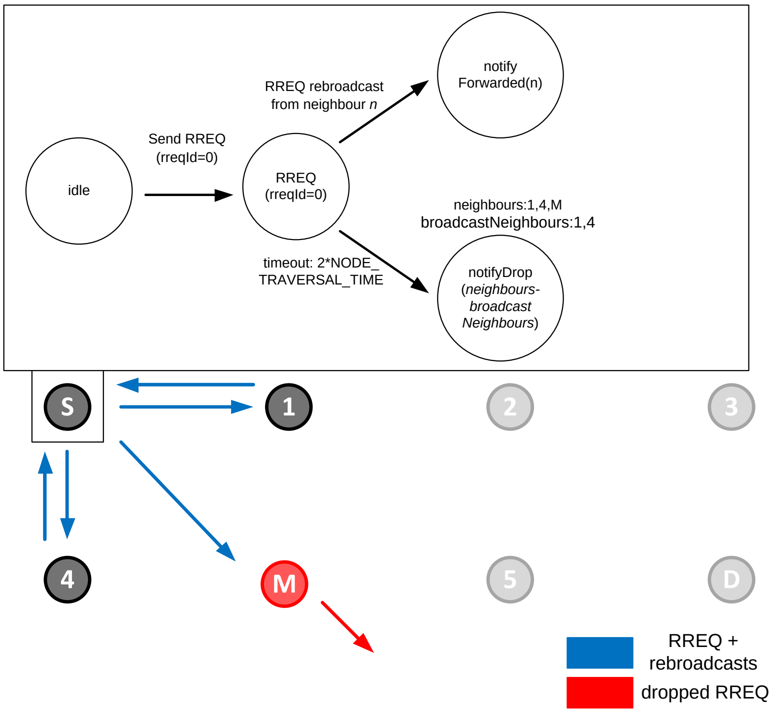


Figure 5.7: Detection of dropped route requests from M by S, represented as a state transition diagram. Assume that M is sending data, hence S knows its current neighbours are 1, 4 and M.

An idle selfish node is difficult to discover as it is invisible to neighbours' routing and network layers. If the selfish node is actively using the network, then an *imperfect* strategy to detect dropped requests can be devised, since the selfish node originates broadcast packets during route discovery and sends periodic HELLO messages to sustain the links to neighbours it is using. Any broadcast packets sent by the selfish node can be used by neighbouring nodes during detection to indicate the selfish node is a neighbour.

A selfish node can be detected *imperfectly* if the monitoring node observes recent broadcast traffic from the selfish node, but the selfish node fails to broadcast the monitoring node's own route requests. Whilst AODV discards re-broadcasted RREQs if they have already been processed, this information can be used to construct a list of rebroadcasters of the RREQ to be compared to the list of neighbours that *should* have broadcast the RREQ. Those nodes that should have broadcast the RREQ but are not in the list of rebroadcasters may have dropped the route request. The mechanism is illustrated in Figure 5.7.

To formalise this as an implementation, when a node originates a route request for a given destination, it caches this structure in a buffer, adding a timeout and the current set of neighbours to it. A `forward_rreq` notification is generated for those nodes observed to have rebroadcasted the RREQ. When the timeout elapses, a `drop_rreq` notification is generated for the remaining nodes.

Algorithm 12 expresses the actions taken by the IDS when a RREQ is originated, Algorithm 13 lists the actions taken by the RREQ originator when it receives the re-broadcasted RREQ from a neighbour, and Algorithm 14 shows the actions taken when the `entry` times out.

Algorithm 12 `Ids.send(msg,interfaceId,nextHop)` pseudo-code

```
if $msg.payload$ is RREQ and node is rreq originator then
    $RREQ \leftarrow msg.payload$
    $entry.origIp \leftarrow RREQ.origIp$
    $entry.rreqId \leftarrow RREQ.rreqId$
    $entry.neighbours \leftarrow$ ids.getMacIpPairs()
    insert rreq entry $entry$ into $rreqEntries$
    assign timeout to $entry$
end if
```

Algorithm 13 `Ids.receive(msg,lastHop,interfaceId,promisc)` pseudo-code

```
if $msg.payload$ is RREQ then
    $RREQ \leftarrow msg.payload$
    $size \leftarrow$ length($rreqEntries$)
    for $i = 0$ to $size$ do
        $entry \leftarrow rreqEntries[i]$
        if $entry.origIp = RREQ.origIp$ and $entry.rreqId = RREQ.rreqId$ then
            add $lastHop$ to $entry.broadcastNeighbours$
            notSelfish($lastHop$)
            return
        end if
    end for
end if
```

Like dropped packet detection, this scheme is imperfect and is prone to generating more false positives than dropped packet detection, as 802.11 broadcast messages are sent without acknowledgments and are unreliable (Lundgren *et al.*, 2002). Some of the possible reasons a monitor could falsely accuse a neighbour of dropping route requests include the following:

- the neighbour, though in range, may not receive the broadcasted RREQ message due to local congestion at the MAC layer,
- the neighbour may receive the broadcasted RREQ message but may be heavily loaded and have large queuing delays and may not rebroadcast the RREQ by the allotted time.

Algorithm 14 `Ids.rreqEntryTimeout(entry)` pseudo-code

```
remove neighbours in $entry.broadcastNeighbours$ from $entry.neighbours$
$size \leftarrow$ length($neighbours$)
for $i = 0$ to $size$ do
    notifySelfish($neighbours[i]$)
end for
delete $entry$ from $rreqEntries$
```

- the monitor could miss the observation of the re-broadcasted RREQ from the neighbour.
- The neighbour list maintained by the monitor may be out-of-date and include nodes that are no longer in communication range. This is why entries in the neighbour table (§5.2.1) are subject to a timeout equal to that which AODV uses to determine a next hop as unreachable.

Only RREQs **originated** by the monitor node are monitored by the IDS, purposely excluding forwarded RREQs. The reason for this is that it is not clear which neighbours have processed and forwarded the route request in the past. These neighbours will not re-broadcast the RREQ by the definition of the route discovery procedure (Perkins, 1997, §6.5), and would otherwise lead to the generation of a false positive.

5.4.5 Route invasion detection

5.4.5.1 Fake route reply as destination

In this type of route invasion, an adversary advertises a route as if it is the requested destination, even if it is not, or currently has no route to it. The adversary can make the advertisement more favourable to the target by appending an arbitrarily large destination sequence number onto it, to overwrite any existing entries in the target's routing table.

This attack can be detected by any of the next hop neighbours of the adversary. When a destination replies to a route request, the RREP's *Hop Count* field is set to 0. The IDS checks that the *Destination IP Address* field of the RREP payload is equal to the *Source IP Address* on the IP header. If it is not equal a `fake_rrep` notification is generated. This can be implemented as a simple rule, such as that illustrated in the flow diagram in Figure 5.8, which detects this attack.

5.4.5.2 Unsolicited route reply

An unsolicited route reply is a form of active route invasion where a malicious node sends an advertisement for a destination to a target node it observes as forwarding traffic to

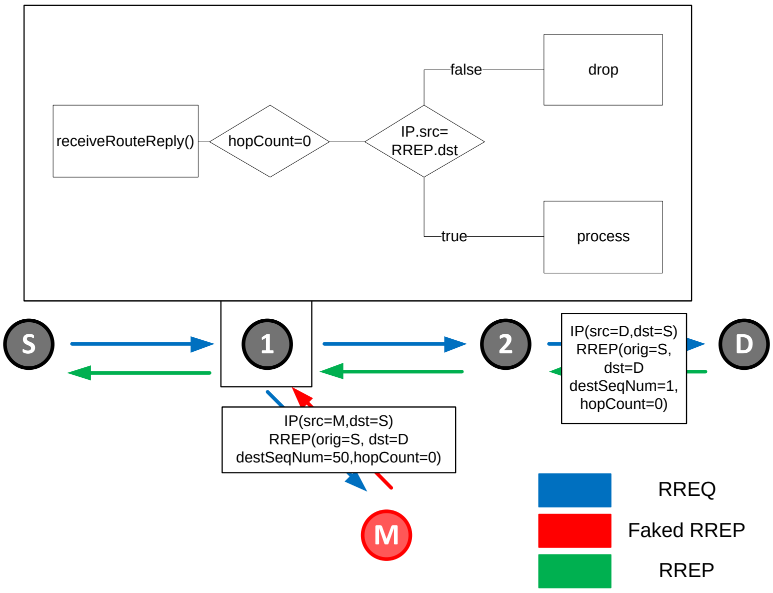


Figure 5.8: Detection of a faked route reply "as destination" from M by 1, represented as a flow diagram. In this case, 1 can safely drop the route advertisement from M, and send the true route advertisement (in green) to S.

that destination, in an attempt to overwrite the existing route table entry and redirect the traffic toward the adversary. The attack was described in detail in §4.2.2. The simplest form of the attack is to send an unsolicited route reply for the flow destination to the flow source via a chosen intermediate node. Though the flow source may have been sending traffic via a stable route and did not initiate route discovery for the destination, it processes the route reply regardless and uses it if it is more favourable.

To detect an unsolicited route advertisement *imperfectly*, the IDS defines a period of time after the request has been sent out that replies should be processed as normal. Once the request has been fulfilled, any further advertisements for the destination received by the requesting node generate an `unsolicited_rrep` notification unless as the result of requestor initiated route maintenance. Since it is entirely possible that multiple route advertisements may be received during normal operation of the network, a duration is defined by which notifications are suppressed.

To formalise this, all nodes store route discoveries for nodes in a buffer and during the period of time a node remains in there route advertisements are processed without notifying the IDS. Once an advertisement has been processed, the entry for the node is removed from the buffer after *NET_TRAVERSAL_TIME*, to allow for any other advertisements to arrive. Any arriving thereafter generate an `unsolicited_rrep` notification. The detection of this is illustrated in Figure 5.9.

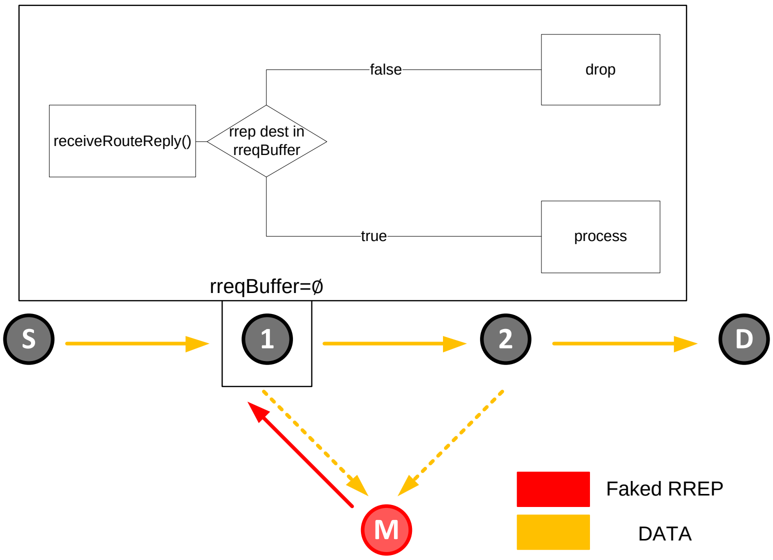


Figure 5.9: Detection of an unsolicited route reply from M by 1 using *sIDS*, represented as a flow diagram. In this case, 1 can safely drop the route advertisement from M, using the existing route with nextHop = 2.

5.4.5.3 Additional notes on route invasion

The correct maintenance of destination sequence numbers is key to the loop-free property of AODV (Perkins, 1997). In the above route invasion approaches, the hop count and destination sequence number fields are specifically targeted by the adversary and can introduce routing loops into the network, either as an intended or unintended side-effect. The propagation of a routing loop is shown in Figure 5.10.

So, in addition to being able to detect the forms of route invasion described in §5.4.5.1–5.4.5.2 and reacting instantaneously (to prevent the proliferation of false routing information network wide) routing loop detection is also advisable to maintain high levels of packet delivery. This can be implemented by creating a time window of monitored packets (e.g. §5.4.1) and checking if the received packet has been forwarded in the past.

5.5 Computing metrics

A 12-dimensional notification vector stores a count of each notification from the set of detectors for each neighbour n over the time period specified by $INTERVAL$ and represents the compressed form of the audit data, which is used for the purposes of intrusion detection. The structure of this vector is $D_n = \{$`forward_data`, `drop_data`, `forward_routing`, `drop_routing`, `modify_data`, `modify_routing`, `forward_rreq`, `drop_rreq`, `fake_rrep`, `unsolicited_rrep`, `rreq_count`, `rrep_count`$\}$. This compression reduces the computa-

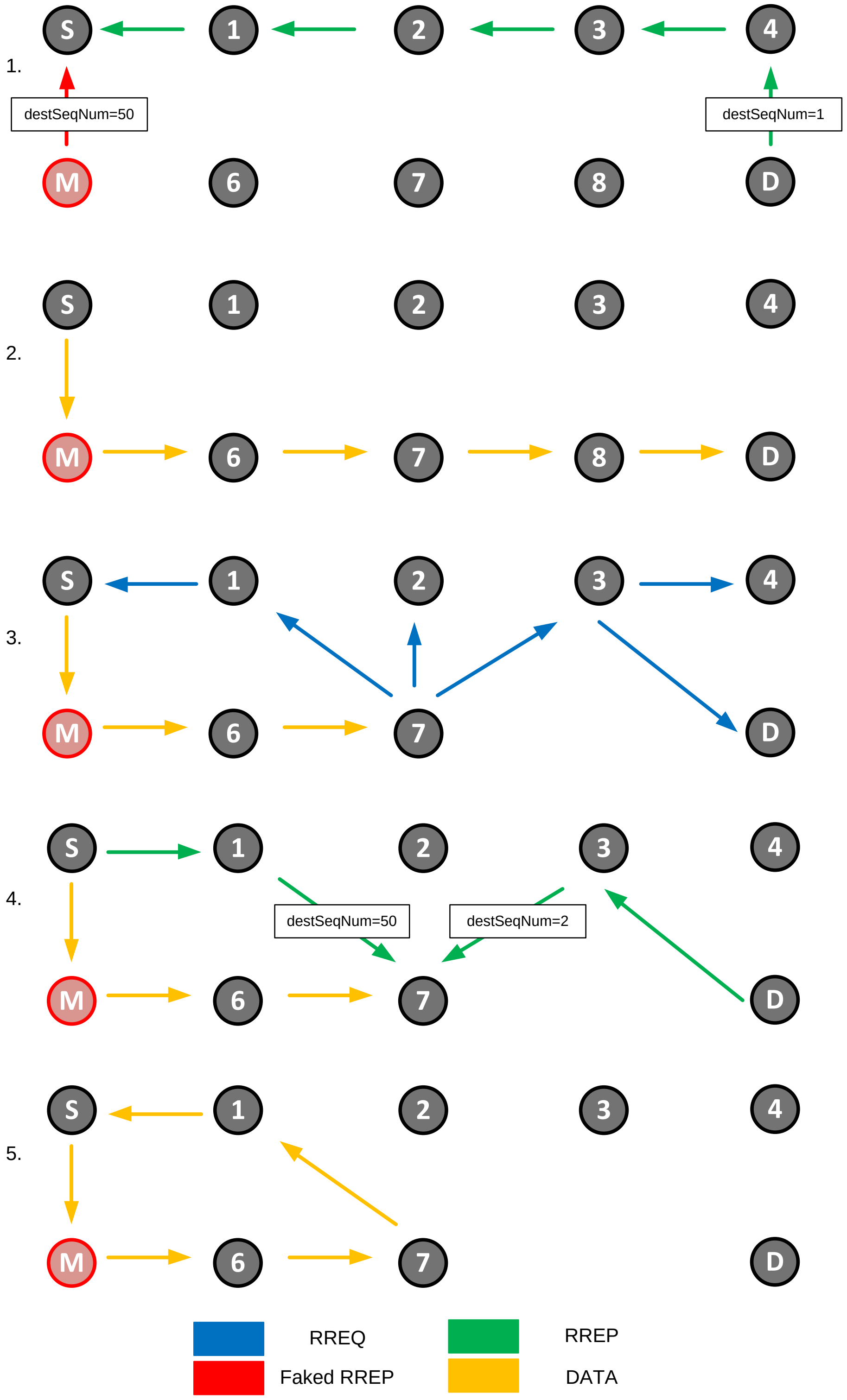


Figure 5.10: A routing loop caused by a route invasion attack. 1. and 2. A route invasion attack by M, who fabricates an advertisement with a high destination sequence number, *destSeqNum=50*. 3. Node 7, discovering that 8 has roamed out of range, initiates local repair for a new route to destination D. 4. Both D and S reply, but the reply from S has a larger destination sequence number, so is chosen. 5. This results in the routing loop shown.

tional burden and memory required to store the historical data about neighbours.

The notifications are transformed into a number of metrics defined as ratios that enable the IDS to discriminate between benign and malicious behaviour for a given function. Ratios with a value close to one signify benign behaviour for that function (e.g. forwarding data, forwarding data without modification or forwarding control packets) whilst a ratio close to zero indicates behaviour that deviates from the expectations of the routing protocol, and may be malicious. Each IDS computes an 8-dimensional vector (M_n) of metrics for the neighbour n, every *INTERVAL* seconds, from the notification vector about a neighbour n (D_n), generated above. M_n consists of the following elements:

1. Forwarded data rate (m_1) is a ratio of `forward_data` notifications to the sum of `drop_data` and `forward_data` notifications for n. $m_1 = 1$ means all monitored packets were forwarded by n.

$$m_1 = \frac{forward\_data_n}{forward\_data_n + drop\_data_n} \tag{5.1}$$

2. Forwarded routing rate (m_2) is a ratio of `forward_route` notifications to the sum of `drop_route` and `forward_route` notifications for n.

$$m_2 = \frac{forward\_route_n}{forward\_route_n + drop\_route_n} \tag{5.2}$$

3. Unmodified data rate (m_3) is a ratio of `modify_data` notifications to `forward_data` notifications for n. As n modifies more data packets, this ratio decreases.

$$m_3 = \frac{modify\_data_n}{forward\_data_n} \tag{5.3}$$

4. Unmodified routing rate (m_4) is a ratio of `modify_route` notifications to `forward_route` notifications for n.

$$m_4 = \frac{modify\_route_n}{forward\_route_n} \tag{5.4}$$

5. Forwarded RREQ rate (m_5) is a ratio of `forward_rreq` notifications to the sum of `forward_rreq` and `drop_rreq` notifications for n.

$$m_5 = \frac{forward\_rreq_n}{forward\_rreq_n + drop\_rreq_n} \tag{5.5}$$

6. Fake RREP rate (m_6) is a ratio of `fake_rrep` notifications to the sum of route replies originated by n stored in `rrep_count`.

$$m_6 = 1 - \left(\frac{fake\_rrep_n}{rrep\_count_n}\right) \tag{5.6}$$

7. Unsolicited RREP rate (m_7) is a ratio of the sum of `unsolicited_rrep` notifications to the total number of route replies originated by n, stored in `rrep_count`.

$$m_7 = 1 - \left(\frac{unsolicited\_rrep_n}{rrep\_count_n}\right) \tag{5.7}$$

8. RREQ flood rate (m_8) is expressed by the following function:

$$m_8 = \begin{cases} 1 & 0 \geq \texttt{rreq\_count} \leq RREQ\_RATELIMIT \\ 0 & otherwise \end{cases} \tag{5.8}$$

Once the metrics have been computed, D_n is reset to collect the notifications in the next interval. M_n is used by the IDS as the basis of any reasoning before it invokes a response.

5.5.1 Considerations when calculating metrics

Evaluating new values for each of the metrics in M_n is conditional on having the requisite notifications for n, determined by n's use of the network. Certain conditions may arise where no notifications of a particular type will be received in a period, for example, during light data loads or low mobility. In the first case, the IDS may not observe any data forwarded by n and thus be unable to compute m_1, which expresses the forwarding behaviour of it, whilst in the second case, the IDS will be unable to compute m_2, which expresses the forwarding of routing messages by n. In addition, as described in §5.4, some of the notifications are determined from imperfect detection methods, leading to a low evaluation for one or more metrics. This subsection details the approaches employed by the IDS to mitigate these problems.

5.5.1.1 Imperfect detectors

The detection methods that employ eavesdropping are liable to generate false notifications that may lead to the calculation of low valued metrics in one time period and cause an IDS using the instantaneous value of the metric to retaliate in the following period. To prevent knee-jerk reactions of this nature, each metric was smoothed by applying the following function:

$$new\, m_n = \alpha.old\, m_n + (1 - \alpha).latest\, m_n \tag{5.9}$$

This corresponds to each metric being computed from an effective memory of k samples, with the following relationship to α:

$$k = \frac{1}{1 - \alpha} \tag{5.10}$$

It was postulated that if α was set too low, the IDS would respond to honest nodes as a result of missed observations, whilst if α was too high, the IDS would become increasingly insensitive to misbehaviour. An appropriate value of α was determined by running several simulations in §6.6.1.

5.5.1.2 No requisite audit data

If the IDS is unable to compute a metric from the current notification vector, the new value is calculated from the last known sample and a small increment, δ, is added to it so that nodes are eventually re-introduced. Using m_1 as an example:

$$\lim_{forward\_data+drop\_data \to 0} m_1 = last\, m_1 + \delta \tag{5.11}$$

5.6 Responses

The IDS is equipped a number of responses to provide incentives to malicious and selfish nodes to co-operate whilst mitigating the spread of influence of misbehaviour and minimising the effect on network quality-of-service. The responses against a neighbour n are invoked by a reasoning scheme, such as thresholding, which maps the values of the metrics to zero or more responses. The responses considered in this work are:

1. RESPONSE_IGNORE_ALL: all packets originating from the suspect network or MAC address are ignored. This prevents the suspect from accessing the network entirely.

2. RESPONSE_IGNORE_DATA: the data originating from the suspect network or MAC address is ignored. This prevents the suspect from using the network for its own activities but does not stop it from contributing to the routing service, giving the suspect an opportunity to show it has corrected its behaviour.

3. RESPONSE_IGNORE_ROUTING: control packets originating from the suspect network or MAC address are ignored. If a suspect is accused of tampering or injecting false routing information, this prevents the IDS node from processing further advertisements and circulating this information network-wide. Route requests originating from the suspect are also ignored, which prevents the suspect from becoming a forwarder for the IDS node in order to find an alternative route.

4. RESPONSE_PURGE_ROUTE: the IDS removes all references to the suspect from the routing table. Any destinations that are served by the suspect as the next hop are removed, which when used with RESPONSE_IGNORE_ROUTING, can find an alternative route to the destination that does not include the suspect.

5. RESPONSE_CLEAR: clears any of the above responses and processes the packets with the suspect network or MAC address as normal.

5.7 Using imperfect detection mechanisms: A simulation study

Detectors relying on notifications from promiscuous monitoring or the receipt of broadcast packets often miss observations and generate false positive detections that could deny a benign node access to the network if the IDS retaliates. Whilst missed observations are acknowledged for their effect on IDS performance (Marti *et al.*, 2000), many simulated studies of IDS fail to mention the propagation environment they use (Marti *et al.*, 2000). More realistic propagation environments that employ fading are likely to compound the issue of missed observations.

This study aims to show that simple threshold-based IDS's exhibit good performance in simulations with ideal propagation environments, generating a small number of false positives, but in more realistic propagation environments, false positives increase dramatically; to motivate the importance of reasoning as a mechanism to improve IDS performance.

5.7.1 Method

An IDS using the dropped packet detector described in §5.4.1 was applied to a network with no misbehaviour in two network scenarios:

1. an ideal propagation environment with no fading and Freespace Pathloss,
2. with Rayleigh fading and Two-ray Pathloss.

In another simulation, an IDS using an *ideal* dropped packet detector was applied to the same environments. The ideal detector is sent an omniscient notification for every packet successfully forwarded by the monitored neighbour and correctly issues `forward` notifications where it would have otherwise missed the overheard packet from the neighbour. The dropped packet detector issues `drop` alerts for these missed observations even if the neighbour did forward the packet, as is commonly the case. Thus the ratio of alerts that are false detections from missed observations and those arising from the neighbour dropping the packet can be determined.

5.7.1.1 IDS architecture

The audit data stream for each IDS node was generated from promiscuously and non-promiscuously received packets and those queued for forwarding, as described in §5.3. A threshold-based IDS was equipped with the dropped packet detector or *ideal* dropped packet detector, producing a notification from the set {`drop_data, drop_routing, forward_data, forward_routing`} for each data and routing packet that required forwarding. The IDS was configured as shown in Table 5.1.

The metrics m_1 and m_2 were computed from the relevant notification counts for each observed neighbour n every *INTERVAL* seconds, as shown in §5.5. Once the metrics had

Table 5.1: IDS parameters used in simulation study

Parameter	Value
INTERVAL	1s
MIN_SAMPLES	5
METRIC_THRESHOLD	0.5

been updated a reputation value r_n was calculated for n and added to a reputation table managed by the IDS. r_n was calculated as follows:

$$r_n = \begin{cases} 1 & m_1 \geq METRIC\_THRESHOLD \\ -1 & m_1 < METRIC\_THRESHOLD \end{cases} \tag{5.12}$$

The default reputation assigned to n was zero. Until *MIN_SAMPLES* samples were collected, $r_n = 0$. The following response mapping was applied to n from mean value of r_n and defined as follows:

$$response(n) = \begin{cases} RESPONSE\_CLEAR & r_n \geq 0 \\ \{RESPONSE\_ALL, RESPONSE\_PURGE\_ROUTE\} & r_n < 0 \end{cases} \tag{5.13}$$

The response chosen from this mapping was applied for the duration specified by *INTERVAL*. Thus, if r_n dropped below zero, data and routing packets originating from n would not be processed or forwarded, whilst if r_n was greater than zero, the responses from the previous interval were cleared and packets originating from n were processed and forwarded by the IDS as normal.

5.7.1.2 Scenario

A simulated MANET was subjected to two different propagation environments: one with no fading and Freespace Pathloss, and another with Rayleigh fading and Two Ray Pathloss. This varies transmission radius of a node, so to keep the average route length constant, the transmission radius was determined and the simulation area adjusted appropriately.

To compute the transmission range of a node, a simulation sent 100 messages between two nodes placed at increasing distances from one another and reported the number of received messages at each of these distances. The results were averaged by using the Monte-Carlo approach described in §3.4. The graph shown in Figure 5.11 shows the PDR for each distance for the two propagation environments used.

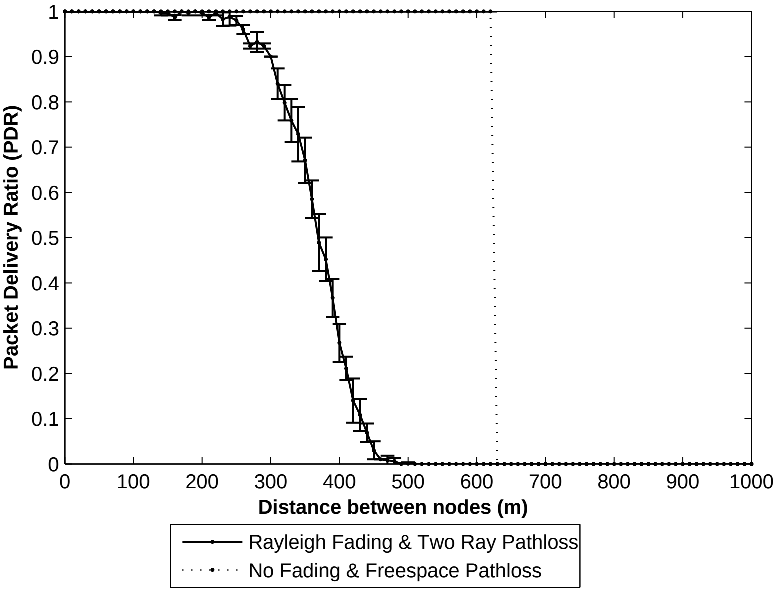


Figure 5.11: PDRs achieved over various transmission ranges (determined by `TransmissionRadius` driver) and propagation models.

Table 5.2: Transmission radius' chosen for different propagation environments, and field length required to keep average route length constant.

Propagation environment	Transmission radius	Field length
Freespace Pathloss, No fading	$r_{t_{fn}} = 600$	$len_{fn} = 2009$
Two Ray Pathloss, Rayleigh Fading	$r_{t_{tr}} = 365$	$len_{tr} = 3300$

To keep the average route length constant between the two environments, the field length was computed using the following formula whilst keeping the required number of nodes, $N = 77$, the average number of neighbours, $< n >= 8$, and reading the transmission radius obtained from the `TransmissionRadius` simulation. This produced the field lengths shown in Table 5.2.

$$len = \sqrt{\frac{N\pi r_t^2}{< n >}}$$

5.7.1.3 Simulation configuration

The simulation environment was configured as described by Perkins *et al.* (2001) and is listed in Appendix A. The properties in Table 5.2 override this environment to complete the definition of the Rayleigh fading and Two Ray Pathloss scenario, and Table 5.3 completes the definition of the No Fading and Freespace Pathloss scenario.

Table 5.2: Basic environment overridden to define 'Rayleigh fading and Two Ray Pathloss' simulation.

Simulation Property	Class	Property	Value
Fading	`jist.swans.field.` `Fading.Rayleigh`		
Pathloss	`jist.swans.field.` `Pathloss.TwoRay`		

Field		`len`
		`bounds`
		`nodes`
			len_{tr}
			len_{tr}, len_{tr}
			77

Table 5.3: Basic environment overridden to define 'No Fading and Freespace Pathloss' simulation

Simulation Property	Class	Property	Value
Fading	`jist.swans.field.` `Fading.None`		
Pathloss	`jist.swans.field.` `Pathloss.Freespace`		

Field		`len`
		`bounds`
		`nodes`
			len_{fn}
			len_{fn}, len_{fn}
			77

5.7.2 Results

Table 5.5 shows the number of false alerts generated by the different propagation environments, demonstrating that a simple change in propagation environment dramatically affects detection performance. Four times more false alerts were generated in the Rayleigh fading environment compared to no fading, resulting from more promiscuous observations being missed. Each time the ideal detector receives an omniscient `forward` notification when it was about to generate a `drop` notification from the use of promiscuous listening alone, a false alert is recorded.

Figure 5.12 illustrates the number of responses invoked by the IDS in the two environments with the drop detector and its ideal counterpart. A response count is ncremented each time the IDS takes an action against a packet originating from n. The most important feature of the chart is that the number of responses generated by the drop detector

Table 5.5: Average number of false alerts generated by detector network-wide

Scenario	Average false alert count	Standard Deviation
Freespace Pathloss, No fading	1228.17	391.47
Two Ray Pathloss, Rayleigh fading	5475.29	1055.29

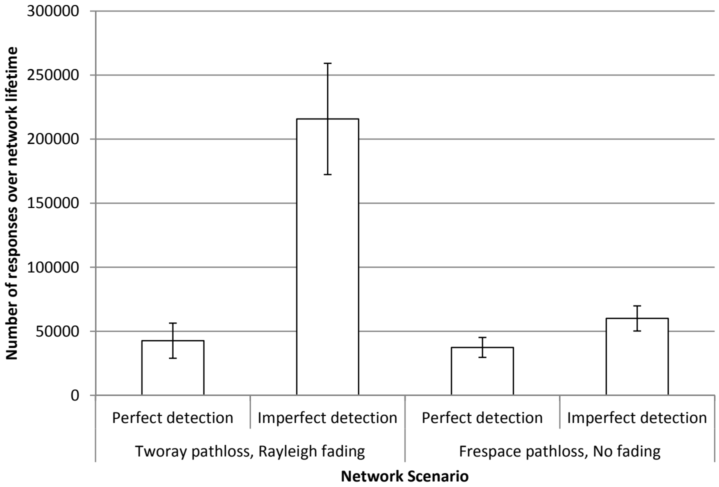


Figure 5.12: Number of responses in fading and ideal environment using a detector with perfect/imperfect signalling

using imperfect observations is higher in the Rayleigh fading environment as a result of the large number of alerts generated and this is not as a result of misbehaviour. Note that the response count from the ideal drop detector remains constant because the omniscient notifications negate the effect of changing the propagation model.

These results shown that even in the absence of misbehaviour, a typical threshold based IDS generates false positives, which increases dramatically in more challenging propagation environments. These corresponded to responses entirely toward other benign nodes, as there were no misbehaving nodes present in this scenario, motivating the need for additional reasoning in these environments.

5.8 Summary

In this chapter, a number of detection methodologies were described and implemented within Jist/SWANS as part of the basic services to support an intrusion detection system. The detectors used a combination of imperfect and perfect audit data, from packets addressed to a node, those overheard, and those to be forwarded.

A threshold IDS with an imperfect dropped detector was applied to networks with varying propagation environments, and its performance evaluated against the same IDS using a perfect dropped packet detector, demonstrating the negative effect that the propagation of alerts from missed observations can have on the IDS response. These effects were much more pronounced in the challenging propagation environment as the imperfect detector generated many more false alerts from missed observations. Thus, in more realistic environments, the ability to reason about alerts generated by detectors is both desirable and necessary. In Chapter 6, an approach to reasoning about the imprecision of certain observations provided by game theory will be explored and applied to network scenarios.

CHAPTER 6

SIDS AND GAME THEORETIC REASONING

6.1 Introduction

In Chapter 5 we showed that employing an IDS that responds directly to notifications from imperfect detectors can negatively affect the quality of service experienced by benign nodes within a MANET due to frequent retaliation by the IDS arising from false detections. Moreover, this effect worsens with more progressively realistic radio propagation environments as the reliability of overhearing neighbour transmissions decreases.

This chapter investigates the use of game theory within the IDS framework discussed in the previous chapter to develop a soft-Intrusion Detection System (*sIDS*) that reduces the likelihood of retaliating from false detections. This is achieved by developing a reasoning component in the form of a game that evaluates the quality of the metrics computed in §5.5 and determines if, given the evidence collected in the previous periods, a response is really necessary.

First, the performance of each detector was studied to determine its performance, which provided the basic grouping of the metrics into two sets. Then, the metrics were combined into a form suitable for a game formulation. Basic game theoretic definitions and terminology used were presented in §2.5. An implementation of the game-theoretic IDS is provided for the Jist/SWANS network simulator.

6.1.1 Desirable properties for a MANET IDS

The distributed nature of MANETs requires a suitable IDS possess a number of desirable properties, summarised in review papers by Deng *et al.* (2002) and Zhou & Haas (1999). Firstly, IDS should be decentralised such that each member node contributes to intrusion detection. In fact, any critical services responsible for promoting network *availability* such as certification and signing should not be centralised as doing so introduces a single point of failure that could be targeted by the adversary to disrupt network communications. Any IDS schemes employed should minimise the computational, memory and control overheads required since MANET nodes are often energy and compute constrained and communicate through low bandwidth wireless links. False detection rates should be minimised to preserve quality of service, and if a response is invoked, it should be appropriate to the type of adversary in order to incentivise it to co-operate.

sIDS applies additional reasoning in the form of a game within the IDS framework introduced in Chapter 5 and incorporates a number of the desirable properties described above. First, detection activities are distributed to each MANET node, which process and analyse audit data independently. Secondly, minimum state is maintained by the use of a sliding window that provides an effective memory (§5.5.1.1), reducing the memory overheads required significantly. Thirdly, adversaries are re-introduced over time if they are observed to have corrected their behaviour. The aim of the game theoretic component is to reason about metric quality, which should reduce the number of false detections in the imperfect radio medium, meeting a further desirable property.

6.2 Metric performance

The basic IDS framework introduced in Chapter 5 computes a number of metrics ($m_1 - m_8$) in a recent interval for each neighbour from notifications generated by a set of detectors (§5.5) that each analyse audit data for suspected misbehaviour by neighbouring nodes. Each metric is a normalised continuous variable, so $m_n \in [0, 1]$. Some of the metrics are computed from notifications by imperfect detectors and are liable to generate false positives in networks free of misbehaviour, whilst the remaining metrics are computed from notifications by perfect detectors that are guaranteed not to produce false positives. Imperfect schemes are important to provide additional information that would otherwise make certain types of misbehaviour impossible to detect, but their ability to cause an IDS to retaliate should be lower than from perfect detection schemes. Thus we grouped each metric into one of two sets: m_p consisting of metrics driven by perfect detectors and m_i consisting of those driven by imperfect detectors.

To evaluate the set a given metric belongs to, a series of simulations were configured as defined in Appendix A, with 100 nodes, Rayleigh fading and Two-ray pathloss. Each node was assigned a type from $\Theta = \{IDS, adversary\}$, selected at random according to the ratio of adversaries required. Adversaries misbehaved continuously over the period of time $t_s - t_e$ where $t_s = 50s$ and $t_e = 500s$.

IDS nodes were configured with an alert-only IDS, `IdsAlert.java`, which employed the full detector set described in §5.5. Detector notifications from IDS nodes and misuses from each adversary were logged to enable detection performance statistics to be extracted for each detector. Each detector was configured as shown in Table 6.1.

The results in Figures 6.1–6.2 illustrate the comparative performance of the imperfect

Table 6.1: Scenarios and detector configuration used to evaluate the performance of individual detectors.

Scenario	Detector	Detector Parameter	Value
Black Hole	DETECTION_DROP	timeout	1s
Modify Data	DETECTION_MODIFY	timeout	1s
Passive Route Invasion	DETECTION_FAKE_ RREP	-	
Active Route Invasion	DETECTION_ UNSOLICITED_ RREP	-	
RREQ Flooding	DETECTION_RREQ_ FLOOD	rreqRateLimit	10
Selfish	DETECTION_DROP_ RREQ	timeout	0.25s

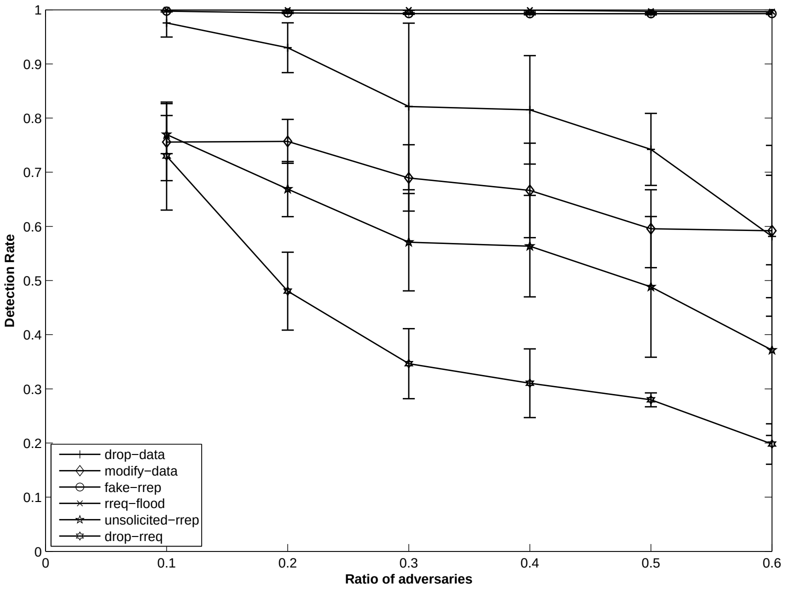


Figure 6.1: Detection rates of detection mechanisms at varying percentages of malicious nodes when applied to scenario detailed in Table 6.1

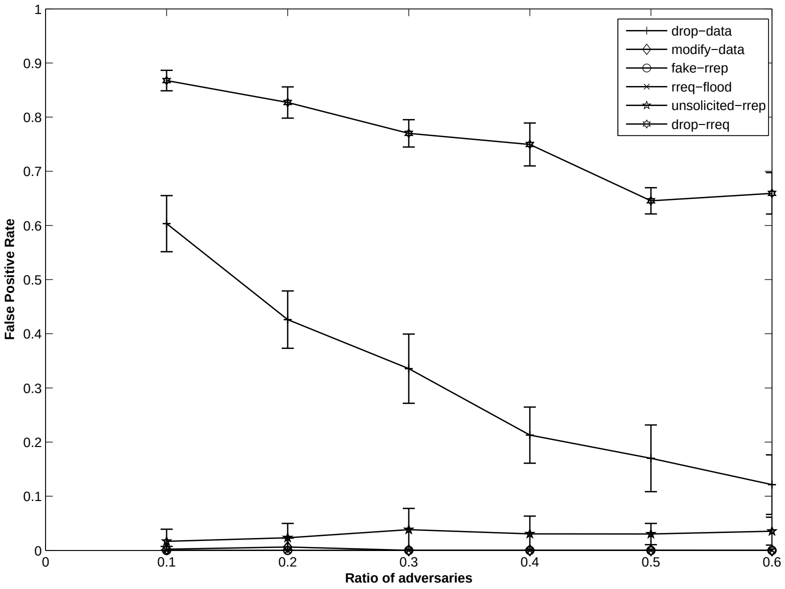


Figure 6.2: False positive rates of detection mechanisms at varying percentages of malicious nodes when applied to scenario detailed in Table 6.1

and perfect detectors, used to determine the members of m_i and m_p respectively. The set of imperfect metrics m_i consisted of forwarding rates (m_1 and m_2), forwarded RREQ rate (m_5) and unsolicited RREP rate (m_7). Of the imperfect detectors, *drop-rreq* performed worst with the largest false positive rate as a result of its reliance on overhearing rebroadcasted RREQs from neighbours, which is unreliable (as described in §5.4.4). The best performing imperfect detector was *drop-data* since data packets are unicast toward their destination and benefit from MAC layer acknowledgements. These acknowledgements enable the MAC layer to determine if a frame is undeliverable and is used by the *drop-data* detector to delete packets in its buffer that would otherwise trigger a dropped packet alert, reducing its false positive rate.

The set of perfect metrics (m_p) consisted of the modified rates (m_3 and m_4), the fake RREP rate (m_6) and RREP flood rate (m_8), compiled from detectors that demonstrated a zero false positive rate over the set of network simulations. Summarised below are the members of the sets m_i and m_p:

$$m_i = m_1, m_2, m_5, m_7 \tag{6.1}$$

$$m_p = m_3, m_4, m_6, m_8 \tag{6.2}$$

Metrics were combined to return a value suitable for inclusion into a utility function. The sigmoid function of equation (6.3) was chosen to generate an output over the range $|m_j| \to [0,1]$. Moreover, it discriminated over the full range of each of the independent metrics.

$$|m_j| = 1 - \frac{(\sum_{j=1}^{n}(1-m_n)^k)^{1/k}}{n^{1/k}}, k = 4 \tag{6.3}$$

Thus for m_i:

$$|m_i| = 1 - \frac{((1-m_1)^4 + (1-m_2)^4 + (1-m_5)^4 + (1-m_7)^4)^{1/4}}{4^{1/4}}, \tag{6.4}$$

and for m_p:

$$|m_p| = 1 - \frac{((1-m_3)^4 + (1-m_4)^4 + (1-m_6)^4 + (1-m_8)^4)^{1/4}}{4^{1/4}}. \tag{6.5}$$

6.3 Design of sIDS

sIDS aims to reduce the number of responses from false positives by evaluating the quality of the metrics maintained for each monitored neighbour. Intuitively, node misbehaviour from $m_p \to 0$ should result in a response being taken with absolute certainty. m_i corresponds to *possible* misbehaviour by the neighbour. Thus when $m_i \to 0$ but m_p remains high, a *benefit of doubt* should be applied, in which a probabilistic response is taken. As

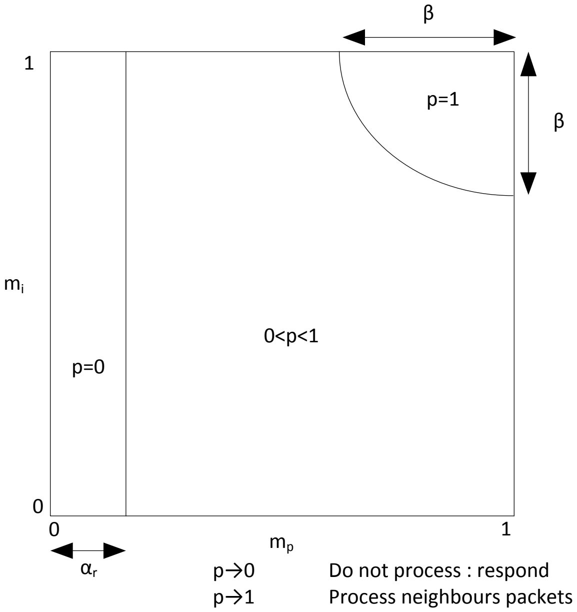


Figure 6.3: Phase-space of desired *sIDS* operation, where m_i is the calculated value from the set of imperfect metrics and m_p is the calculated value from the set of perfect metrics for an observed neighbour. p is the probability of processing the packet. If $p \to 0$, *sIDS* responds whilst if $p \to 1$, *sIDS* processes the neighbour's packets. Constants α_r and β control the thresholds at which respond or process is selected respectively.

m_p decreases, the benefit of doubt given to the neighbour diminishes. If m_p and m_i are both above a certain value, the monitoring node processes the neighbour's packet with certainty.

Figure 6.3 represents the desired behaviour as a phase space diagram, resulting in three distinct domains and the IDS choosing the following action over combinations of m_p and m_i:

$$action = \begin{cases} Process & 0 < m_p < \alpha_r \\ Respond & \beta < m_p, m_i < 1 \\ mix(Process, Respond) & otherwise \end{cases} \tag{6.6}$$

A game was designed to provide these modes of operation in which the IDS chooses an action based on its payoff given the current values of m_p and m_i for monitored neighbours.

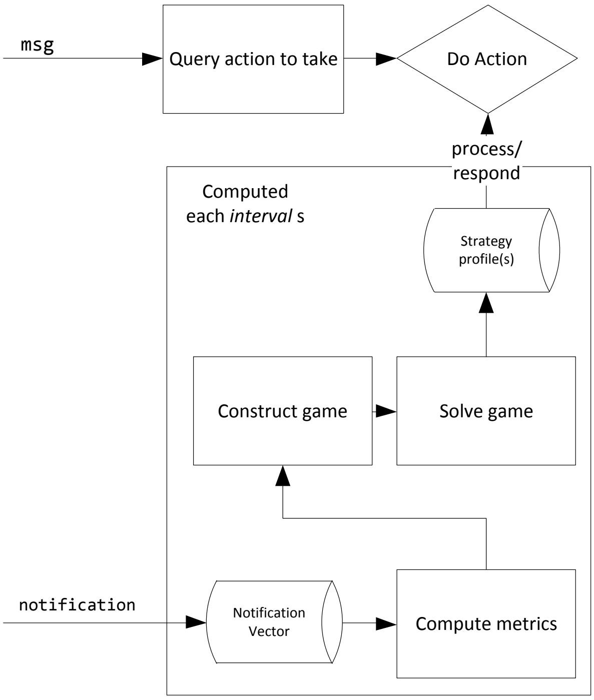


Figure 6.4: Flow diagram of *sIDS* in the context of a message being received by the network layer. Strategy profiles are computed each interval for each known neighbour, and are used to determine whether to process/respond to the current message.

The flow diagram shown in Figure 6.4 provides a brief overview of the resulting game-theoretic inspired IDS, entitled *sIDS*. At each interval, m_i and m_p are computed for each observed neighbour and a game constructed and solved using these metrics. As packets arrive at the network layer, they are passed to *sIDS* in parallel with the relevant application layer, to not introduce any further delays in network layer handling that would decrease the scalability and applicability of the proposed solution. Thus, IDS performs delayed processing of each received audit data, which may result in the processing of malicious payloads before they are detected as so. Appropriate responses have been designed to minimise the effect of these payloads.

The output of the game is a strategy profile that is cached in a look-up table and is applied for any decisions in that interval about packets originating a given neighbour. The generation of notifications by detectors and the procedure used for calculating metrics was previously described in Chapter 5 and §6.2 respectively. The construction of the game, the development of an appropriate game solver and the generation of appropriate strategy profiles is the subject of the remainder of this chapter.

Table 6.2: The bi-matrix game played by the IDS

		Suspect (j)	
		Not Attack	Attack
IDS (i)	Process	a, w	b, x
	Respond	c, y	d, z

6.4 Game construction

6.4.1 Game model

In *sIDS*, a **two-player**, **general sum** game in **normal form** computes appropriate strategy profiles for pairwise interactions between an *IDS* node, denoted as player i, and the *suspect*, denoted as player j. Two-player games are appropriate in intrusion detection scenarios as they minimise the computation required in a MANET environment. Very few algorithms currently exist to compute equilibria for N-player games (Shoham & Leyton-Brown, 2009). Since detection and response in this instance occurs locally, the use of such games is of limited value. However, it has been demonstrated that the use of N-player games could provide an advantage if coalitions of IDS nodes were permitted and detection was completed in an extended manner (Otrok *et al.*, 2007; Michiardi & Molva, 2005). General sum games provide more flexibility over the formulation of the payoffs than zero-sum games, and cater for more realistic scenarios where i's loss is not equal to j's gain or visa-versa.

In typical game-theoretic inspired IDS for MANET, the output of the game determines whether or not to detect (Patcha & Park, 2004), switch between lightweight or heavyweight intrusion detection schemes (Liu *et al.*, 2006; Otrok *et al.*, 2008b), or optimise the placement of IDS monitors in the network environment (Schmidt *et al.*, 2008), with the primary concern being to reduce the energy and computational requirements of intrusion detection. In *sIDS* detection is always-on, but a novel approach is proposed in which the game output determines whether the IDS should respond or process a packet from a neighbour given the metrics computed about that neighbour in the past.

The *adversary* can choose one of two actions, $A_j = \{Attack,\ NotAttack\}$, based on the IDS's computation of m_i and m_p from past observations of the *adversary*'s behaviour. The *IDS* also has two actions, $A_i = \{Process,\ Respond\}$. *Process* corresponds to normal packet processing, whilst *Respond* invokes a response that is appropriate to the context and the type of the packet. A pure strategy is the choice of an available action with certainty. In this and any other game there are also an infinite number of valid mixed strategies, each of which assigns a probability over the pure strategies for each player. If such a strategy results, *Process* or *Respond* is chosen according to the mixing probabilities returned. To meet the desired IDS operation, a pure strategy should be chosen at extremes of behaviour whilst a mixed strategy would be employed when the IDS observes the *adversary* as demonstrating possible misbehaviour.

The resulting bi-matrix game implemented by the IDS for players i and j is shown in Table 6.2. Utility functions for players i and j (u_i and u_j respectively) were constructed, describing the preferences of the IDS and suspect, and have been designed to ensure the modes of operation emerge as described in §6.3.

6.4.2 Construction of utility functions

To construct the utility functions for the IDS and the adversary, common game types were evaluated for properties that matched expected behaviour of both types of nodes in the three regions of the phase space shown in Figure 6.3. In these regions, the inequalities between payoffs differ, resulting in an appropriate game solution.

$m_p < \alpha_r$

If m_p falls below α_r, the IDS would prefer to play *Respond* and knowing this, a rational adversary would always choose *Not Attack.* Thus, the IDS plays the pure strategy *Respond* regardless of the adversary's action. In this case, *Process* is dominated by *Respond,* and *Attack* is dominated by *Not Attack.* The resulting game is shown in Table 6.3.

In order to produce this game, the following inequalities in the utility functions within the domain $m_p < \alpha_r$ are required:

$$c > a, \quad d > b, \quad w > x, \quad y > z \tag{6.7}$$

$m_p, m_i > \beta$

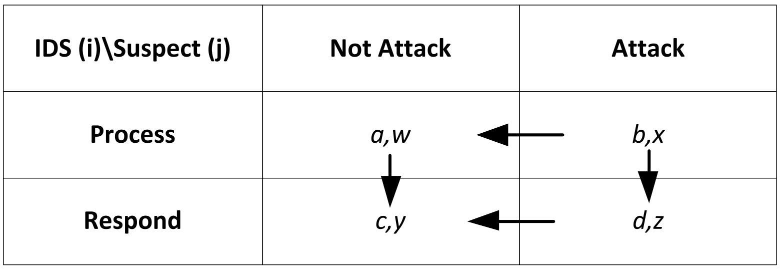
If m_p and m_i are both greater than β, the IDS would prefer to choose *Process* and the rational adversary would always choose *Attack.* Thus, the IDS plays the pure strategy *Process* regardless of the adversary's action. In this case *Respond* is dominated by *Process*, and *Not Attack* is dominated by *Attack.* The game played in this domain is shown in Table 6.4, and requires the inequalities in the utility functions within the domain $m_p, m_i > \beta$ shown in Equation 6.8.

$$a > c, \quad b > d, \quad x > w, \quad z > y \tag{6.8}$$

$\alpha_r \geq m_p \leq \beta, \ m_i < \beta$

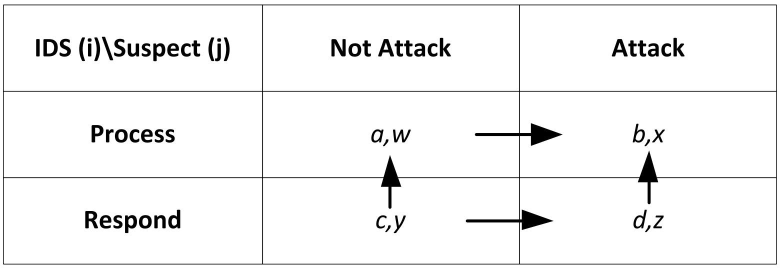
In this domain, there should be no pure strategy equilibrium for the IDS. Instead, a *discoordination game* emerges where the preferences for the IDS and adversary are directly opposed to one another (Rasmusen, 2001). From the perspective of the IDS the preferred strategy combinations are (1) *Process* when *Not Attack*, and (2) *Respond* when *Attack.*

Table 6.3: Game in domain $m_p < \alpha_r$



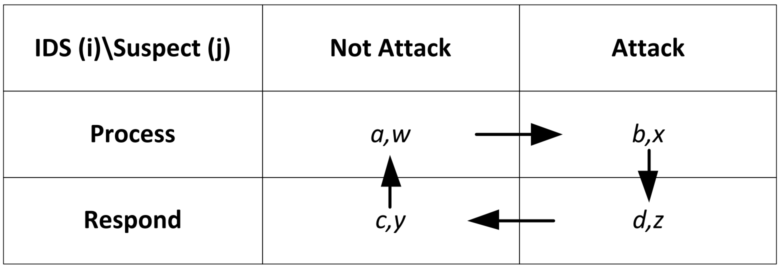
IDS (i)\Suspect (j)	Not Attack	Attack
Process	*a,w*	*b,x*
Respond	*c,y*	*d,z*

Table 6.4: Game in domain $m_p, m_i > \beta$



IDS (i)\Suspect (j)	Not Attack	Attack
Process	*a,w*	*b,x*
Respond	*c,y*	*d,z*

Table 6.5: Game in domain $\alpha_r \geq m_p \leq \beta,\ m_i < \beta$



IDS (i)\Suspect (j)	Not Attack	Attack
Process	*a,w*	*b,x*
Respond	*c,y*	*d,z*

From the suspect's perspective, as a malicious node, the preferred strategy combinations are (1) *Attack* when *Process*, and (2) *Not Attack* when *Respond.* The game played in this domain is shown in Table 6.5 and requires the following inequalities in the utility functions within the domain $\alpha_r \geq m_p \leq \beta,\ m_i < \beta$:

$$a > c, \quad d > b, \quad x > w, \quad y > z \tag{6.9}$$

A desirable property of a discoordination game is that there is a single equilibrium in mixed strategies, whose mixing probabilities are defined by the cardinal values of the payoffs (Rasmusen, 2001). Thus, as long as the preference structure is kept intact by enforcing the inequalities specified in Equation 6.9, the ratios between the absolute payoffs can be adjusted to result in an appropriate strategy mixture. This is further exploited to provide a *benefit of doubt* to an adversary, which falls in this mode of operation.

6.4.2.1 Finalisation of utility functions

The utility functions were developed further from the conditions of §6.4.2 to provide a *benefit of doubt* factor to those nodes who an IDS observes as having low m_i but high m_p and is represented by γ, increasing the benefit of the doubt as γ decreased, and visa-versa. Following this, numerical experiments were run using the payoff equating method of Rasmusen (2001) to identify suitable candidates for the utility function whilst preserving each inequality. This method yields the probabilities p and q that correspond to the mixed strategies employed by the IDS and adversary respectively:

$$p = \tfrac{d-b}{(d-b)+(a-c)}, \quad q = \tfrac{z-y}{(z-y)+(w-x)} \tag{6.10}$$

The absolute values of a, b, c and d were manipulated until the desired distribution of p was obtained. An instance of this distribution with tuning parameters α_r, β and γ is shown in Figure 6.5. The regions where $p < 0$ and $p > 1$, though impossible, show the combinations of m_p and m_i that offer only the required pure strategy solution.

6.4.3 Game bi-matrix

The game implemented in *sIDS* is shown in Table 6.6. It assumes the adversary is rational from the perspective of the IDS and has preferences that stay constant through the game. The adversary always prefers to (1) *Attack* when *Process* and (2) *Not Attack* when *Respond.* Since we wanted to manipulate the mixed strategy over *Process* or *Respond* obtained by the IDS, the utility function of the adversary (in the game played at the IDS) contains m_i and m_p.

If a game were to be played by the adversary, its own utility function would not contain m_i and m_p as these are *private* to the IDS. Instead, the adversary would make observations

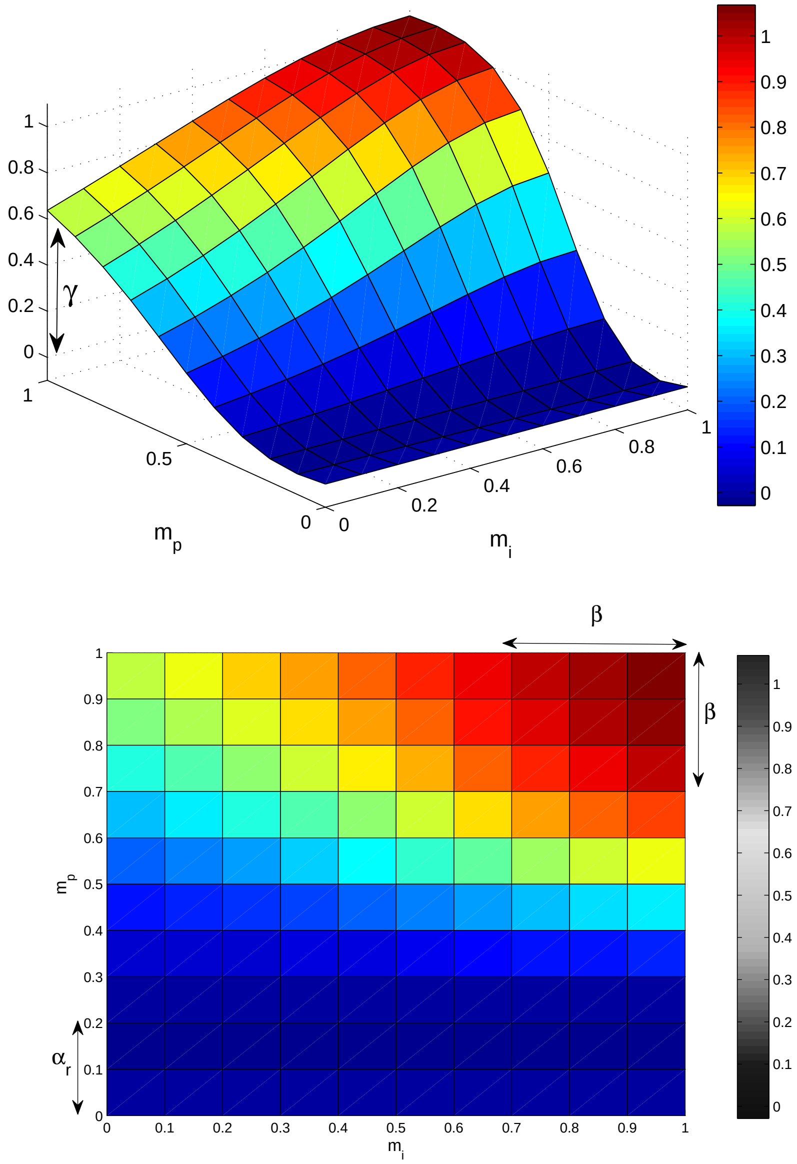


Figure 6.5: Distribution of p for various m_i and m_p, where $\alpha_r = 0.2$, $\beta = 0.7$ and $\gamma = 0.5$. Notice that $p < 0$ for $m_p < \alpha_r$ and $p > 1$ for $m_p, m_i > \beta$ produces appropriate pure strategy solutions.

Table 6.6: Game played by nodes equipped with sIDS.

		Suspect (j)	
		Not Attack	Attack
IDS (i)	Process	$1, 0$	$0, \gamma((1 - m_i)^2 + (1 - m_p)^2 - (1 - \beta)^2)$
	Respond	$0, \alpha_r m_p(m_p - \alpha_r)$	$1, 0$

on the performance of IDS (such as an estimate of detection performance) when deciding between *Not Attack* or *Attack*, or the mixture thereof. In this research, the adversary is rational but is not adaptive.

6.5 Solving the game

A number of solution concepts exist that predict the outcome(s) of game G given the players' utility functions. Simple solution concepts, such as dominant-strategy equilibrium, can be implemented with computationally inexpensive algorithms, but may not find a solution for the set of all general sum games.

The most well-known solution concept is the *Nash Equilibrium* (NE), for which many algorithms exist. The result of applying solution concepts is the generation of one or more strategy profiles that consist of the best response for each player in the game. A number of algorithms were reviewed in §2.5.1 to find NE. Minimax or payoff-equating (Rasmusen, 2001) are two such examples, and whilst these are computationally inexpensive, they only apply to a subset of game types; Minimax is only applicable for finding the equilibria of two-player zero-sum games whilst the payoff-equating method can produce erroneous results if a mixed strategy solution does not exist in the game (Rasmusen, 2001). Furthermore, algorithms specific to a subset of games can be non-terminating (Shoham & Leyton-Brown, 2009) or produce a single equilibrium when multiple equilibria exist, so special care needs to be taken when selecting an appropriate algorithm. Thus, a robust, terminating algorithm was required to find the equilibria of a general sum bi-matrix game.

6.5.1 The Nash Equilibrium

The Nash Equilibrium, proposed by John Forbes Nash (Nash, 1950), is a widely used solution concept in game theory. There is at least one NE in any general sum game (Rasmusen, 2001), which makes it a particularly attractive solution concept from a computational perspective. NE tests that a player's strategy s_i is the best response to the other player's strategies, defined by s_{i-1}. If a player cannot benefit by changing his strategy whilst the strategies of other players are fixed then the proposed strategy profile $s^* = (s_i^* ... s_{i-1}^*)$ is a NE.

NE is formally defined in Rasmusen (2001) as:

Definition. $\forall i, \pi_i(s_i^*, s_{-i}^*) \geq \pi_i(s_i', s_{-i}^*), \forall s_i'$.

The resulting equilibria can be in terms of pure or mixed strategies. For a pure strategy NE:

Definition. *The strategy profile $s_i^*, ..., s_N^*$ is a NE if for each i, s_i^* is a best response to the other players choices s_{-i}^*.*

For a mixed strategy NE:

Definition. *The mixed strategy profile $p_i^*, ..., p_N^*$ is a NE if for each player i, p_i^* is a best response to the other players choices p_{-i}^*.*

A *pure strategy* states that a player should play the given strategy with certainty, whilst a *mixed strategy* states the mixing probabilities that players should assign to their pure strategies, thus $p_i(s_i)$ is the probability i assigns to the pure strategy s_i.

6.5.2 Methods to find Nash Equilibria

A number of algorithms exist to find sample NE in two player games. The NE of a bi-matrix *zero sum* game can be found by applying the Simplex algorithm, in which the game is expressed as an optimisation problem within a linear program (Shoham & Leyton-Brown, 2009). Simplex is a pivoting algorithm that operates over a linear program in which the payoff matrices for both players have been transformed into a system of equations.

More challenging is finding the NE of a *general sum* bi-matrix game, which is no longer a straightforward optimisation problem and cannot be represented as a linear program. Instead general sum games can be represented as a Linear Complementarity Problem (LCP) and the Lemke-Howson (LH) algorithm employed. LH guarantees to find sample equilibria in a problem (Shoham & Leyton-Brown, 2009). The complexity class of LH is PPAD-complete (Polynomial Parity Arguments on Directed graphs) and means computing the equilibria by LH can take exponential time depending on the size of the game (Roughgarden *et al.*, 2007). Even though this is the case, few algorithms are as flexible or provide guarantees on producing Nash Equilibria as LH, making it appropriate for use within the game theoretic component of *sIDS*.

Lastly, and most challenging is to compute equilibria for general sum N-player games, which cannot even be represented as an LCP. Shoham & Leyton-Brown (2009) and Widger & Grosu (2009) propose the use of a sequence of LCPs (SLCP) which each correspond to an approximation of the problem and have designed a parallel NE enumeration algorithm suitable for use on multi-core architectures.

Table 6.7: Sample game where $m_i = 0.7$, $m_p = 0.5$, $\alpha_r = 0.2$, $\beta = 0.7$ and $\gamma = 0.5$.

		Suspect (j)	
		Not Attack	Attack
IDS (i)	Process (p)	$1, 0$	$0, 0.125$
	Respond $(1 - p)$	$0, 0.15$	$1, 0$

Table 6.8: Payoff matrices A and B used in tableaux construction

A	Not Attack	Attack
Process	1	0
Respond	0	1

B	Not Attack	Attack
Process	0	0.125
Respond	0.15	0

6.5.3 Generating Nash Equilibria using the Lemke-Howson algorithm

The Lemke-Howson algorithm (Lemke & J. T. Howson, 1964) finds NE on LCP formulations of a general-sum game and is similar in principle to the Simplex algorithm (Pritchard, 2007) as it uses iterated pivoting. The algorithm was implemented as a component within Jist/SWANS and computes a number of NE for payoff matrices of players i and j. Internally, the tableaux method described by Pritchard (2007) is used, which employs four steps to find the NE in the game. In this section these steps are described for an instance of the game developed in §6.4.3 shown in Table 6.7 to illustrate the operation of the algorithm.

6.5.3.1 Pre-processing

The bi-matrix game is first separated into payoff matrices A and B for players i and j respectively as shown in Table 6.8. To ensure the game satisfies the conditions required by LH, the payoff matrices cannot contain any negative values, thus a suitably large constant C is added to each entry. This keeps both the ordinality and cardinality of the preferences intact so the same equilibria will be computed from the pre-processed payoff matrices.

6.5.3.2 Initialising tableaux

Two tableaux are required for the two players in order to solve the game. The term r_i is the slack in the constraint $A^y \leq 1$ and s_j is the slack in the constraint $x^T B_j \leq 1$, so the following system is obtained:

$$\begin{aligned} Ay + r &= 1 \\ B^T x + s &= 1 \end{aligned} \tag{6.11}$$

Thus, the tableaux required are $r = 1 - Ay$, stated as Tableaux A and $s = 1 - B^T x$ stated as Tableaux B:

Tableaux A:

$$\begin{aligned} r_1 &= 1 - y_3 \\ r_2 &= 1 - y_4 \end{aligned} \tag{6.12}$$

Tableaux B:

$$\begin{aligned} s_3 &= 1 - 0.15x_2 \\ s_4 &= 1 - 0.125x_1 \end{aligned} \tag{6.13}$$

The r terms are the *duals* of the x's, whilst the s's are the duals of the y terms, also known as the *slack variables* in the system.

6.5.3.3 Pivoting

The pivoting process starts by arbitrarily choosing a variable x_i from the tableaux to bring into the basis. Then, a *minimum ratio test* determines the *slack variable* (or *dual*) to be removed by considering the coefficients of x_i, and the equation for the *slack variable* just removed is solved. The remaining equations are then solved in the chosen tableaux. The *dual* which left the basis determines the variable to enter the basis next.

The variable $x_1$is arbitrarily brought in, so by the *minimum ratio test*, s_4 leaves the basis, and solving s_4 for x_1 gives the following equation:

$$x_1 = -0.125s_4 + 8 \tag{6.14}$$

The variable x_1 is substituted into the remaining equations of Tableaux B, to produce:

$$\begin{aligned} s_3 &= 1 - 0.15x_2 \\ x_1 &= -8s_4 + 8 \end{aligned} \tag{6.15}$$

Since s_4 is y_4's dual, y_4 is brought in, and the pivoting process occurs once more, modifying Tableaux A in the process.

The procedure terminates when the initial variable chosen to enter the basis, x_i, or its *dual*, leaves. The resulting tableaux from this iterated pivoting are:

Tableaux A:

$$\begin{aligned} y_3 &= 1 - r_1 \\ y_4 &= 1 - r_2 \end{aligned} \tag{6.16}$$

Tableaux B:

$$\begin{aligned} x_2 &= 6.67 - 6.67s_3 \\ x_1 &= 8 - 8s_4 \end{aligned} \tag{6.17}$$

6.5.3.4 Nash Equilibria

To read the NE from the tableaux, the slack variables r_n and s_n are set to 0 and the resulting values in x_n and y_n are expressed as probabilities, resulting in the final form of the NE. Thus, Tableaux A becomes:

$$\begin{aligned} y_3 &= 1 \\ y_4 &= 1 \end{aligned} \tag{6.18}$$

and Tableaux B becomes:

$$\begin{aligned} x_2 &= 6.67 \\ x_1 &= 8 \end{aligned} \tag{6.19}$$

This results in the following Nash Equilibrium:

$$NE = \left[\frac{x_1}{x_1 + x_2}, \frac{x_2}{x_1 + x_2}, \frac{y_3}{y_3 + y_4}, \frac{y_4}{y_3 + y_4}\right] \tag{6.20}$$

$$NE = [0.545, 0.455, 0.5, 0.5] \tag{6.21}$$

Thus $p = 0.545$ and $q = 0.5$. To find additional equilibrium points (if any more exist), this process (§6.5.3.1–6.5.3.4) is repeated and a different initial variable x_i is chosen. If the solution differs from those currently in the list, it is added to the set of equilibria.

6.5.3.5 Validation of Lemke-Howson algorithm

In order to test the correctness the implementation of LH in Jist/SWANS, several general-form games with well-known results were created and the algorithm applied to generate the corresponding NEs. The expected NEs were computed by hand, through the game theory solver Gambit (McKelvey *et al.*, 2010), or from published results in the game-theoretic literature. The results obtained by the implementation of LH described above coincided with the expected results, and are shown in Appendix H.

6.5.4 Lemke-Howson algorithm applied to the game

The plots shown in Figure 6.6 show the distribution of p, the probability of the IDS choosing $Process$, for various m_i and m_p when the game of §6.6 is solved using the LH algorithm. The distribution corresponds to that achieved from the payoff-equating method (Figure 6.5), but the LH algorithm selects the relevant pure strategy equilibrium when $m_p < \alpha_r$ and $m_p, m_i > \beta$.

The game solver produces an array of equilibria that correspond to strategy profiles specifying a probability for each action over the IDS and adversaries' action set, A_i and A_j. In this case, there is a single equilibrium in mixed or pure strategies for any combination of

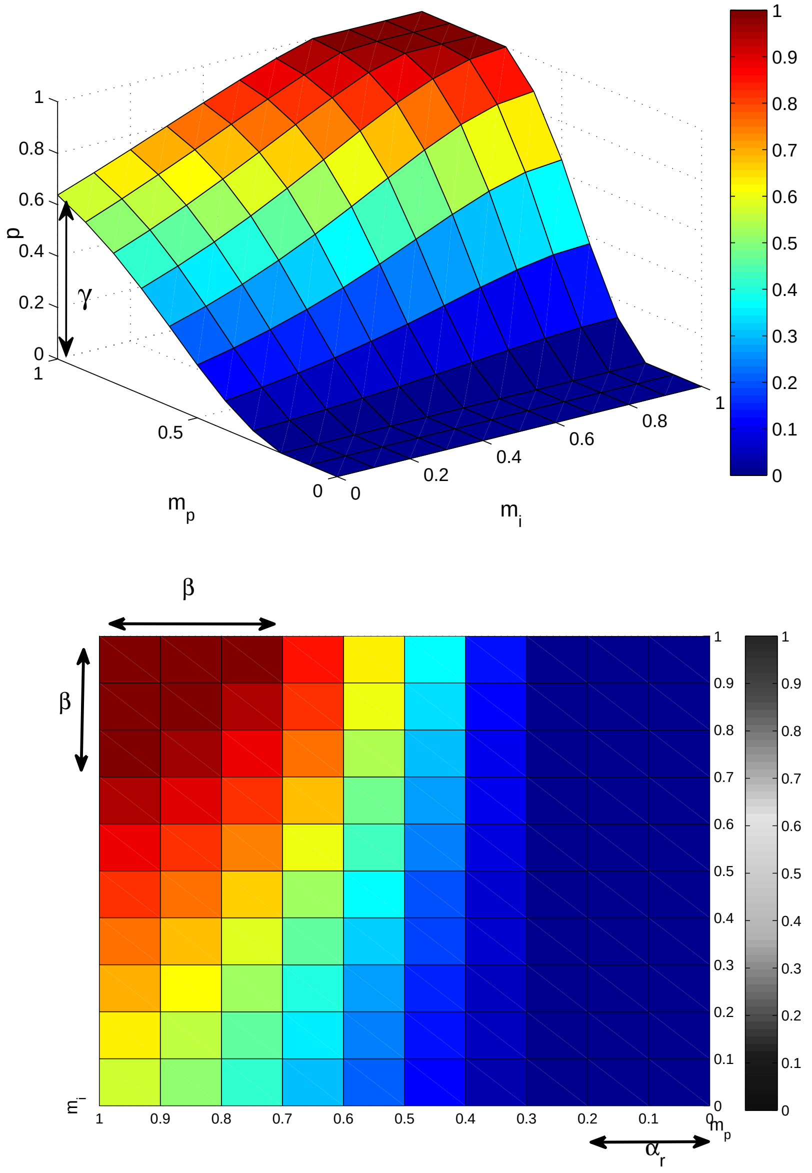


Figure 6.6: Distribution of *p* for various m_i and m_p, where $\alpha_r = 0.2$, $\beta = 0.7$ and $\gamma = 0.6$. Notice that $p < 0$ for $m_p < \alpha_r$ and $p > 1$ for $m_p, m_i > \beta$, producing appropriate pure strategy solutions in these regions.

$m_p \in [0,1]$ and $m_i \in [0,1]$. The strategy profile is cached in a look-up table and refreshed each *interval*. In the period between intervals, the pre-computed cached profile is consulted each time the network layer queries the IDS, to prevent additional computational burden from applying the LH algorithm.

6.5.5 Responses

The output of the Lemke-Howson algorithm is a NE that defines the best response for the IDS to take given the current observations m_p and m_i for each known neighbour. The NE defines a probability distribution over which *Process* or *Respond* should be chosen by the IDS. The `send` and `receive` processes of an IDS-equipped network layer queries the action it should take for each packet and responds in a manner appropriate to the current packet context (i.e. send/receive or forward) and type.

For this thesis, these responses are simple, but are effective for countering a large number of attacks, and are chosen from those described in §5.6. These responses are either *reactive* and apply on a per-packet basis, if a packet meets a given criteria, such as RESPONSE_IGNORE_DATA and RESPONSE_IGNORE_ROUTING, or, in the case of RESPONSE_PURGE_ROUTE are applied *proactively* at the start of the interval when the appropriate strategy mixture has been computed.

6.6 Simulation configuration

Network simulations were used to compare the performance of the game theoretic IDS relative to a baseline non-adaptive, threshold-based IDS. In addition, defenceless and omniscient networks were simulated to provide lower and upper bounds for the adopted metrics. Metrics were generated by post-processing the network trace output by Jist/SWANS with a number of scripts (described in §3.3), driven by the Monte-Carlo method introduced in §3.4, which terminated a study (the scenario and choice of independent variables) only when the metric(s) under observation had converged. This reduced the number of runs required, although a minimum of five simulation runs per study was specified to prevent early convergence.

A number of attacking scenarios were defined to have a broad range of local and distributed attacks. These are listed in Table 6.18. The environment detailed in Appendix A was used in each case. This section introduces the network model, the environmental parameters and node models used during network simulation. The scenarios used to evaluate the performance of the game theoretic IDS are also detailed.

6.6.1 Network Model

The simulated MANET is represented by the directed graph $G(V, E)$, where $V = \{v_0, ..., V_m\}$ is the set of MANET nodes and $E = \{e_0, ..., E_n\}$ is the set of directed links between the nodes. At simulation initialisation, $t = 0s$, the members of V are partitioned into two subsets: V_i is the set of IDS nodes, V_j the set of adversarial nodes. Membership of these subsets is decided at runtime by randomly selecting nodes according to the percentage of adversaries required, an independent variable controlled during simulation. Once nodes are assigned to V_i and V_j, the relevant protocol stack is created and they remain a permanent member of the subset for the duration of the simulation.

An edge e_n connects two vertices v_m and v_n if v_m is able to communicate with v_n, thus $(v_n, v_m) \in E$. Communication is asymmetric due to the use of Rayleigh fading, thus the reverse, $(v_m, v_n) \in E$, is not always true. A node v_i is able to overhear packets that originate from node v_j destined to v_k if v_j is in range of v_i.

6.2 Node models

A number of protocol stacks were developed to define the simulated models of defenceless, omniscient, IDS and adversary nodes. In each of the following cases only the network and routing layer instances differ between the types of node, but the full protocol stacks are presented for completeness. The source code for the layers used by each of the node models can be obtained by following the instructions in Appendix D.

Adversaries

Adversaries were created with the `RouteAodvMultiInterfaceAdversary` router and `NetIpAdversary` network layer. The adversarial routing layer provided a number of methods used to control misbehaviour during simulation (listed in Table G.1), whilst the network layer maintained a number of data structures used by the adversary during intrusion, such as an up-to-date list of neighbours and active data flows. The protocol stack of a simulated adversary is presented in Table 6.9).

Defenceless nodes

Defenceless nodes were created with the validated `RouteAodvMultiInterface` AODV router and the network layer, `NetIp`, and are represented by the protocol stack shown in Table 6.10.

IDS nodes

Each IDS node was created with the `NetIpIds` network layer and `RouteAodvMultiInterfaceIds` router and both provide a `respond` method that is invoked by the IDS when

Table 6.9: Jist/SWANS simulated adversary node protocol stack

Application	pkiddie.app.AppUdp
Transport	pkiddie.trans.TransUdp
Network	pkiddie.net.NetIpAdversary
Routing	pkiddie.route.RouteAodvAdversary
Mac	pkiddie.mac.Mac802_11
Physical	pkiddie.radio.RadioNoiseIndep

Table 6.10: Jist/SWANS simulated defenceless node protocol stack

Application	pkiddie.app.AppUdp
Transport	pkiddie.trans.TransUdp
Network	pkiddie.net.NetIp
Routing	pkiddie.route.RouteAodvMultiInterface
Mac	pkiddie.mac.Mac802_11
Physical	pkiddie.radio.RadioNoiseAdditive

Table 6.11: Intrusion Detection System implementation used in each scenario.

Node Model	IDS class used
Omniscient	`IdsOmniscient`
Baseline	`IdsBaseline`
Game Theoretic	`IdsGameTheory`

an intrusion is detected. IDS models implemented a common interface (`IdsInterface`), to allow code reuse and to simplify the implementation. Each IDS used in the scenarios that follow are listed in Table 6.11.

The IDS is created and managed by the network layer, which passes audit data directly to the IDS (as shown in Figure 5.3) as messages are sent or received. The IDS network layer maintains a number of data structures used during detection, such as a list of current neighbours and node identities, described previously in Chapter 5.

Omniscient Nodes

Omniscient nodes were created with an `IdsOmniscient` IDS, which comprises of a single omniscient detector that is notified by adversaries as they are about to misbehave. To achieve this, the strict layering imposed by Jist/SWANS was circumvented by creating a `Globals` class that served as a cache for global knowledge and was updated by adversaries directly. Through this perfect signal, omniscient nodes respond appropriately to all attacks from adversaries and are not susceptible to false alarms. This class was inspired by the General Operations Director (GOD) object in Ns2 that serves a similar purpose, storing global information about the network environment (Aad & Hubaux, 2004).

When an omniscient node is alerted to the presence of an adversary by the `Globals` class it retaliates immediately by ignoring any future transmissions originating from the adversary, and removes any entries that use this node as a next hop from the AODV routing table. Each omniscient node is pre-loaded with the full identity of all other nodes in the simulation.

Threshold IDS nodes

These nodes are created with a non-adaptive IDS (`IdsThreshold`) that employs a threshold to discriminate between benign and malicious behaviour; a common strategy employed by several published MANET IDS' (Vigna *et al.*, 2004; Marti *et al.*, 2000). The threshold IDS computes the metrics m_p and m_i from M_n each $INTERVAL$ for each known neighbour and applies one of the following actions to that neighbour during the next interval.

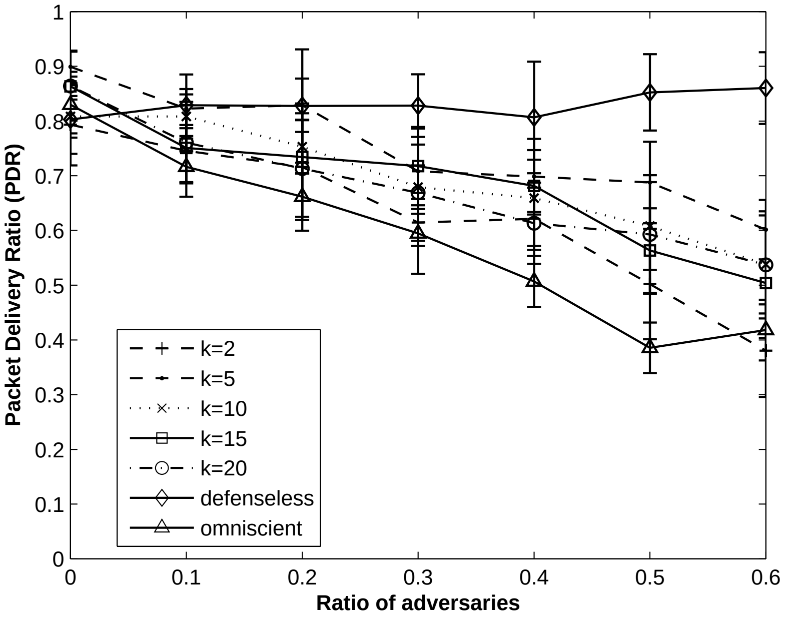


Figure 6.7: Packet Delivery Ratio (PDR) achieved in simulations with increasing ratio of 'black hole' adversaries vs. defenceless, omniscient and threshold IDS nodes with different window size k.

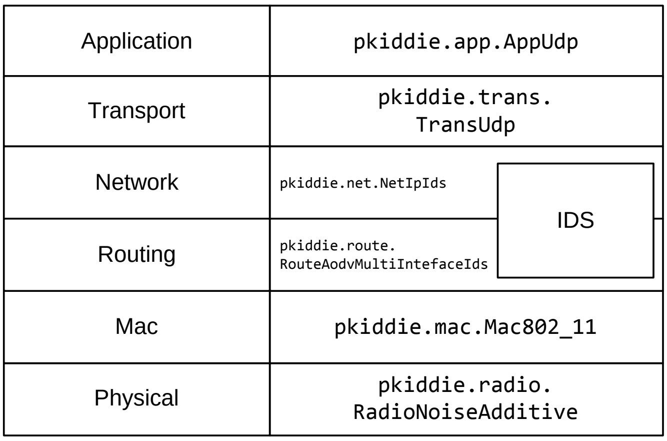
$$action = \begin{cases} Process & 0.5 < m_p, m_i < 1 \\ Respond & 0 < m_p, m_i < 0.5 \end{cases} \tag{6.22}$$

Thus, when m_p or m_i drops below 0.5, the IDS retaliates, removing any routing dependencies and ignoring any messages that originate from the neighbour. If m_p and m_i are greater than or equal to 0.5, the IDS clears any responses that may still be active from earlier rounds.

The computation of m_p and m_i is influenced by the choice of α, which provides an effective memory over each of the individual metric contributions, as described in §5.5.1.1. To find a suitable value of α, network simulations were set up employing nodes fortified with a threshold IDS subjected to misbehaviour from increasing numbers of black hole adversaries. The values of α shown in Table 6.13 were used.

Figure 6.7 shows the PDR achieved by the threshold IDS with different window size k, whilst Figure 6.8 shows the effect that adjusting k has on the response rate of the IDS. Together they illustrate that in general, using small values (e.g. $k \leq 2$) or large values (e.g. $k > 10$) reduces overall packet delivery. For small values this occurs because of the knee-jerk reactions that lead to a large number of false responses and the isolation of IDS nodes. For larger values, the reduction in PDR results from the insensitivity of the IDS to misbehaviour due to the long convergence time before the behaviour of the misbehaving

Table 6.12: Jist/SWANS simulated IDS node protocol stack



Application	pkiddie.app.AppUdp	
Transport	pkiddie.trans. TransUdp	
Network	pkiddie.net.NetIpIds	IDS
Routing	pkiddie.route. RouteAodvMultiIntefaceIds	
Mac	pkiddie.mac.Mac802_11	
Physical	pkiddie.radio. RadioNoiseAdditive	

Table 6.13: Effective window size in samples (k) and corresponding value of α used in simulations

Window size (samples) (k)	Corresponding value of α
2	$^1/_2$
5	$^4/_5$
10	$^9/_{10}$
15	$^{14}/_{15}$
20	$^{19}/_{20}$

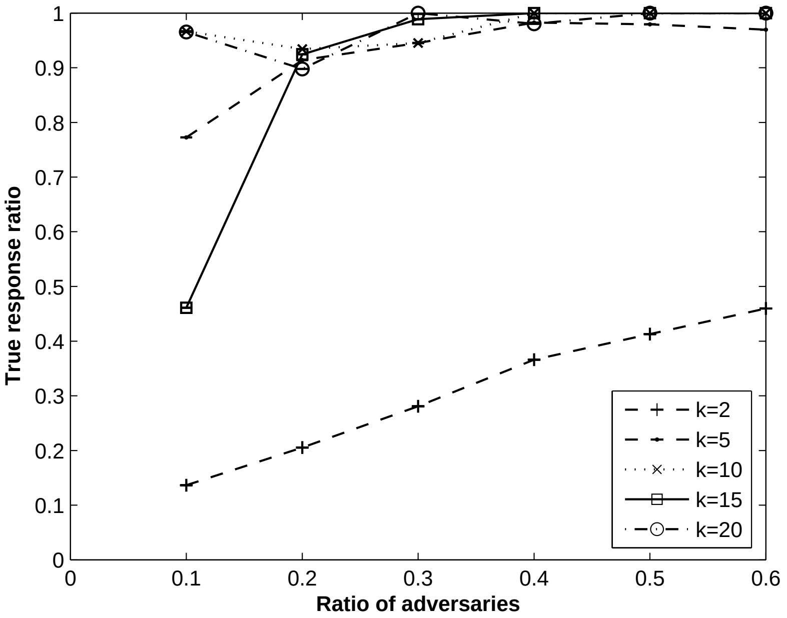


Figure 6.8: Response rate achieved in simulations with increasing ratio of 'black hole' adversaries vs. threshold IDS nodes with varying window size k.

Table 6.14: Threshold IDS parameters

Parameter	Value
INTERVAL (§*5.5*)	1s
α (§5.5.1.1)	0.8
δ (§5.5.1.2)	0.005

nodes is reflected by the individual metrics. Thus adversaries drop a larger percentage of traffic before being detected than with the lower values of k.

The value $k = 5$ ($\alpha = 0.8$) was chosen for its better performance over the full range of adversaries. For low ratios of malicious nodes the threshold IDS tracked the performance of the omniscient protocol, and for larger ratios it maintained the highest packet delivery ratios of all the values of k whilst showing few false positives. The parameters for the threshold IDS were set as shown in Table 6.14.

6.6.1.1 Game Theoretic IDS nodes

Game theoretic IDS nodes are created with an instance of `IdsGameTheory,` which uses the full set of detectors introduced in §5.4 to maintain m_p and m_i for each monitored neighbour. The game, described in §6.3, computes an appropriate strategy mixture that determines whether the node should process or respond to incoming traffic. The free parameters for the game theoretic IDS were set as shown in Table 6.15.

Table 6.15: Game Theoretic IDS parameters

Parameter	Value
α_r(6.4.2.1)	0.2
β(6.4.2.1)	0.7
γ(6.4.2.1)	0.6
INTERVAL (§*5.5*)	1s
α (§5.5.1.1)	0.8
δ (§5.5.1.2)	0.005

6.6.2 Simulation parameters

The basic simulation parameters are shown in Appendix A. The following sub-sections discuss some of the important parameters used in this simulation environment, the placement of adversaries and filtering of topologies to reduce the spread of metrics output from the simulation.

6.6.2.1 Radio environment

Rayleigh fading was employed to simulate missed observations and unidirectional links between nodes. Together with Tworay Pathloss, this produced the sigmoidal PDR shown earlier in Figure 5.11. Each node was created with the default `RadioNoiseAdditive` radio model, which used the Signal-To-Noise (SNR) threshold model.

6.6.2.2 Network environment

One hundred (100) nodes were placed randomly within a 1500x500m simulation area. Together with the radio environment, this meant that flow sources and destinations were on average more than one hop away from each other, which ensured that most traffic was routed to give the adversaries an opportunity to affect it.

6.6.2.3 Mobility

Mobility introduces further challenges to the IDS, especially in detection mechanisms that require an up-to-date list of neighbours. The random-waypoint model is a common model employed by numerous MANET studies and was employed and configured as shown in the basic simulation environment.

6.6.2.4 Flows

Ten Constant Bit Rate (CBR) flows were created and source-destination pairs were randomly assigned in the network. Flows were either evenly distributed between adversaries

Table 6.16: Flow properties for use in `AppUdp` traffic generator

Parameter	Value
size	512 bytes
start	10s
burst	*END*-10s
interval	0.25s
type	udp
reps	1

and IDS nodes, or IDS nodes only. A node is a source for at most 1 flow, so for N simulation nodes, a maximum of $N/2$ flows could be created. Each flow lasted the duration of the simulation and had the properties specified in Table 6.16. Each flow terminated 10 seconds before the end of the simulation to ensure each data packet was fully traced.

6.6.2.5 Topologies

Topologies were randomly generated and then filtered to limit the effect of disconnected flows on resulting network metrics. The transmission radius in the chosen environment was determined empirically to be $r_{t_{tr}} = 365\text{m}$ (§5.7) and was used to discard any topologies where any one flow was disconnected to reduce the spread in the computed metrics. To enable this, Dijkstra's algorithm (Dijkstra, 1959) was applied to an undirected graph representation of the network using the JUNG library (JUNG, 2010) to check that for every simulated flow there was at least one path between the source and destination. The `DijkstraDistance` class was used, which returned a numeric value for pairs of nodes with at least one path between them, or `null` if no path existed.

6.6.2.6 Adversary placement

Adversary nodes were placed using the `MaxBetweenness` adversary assignment model, which ranked each node in terms of their node *betweenness*, using the undirected graph representation of the network. The betweenness of a node v reflects the number of shortest paths between pairs of nodes in the field that include v, so nodes with a higher betweenness are more critical in the current topology. For the undirected graph $G(V, E)$, the betweenness $C_B(v)$ for vertex v is

$$C_B(v) = \sum_{s \neq v \neq t \epsilon V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{6.23}$$

where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a vertex v (Brandes, 2001). Adversaries are picked from a ranked list of highest to lowest *betweenness* to ensure that the adversaries chosen have the largest influence possible on the studied metrics.

6.6.2.7 Adversary Misbehaviour

An `AdversaryMisbehaviour` class manages the selection and activation of misbehaviour during the network simulation, and is configured by setting the misbehaviour types available to the adversary (described in Table G.1), the duration of the misbehaviour and pause time between instances. An active misbehaviour may comprise of a number of misuses. For example, each RREQ sent in a flooding attack is a single misuse.

Each misbehaviour activation is recorded with a network-wide unique identifier used only during post-processing. Misbehaviour instances cannot overlap, which makes the post-processing easier without sacrificing flexibility. Thus when an adversary is idle it can activate `rreq_flood` and `passive_route_invasion` simultaneously, but cannot activate another type of misbehaviour if it is already misbehaving. The defaults for each type of misbehaviour are shown in 6.17, and were used in each of the scenarios.

6.6.3 Scenarios

To evaluate the performance of the game theoretic IDS over a broad range of local and distributed attacks, a number of suitable attacking scenarios were designed, and are listed in Table 6.18. Each scenario was repeated for the node models described in §6.6.1, from the defenceless network to each fortified network.

6.7 Results

The percentage of adversaries in the network was the only independent variable used in the simulations, and was varied from 0% to 60%. For each of the scenarios, PDR, average end-to-end (ETE) delay and control overheads were computed (previously defined in §3.3). In addition, response rate and recovery time were used to evaluate the performance of each IDS for moderate numbers of adversaries (defined as 30%). Where necessary, additional metrics have been defined to show the effectiveness of an attack when its effect was not obvious from these base metrics alone.

6.7.1 Flooding

The detection technique used to determine RREQ flooding attacks was described in §5.4.3. Briefly, IDS nodes detect flooding if neighbours originate more than *RREQ_RATE_LIMIT* RREQs for any destination per second. Adversaries that flood the MANET create a large number of RREQs for real or fictitious destinations. This causes nodes to generate a large

Table 6.17: Misbehaviour defaults used in simulations.

Misbehaviour	Parameter	Value
Black Hole	packetType	udp
Modify Data	packetType	udp
Passive Route Invasion	mode	forge-rrep-as-dest
	baseSeqNum	100
Active Route Invasion	mode	forge-rrep
	baseSeqNum	100
	numHops	3
RREQ Flooding	burstTime	1s
	pauseTime	10s
	interval	0.067s
Selfish	-	
Wormhole	interfaceId	2

Table 6.18: Attacking scenarios simulated using each node model

Scenario	Attack
local/perfect	Flooding (§4.1.1.5)
local/imperfect	Unassisted Black Hole (§4.1.1.2)
local/(perfect+imperfect)	Black Hole + Passive Route Invasion (§4.2.2)
distributed/imperfect	Wormhole (§4.1.1.4)
random	Random

Table 6.19: Flooding parameters.

Parameter	Value
duration	1s
pauseTime	50s

amount of control traffic in an attempt to satisfy the RREQ, which reduces the throughput available for data.

In this scenario, the simulation time was reduced to $t = 250s$ as the flooding created an explosion in the size of the output network traces and compute time. The flood parameters were also tuned and are shown in Table 6.19.

Packet Delivery Ratio

Figure 6.9 shows the PDR achieved by each of the node models when subjected to a flooding attack with an increasing number of malicious nodes. In the defenceless network, the attack results in an average decrease of 76.3% of the maximum PDR when 60% of the nodes are adversaries. The omniscient network is also susceptible to the effects of the attack purely as a result of the localised contention hot-spots from the flood attack, which results in larger MAC losses.

In the defenceless, baseline and game theoretic networks, the attack efficacy saturates at around 30% adversaries. A possible reason for this is that at this point the generation of any more flooding traffic exceeds the offered load of the network and results in a large number of flooding packets being dropped at neighbours' queues. The baseline and game theoretic IDS are largely ineffective at preventing the drop of PDR aside from a small advantage at the 10% adversary ratio.

Average end-to-end (ETE) delay

The additional flooding of control packets introduces a large, linearly increasing ETE delay, which saturates between 30% to 40% adversaries (Figure 6.10). The maximum increase of ETE delay is shown in the defenceless network where flooding introduces a ten-fold increase at 40% adversaries compared with no adversaries.

The advantage offered by the IDS for small ratios of adversaries is more difficult to see here; no IDS significantly reduces the ETE delay from that of the defenceless network. The susceptibility of the omniscient protocol is also evident here: even though these nodes drop flooding packets immediately, the local contention effects from flooding are unavoidable and are demonstrated by a five fold increase beyond 30% adversaries.

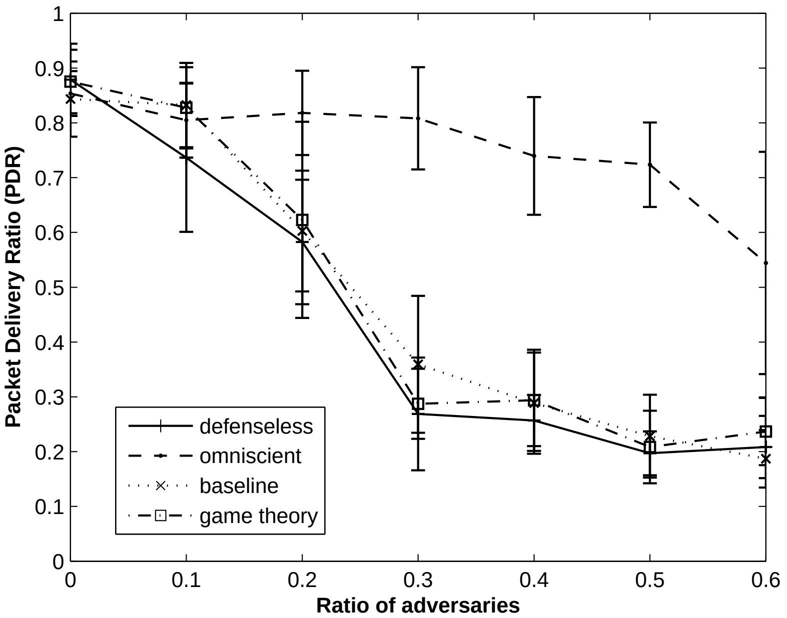


Figure 6.9: PDR achieved in flooding scenario with increasing numbers of adversaries

Control overheads

Figure 6.11 shows the total routing overheads independent of those generated by adversaries from the flooding attack. The total number of overheads increases as the ratio of adversaries increase, regardless of the network type. A portion of the routing overheads in each of the networks arise from additional contention introduced by flooding, which means fewer HELLO packets are delivered to neighbours resulting in a large number of unnecessary link breaks from AODV's route maintenance procedure. RREQs become the largest contributor to the routing overheads in this scenario, route discovery dominates, and the routes formed are unstable.

The baseline and game theoretic networks add an additional overhead through their attempted response to the flood attack. In addition, the contention leads to greater difficulty in observing the forwarding capability of neighbours leading to a large number of reactions to other IDS nodes.

Response Rate

The contention and extra delays introduced by flooding diminishes the ability of the IDS' to discriminate between benign and malicious behaviour. Most responses are directed towards IDS nodes as a result of imperfect detectors firing frequent false alerts (§6.2),

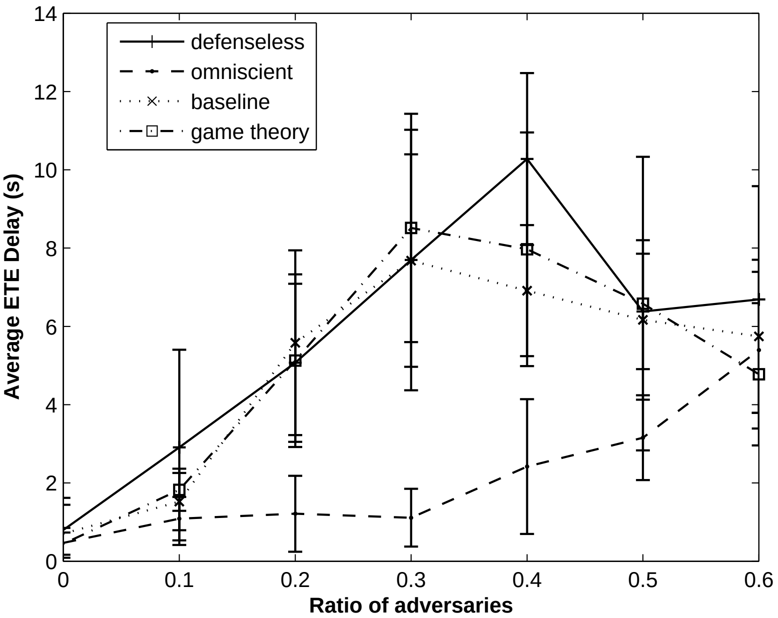


Figure 6.10: Average end-to-end delay achieved in flooding scenario with increasing numbers of adversaries

for example, those that use eavesdropping to verify the behaviour of neighbours. This is evident from Figure 6.12, where the baseline IDS consistently outperforms the game theoretic IDS by up to 20%, but still cannot achieve true response ratios greater than 50% until at least 30% adversaries are present.

Recovery Time and Minimum Goodput

The severity of the flooding attack is shown by the average recovery time for the defenceless network, showing a 1s flood takes 40s for the network to stabilise to pre-intrusion levels of goodput for moderate numbers of misbehaving nodes (30% adversaries). The small advantage in running the baseline or game theoretic IDS is depicted in Figure 6.13. Whilst the additional overheads generated by the responses from the baseline and game theoretic IDS reduce the average minimum goodput of the flooded network compared to the defenceless network, the baseline IDS recovers approximately 25% quicker than the defenceless network. The game theoretic IDS recovers on average around 10% quicker than the defenceless network.

6.7.1.1 Discussion

Flooding leads to congestion and medium contention that defeats the baseline and game theoretic IDS for anything larger than a small minority of adversaries. This is evidenced by the relatively small advantage gained by running either IDS in any of the performance me-

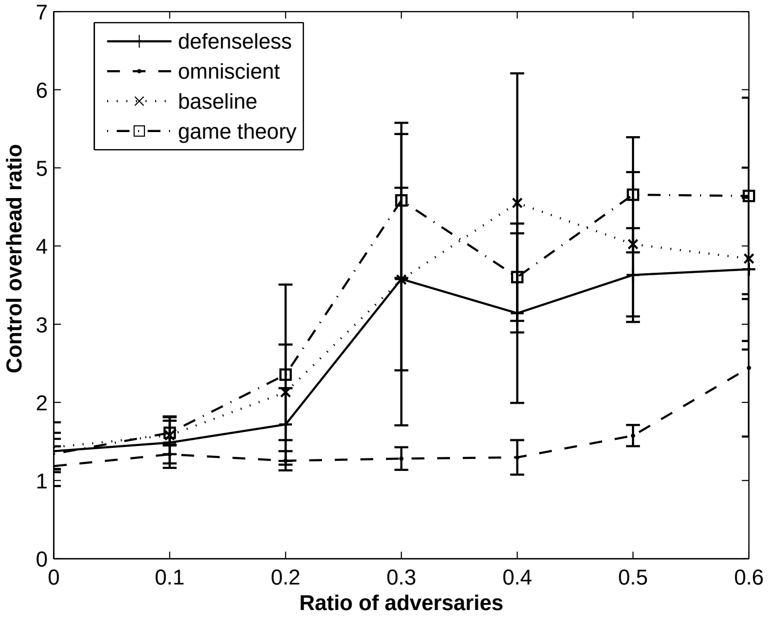


Figure 6.11: Control overheads in flooding scenario with an increasing numbers of adversaries

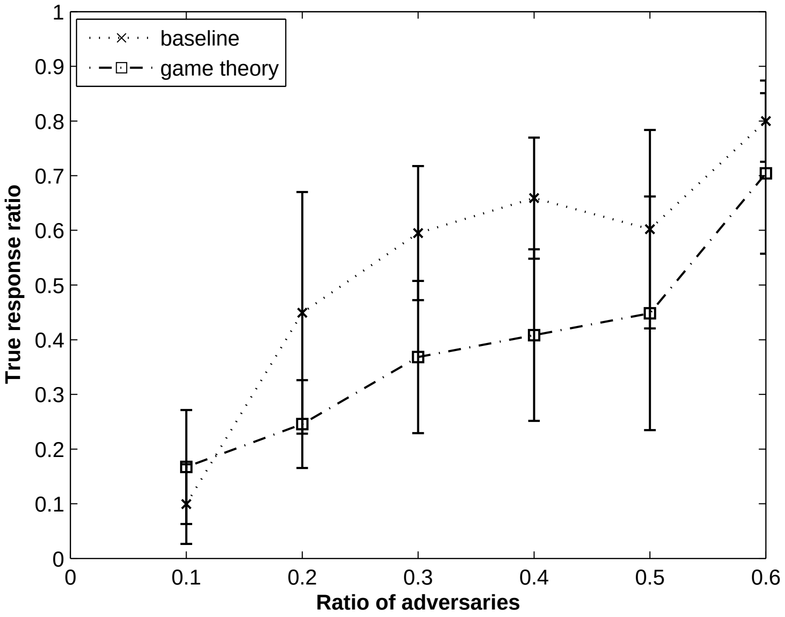


Figure 6.12: True response rate in flooding scenario achieved by baseline and game theoretic IDS with increasing numbers of adversaries

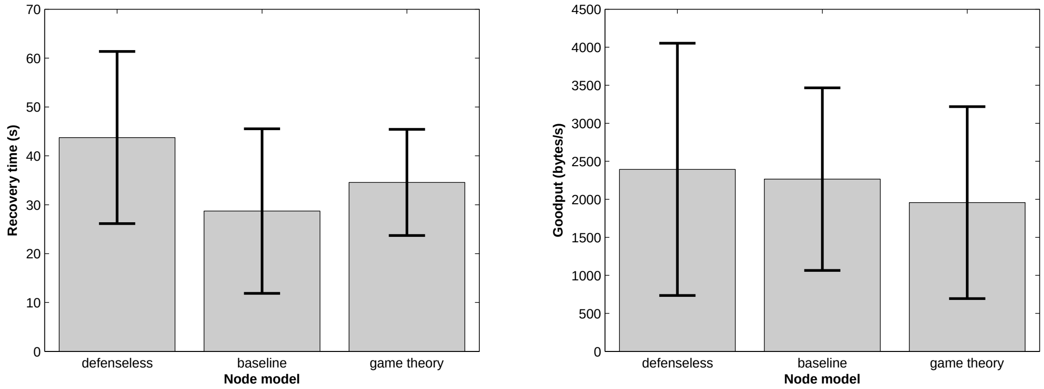


Figure 6.13: Recovery times and minimum goodputs achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in flooding scenario

trics. A number of reasons explain this behaviour. First, the game theoretic IDS is unable to use the additional discrimination to its advantage due to the massive increase of false alerts from the imperfect detectors that use eavesdropping to verify the behaviour of neighbouring nodes. Secondly, flooding results in the overflow of immediate neighbour's queues in which flooding packets from other adversaries may also be dropped, thus, the detection criterion may not always be met (if nodes exceed *RREQ_RATE_LIMIT*). Even if it is met, the response occurs only after IDS nodes have forwarded up to *RREQ_RATE_LIMIT* RREQs from the adversary onto neighbouring nodes. Each of these factors lead to poor preservation of desirable network metrics when flooding adversaries are introduced.

6.7.2 Unassisted Black hole

Eavesdropping was used to verify the forwarding behaviour of neighbours and determine the presence of consistent non-forwarders (described in §5.4.1). In this scenario, adversaries drop data they receive for destinations other than themselves, including data from other adversaries.

Packet Delivery Ratio

The results in Figure 6.14 show the network PDR and dropped ratio achieved in the unassisted black hole scenario for the different IDS' for an increasing ratio of adversaries in the network. As this ratio increases, it is more likely data will be routed via an adversary and dropped. In the worst case, 60% of adversaries drop an average 50% data in defenceless networks.

For small numbers of adversaries (up to 20%), the baseline and game theoretic IDS preserve a similar PDR to the omniscient network, but diverge at 30% adversaries or higher. From this point on, the baseline IDS performs better than the game theoretic IDS, preserving between 50% and 75% of the omniscient maximum for moderate numbers of adversaries. The omniscient network is unaffected over the full range of adversaries.

Average end-to-end (ETE) delay

Figure 6.15 illustrates a general decrease of ETE delay as the number of adversaries increase for each of the networks in the black hole scenario. For the omniscient network, this results from IDS nodes ignoring any routing transmissions from an increasing number of adversaries. For all other types of network, the decrease occurs partly from ignoring routing transmissions from adversaries, but also because more data is dropped from a successful attack, leading to less medium contention. Hence the defenceless network reports the lowest ETE delay, resulting from the increased number of dropped data packets from misbehaviour.

Control overheads

The control overheads for the unassisted black hole scenario are shown in Figure 6.16. The defenceless network offers constant overheads over the full ratio of adversaries. This is expected as the drop rate increases for higher ratios of adversaries, leading to a reduction in the routing overhead required to support the remaining data. The omniscient network exhibits similar overheads as the defenceless network but diverges after 40% adversaries due to the longer routes required avoid the increasing number of adversaries. The baseline and game theoretic networks show a similar overhead curve which increases at a greater rate as the numbers of adversaries increase. This reflects the increased likelihood that responses against numerous adversaries will be necessary to support each data flow.

Response Rate

The game theoretic IDS was found to be slightly less effective overall at reducing the number of dropped packets by black hole adversaries compared to the baseline (Figure 6.14). The advantage of employing the game theoretic IDS is shown in Figure 6.17 where the true response rate for moderate numbers of adversaries is 10–15% better than the baseline. This is as a result of the better discrimination between the misbehaving and benign nodes offered by the game theoretic reasoning. This advantage diminishes when there are equal numbers of adversaries and benign nodes.

Recovery Time and Minimum Goodput

Unlike the flooding attack, which continues to affect network performance for a period after the attack, the activation and deactivation of the black hole attack has an immediate

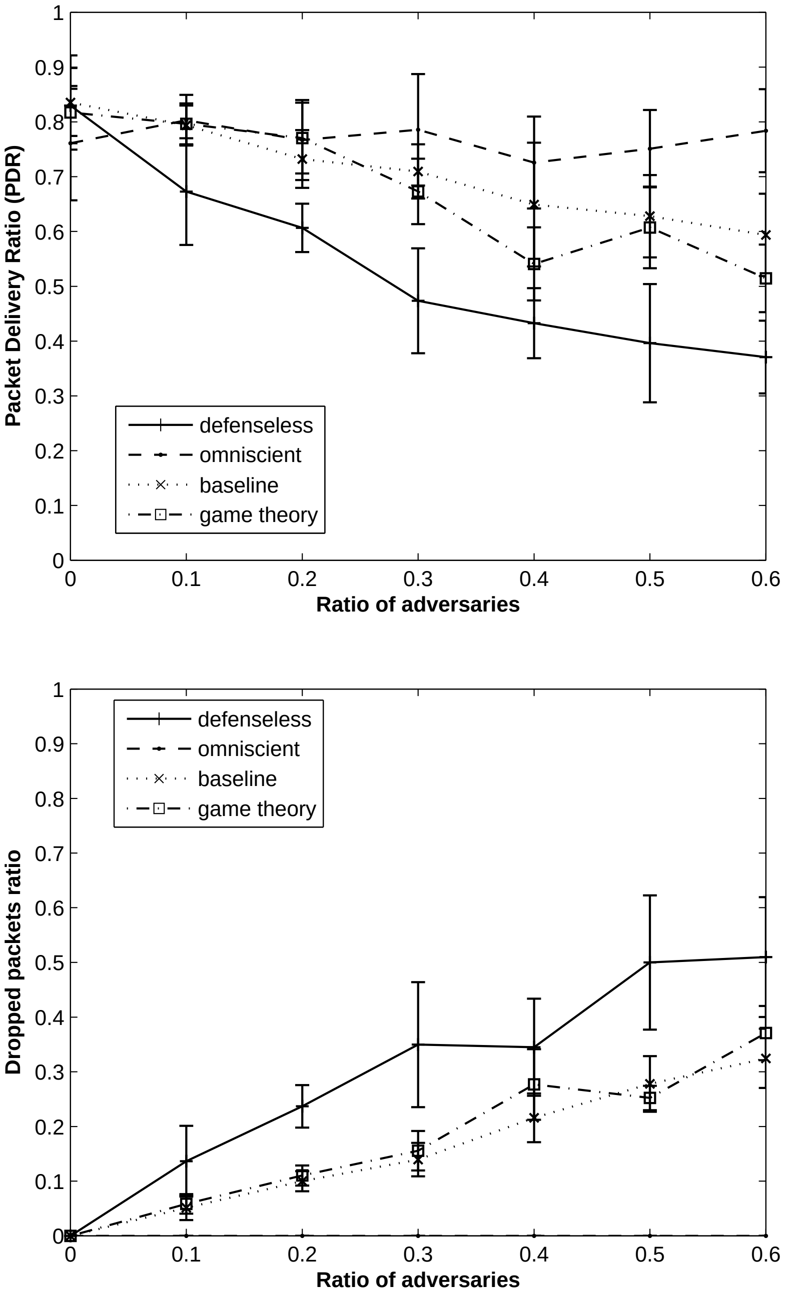


Figure 6.14: PDR and dropped packet ratio achieved in unassisted black hole scenario with an increasing number of adversaries

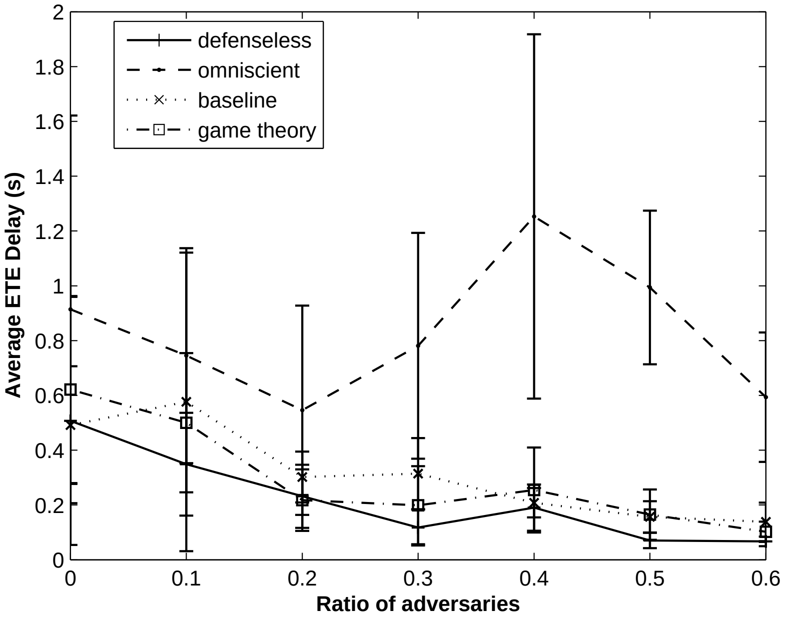


Figure 6.15: Average end-to-end (ETE) delay achieved in unassisted black hole scenario with an increasing number of adversaries

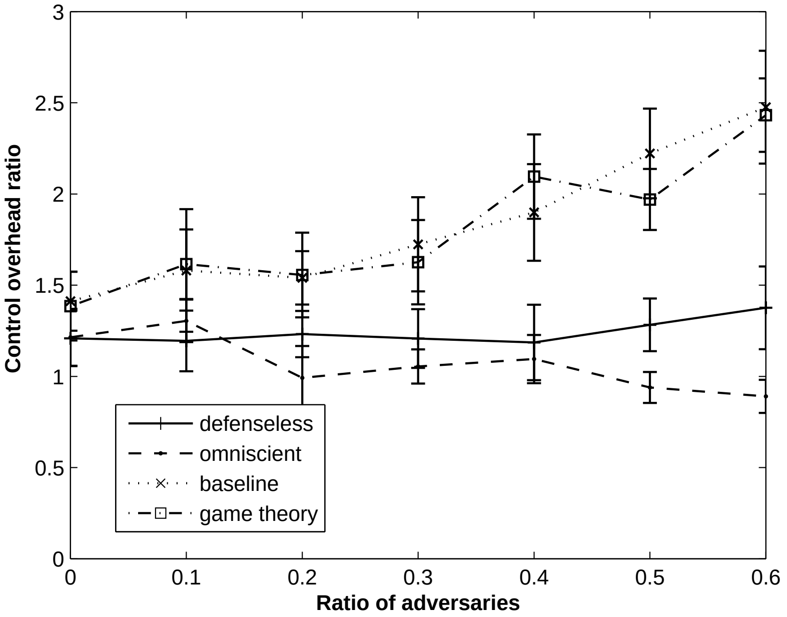


Figure 6.16: Control overhead in unassisted black hole scenario with an increasing number of adversaries

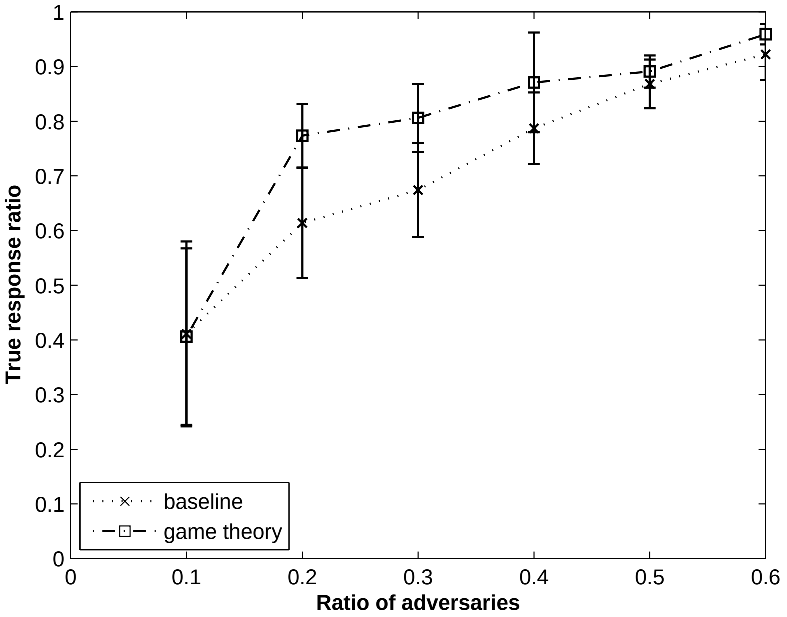


Figure 6.17: True response rate achieved in unassisted black hole scenario with an increasing number of adversaries

effect on network performance. This is shown in Figure 6.18 where the defenceless network takes on average 100s to recover to pre-intrusion levels of goodput - the exact time the misbehaviour is active for.

The advantages of running the game theoretic or baseline IDS is clearly illustrated by a doubling of the minimum goodput experienced during the attack compared to that of the defenceless network. In both cases, recovery time is reduced by more than half of the 100s attack duration. Furthermore, there is very little difference between recovery time and minimum goodput achieved by game theoretic or baseline IDS.

6.7.2.1 Discussion

This scenario illustrates the strength of the game theoretic IDS for moderate ratios of adversaries in the network as it better discriminates between benign and malicious behaviour, as shown by the higher true response rate with negligible difference in PDR compared to the baseline IDS. As the number of adversaries increases still further, the baseline IDS begins to lead in the overall PDR preserved and the difference in true response rate begins to converge with that of the game theoretic IDS. This convergence is to be expected as an indiscriminate approach will work better when the majority of nodes are malicious rather than benign, as there is a higher likelihood that a given response is taken against a malicious node.

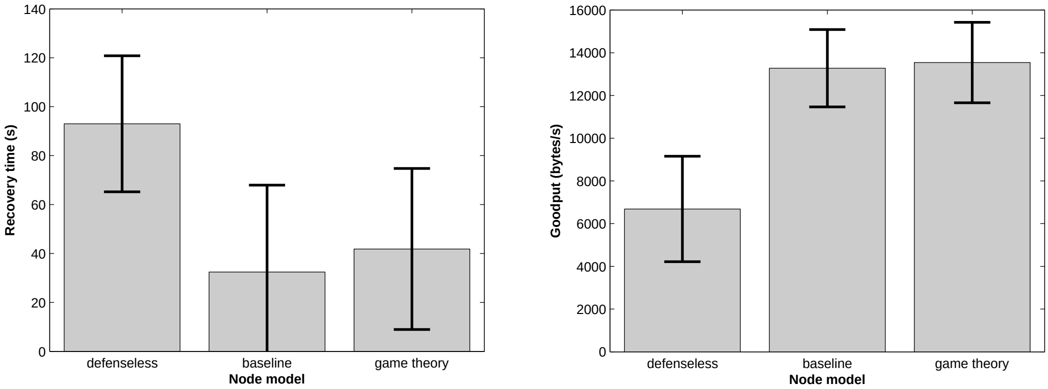


Figure 6.18: Recovery times and minimum goodputs achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in unassisted black hole scenario.

6.7.3 Black hole assisted with passive route invasion

An adversary can vastly increase the effectiveness of a black hole attack by falsely advertising itself as having the shortest route to a requested destination during route discovery. Several route invasion techniques were shown in §4.2.2.5. This attack is the product of two misuses: first the adversary advertises routes to any requested destination and then indiscriminately drops any data it receives that is not addressed to itself. Whilst both attacks can be detected, the detection of the black hole attack is imperfect, but fake routing information can be detected perfectly (§5.4.5.1).

Packet Delivery Ratio

The PDR for each network subjected to increasing ratios of adversaries in the network is shown in Figure 6.19. The presence of adversaries reduces the PDR by 50%, akin to §6.7.2, albeit with only 10% adversaries, at which point the attack saturates. This characteristic reflects the fact that adversaries are rate limited to prevent throughput starvation by the generation of large numbers of fake RREPs: extremely important for higher numbers of adversaries in which the attack would not scale. Omniscient nodes are largely unaffected by the attack as they ignoring any messages originating from known adversaries.

The maximum dropped packet ratio in the defenceless network is lower than Figure 6.14 shows. Since the overall PDR is reduced to similar levels, this may indicate that the additional traffic generated by the route invasion causes additional congestion that leads to higher MAC losses.

The baseline IDS performs the best of all the IDS schemes evaluated, tracking performance of the omniscient network at each ratio of adversaries in the network. The game theoretic IDS performs slightly worse and at its worst preserves approximately 15% less traffic than the baseline IDS.

Average end-to-end (ETE) delay

Figure 6.20 shows the ETE delay for the assisted black hole scenario. Immediately evident is the uniform increase in ETE delays for the defenceless network, caused by a number of factors. Firstly, the route invasion component of the attack generates a significant amount of additional routing traffic and further illustrates the necessity of rate limiting the route invasion behaviour of adversaries. The fake route replies generated by adversaries use an arbitrarily large sequence number that does not follow standard sequence number handling and leads to the emergence of routing loops and isolated large ETE measurements. Lastly, the number of HELLO packets increases accordingly in order for adversaries to persist the fake routes. The baseline, game theoretic and omniscient IDSs show relatively constant ETE delay across the full range of adversaries in the network.

Control overheads

Figure 6.21 shows that aside from the defenceless network, there is a general downward trend in control overheads as the ratio of adversaries increases. This decrease extends to the baseline and game theoretic IDS and includes the additional overheads from responding, showing it is advantageous detect and respond to this attack. The response from all IDS' block RREPs that correspond to a route invasion, which suppresses the additional routing traffic that emerges upon the success of this attack, including the increased number of HELLO packets from fake routes being established.

Response Rate

As Figure 6.22 shows, there is very little to separate the true response rate of the baseline and game theoretic IDS. This results from the attack being formed of two misuses; the route invasion is caught by a perfect detection scheme whilst the dropped packets are caught by an imperfect scheme. The route invasion on its own provides strong evidence of an impending attack and is caught before the target begins to send any data. This is the reason the threshold approach does as well as the more reasoned approach provided by game theory.

Recovery Time and Minimum Goodput

The recovery times and minimum goodputs for the defenceless, baseline and game theoretic networks are illustrated in Figure 6.23. The defenceless network recovers at approximately 100s, whilst the baseline IDS does so in approximately half the time, as it does when subjected to adversaries misbehaving with an unassisted black hole (§6.7.2). The game

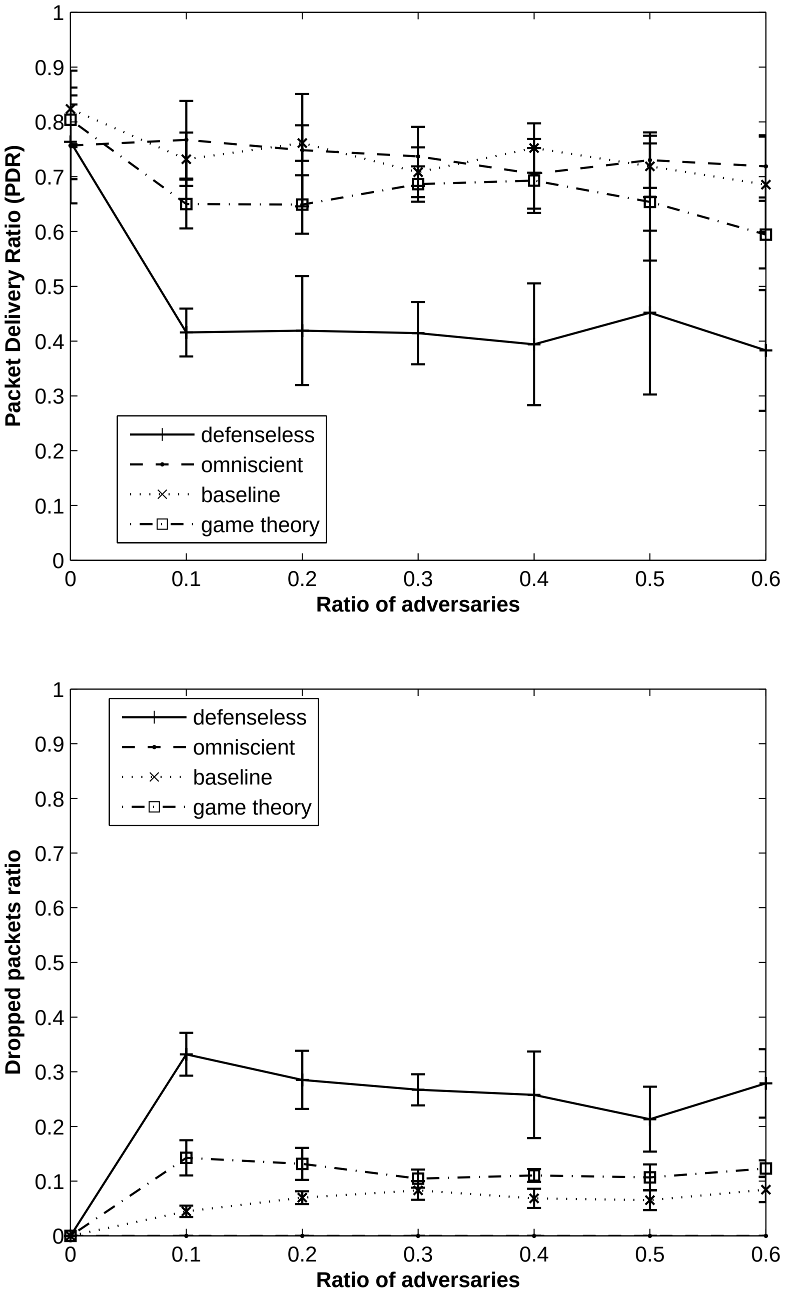


Figure 6.19: PDR and dropped packet ratio achieved in assisted black hole scenario with an increasing number of adversaries

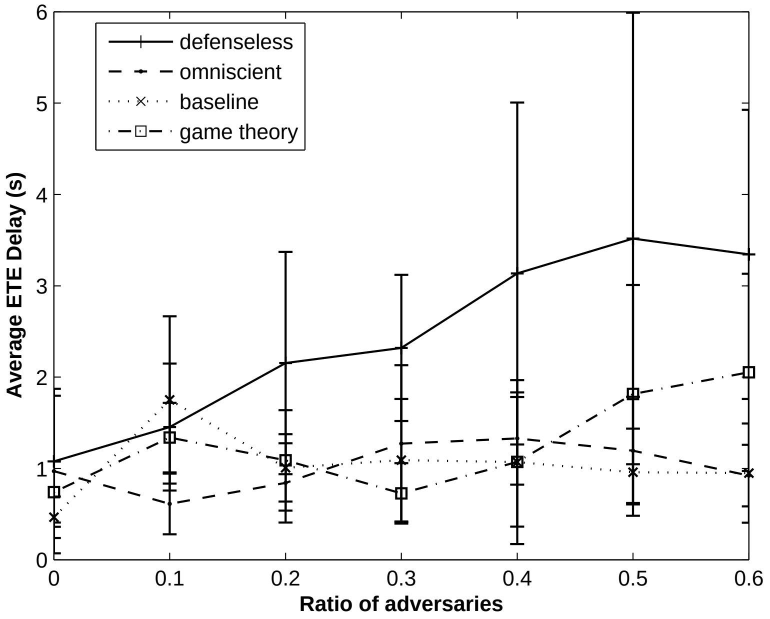


Figure 6.20: Average end-to-end (ETE) delay achieved in assisted black hole scenario with an increasing number of adversaries

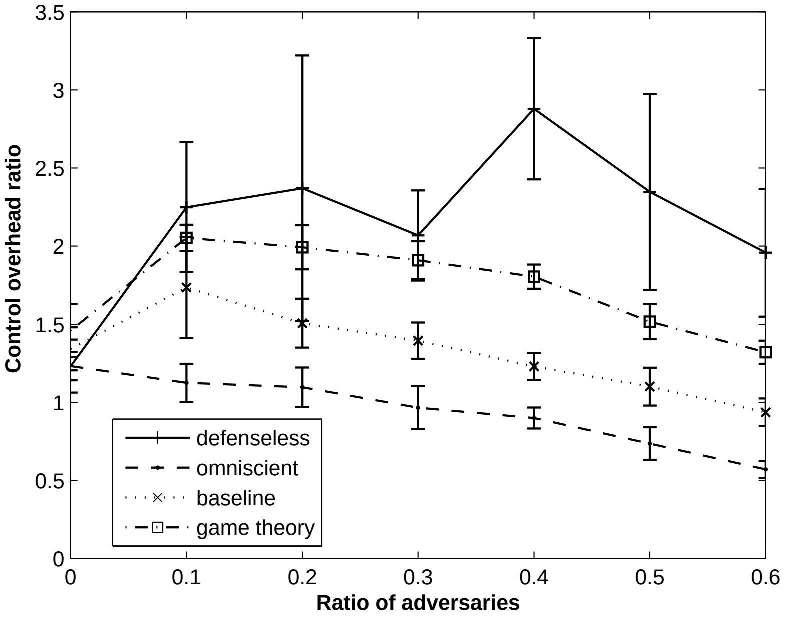


Figure 6.21: Control overhead in assisted black hole scenario with an increasing number of adversaries

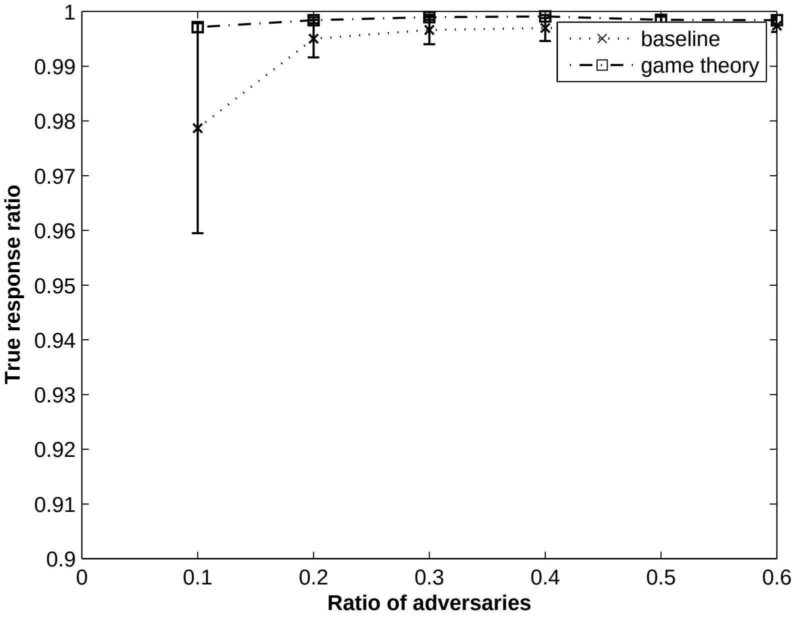


Figure 6.22: True response rate achieved in assisted black hole scenario with an increasing number of adversaries

theoretic IDS is unable to match its performance in this same scenario, averaging a 31.5% reduction in recovery time over the defenceless network. The baseline also doubles the minimum goodput experienced to preserve network performance, whilst the game theoretic protocol is within 15% of the goodput preserved by the baseline IDS. Compared to the unassisted black hole scenario, both IDS's show a slight reduction in the minimum goodput preserved, arising from the more detrimental characteristics of this type of attack.

6.7.3.1 Discussion

The baseline IDS exhibits much better performance than the game theoretic IDS in this scenario and tracks the omniscient protocol for PDR but via a purely local detection scheme that applies a threshold directly to m_i and m_p. Although it may appear that the game theoretic IDS performs worse with increased overheads, the metric is slightly misleading as it is dependant on the number of data packets transmitted, which is lower for the game theoretic IDS as it is unable to respond to adversaries as quickly as the baseline IDS. The "benefit of the doubt" factor used could be responsible for this larger convergence time, thus making it more insensitive to extreme misbehaviour, such as in this case. We can conclude that a threshold with short-term memory is a good measure against this kind of attack.

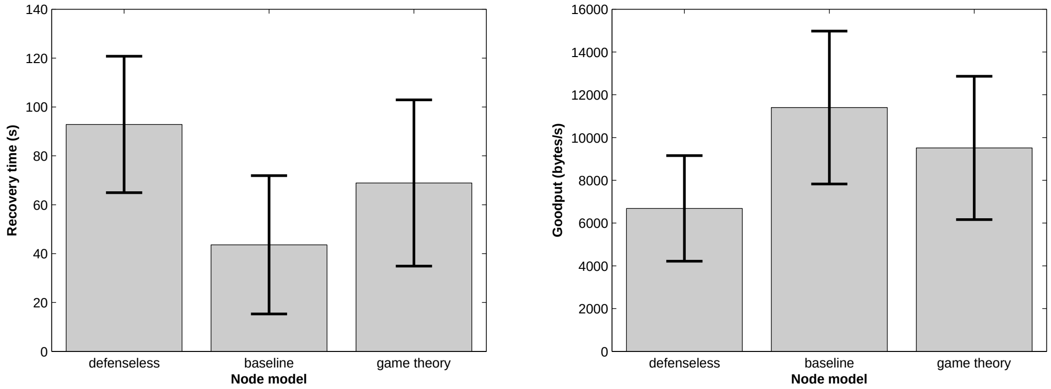


Figure 6.23: Recovery times and minimum goodputs achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in assisted black hole scenario

6.7.4 Wormhole

Wormholes exploit routing *race conditions* whereby the MANET routing protocol takes an action based on the first instance of a given message received and ignores instances of that message received later (Karlof & Wagner, 2003). In this case, only the first RREQ with a given RREQ ID is processed and forwarded by a neighbour: subsequent RREQs with the same ID are dropped (Perkins, 1997, §6.7). If colluding wormhole adversaries connect distant parts of the MANET over a lower latency link than the medium shared by the MANET, the adversaries will be chosen to serve most routes. Wormholes can be detected imperfectly by monitoring the forwarding behaviour of neighbours, also used previously to detect the presence of black hole nodes in the network (§6.7.2).

Packet Delivery Ratio

The resulting PDR for each network subjected to an increasing number of wormhole adversaries is shown in Figure 6.24. The average PDR for this scenario is approximately 80%, as illustrated by the omniscient network which is resilient to this attack, resisting the use of routes advertised by adversaries. With the introduction of a small number of wormhole nodes the average PDR of the remaining networks regularly exceeds 90% due to the use of the wormhole links, resulting in approximately 50% of all packets being served by at least one adversary. The attack peaks at 20% adversaries when 60% of all packets are served by a wormhole node and saturates after this point.

Although it appears from the PDR metric alone that both IDS's are largely ineffective to the effects of this attack, both reduce the ratio of intercepted packets by 49.1% at 10% adversaries to 20.8% at 30% adversaries when compared to the defenceless network. The data show that both IDS' become increasingly ineffective for larger ratios of adversaries in the network.

Average end-to-end (ETE) delay

Figure 6.25 shows that the introduction of a small number of wormhole adversaries results in a 90% decrease in the ETE delay of the defenceless, game theoretic and baseline networks due to the decreased latency offered by the wormhole links for intercepted traffic. The resilience of the baseline IDS to small ratios of adversaries is shown by the higher ETE delay which decreases with moderate numbers of adversaries.

Control overheads

The control overheads for the wormhole scenario are depicted in Figure 6.26, and show that a constant overhead for increasing ratios of adversaries in the defenceless and omniscient networks, whilst there is a linearly increasing overhead in the baseline and game theoretic networks. This additional overhead introduced by the baseline and game theoretic networks is due to the response initiated by the IDS's, which routes around adversaries as they are detected 'dropping' packets. The overheads introduced by the game theoretic protocol range from 10.3-17.2% less than the baseline one.

Response Rate

Figure 6.27 shows the true response ratio achieved by the game theoretic and baseline IDS and demonstrates that the further discrimination employed by the game theoretic IDS results in a 10% reduction in the number of false responses (for 10% adversaries) which is reduced to approximately 5% thereafter when compared to the baseline IDS.

Recovery Time and Minimum Intercepted Throughput

The recovery times and minimum intercepted throughput offered by each of the networks is shown in Figure 6.28. In general, the baseline and game theoretic IDS's offer an appreciable reduction in the recovery time, whilst reducing the number of bytes intercepted by adversaries. The larger spread in the intercepted throughput reported by each of the IDS' results from the local response, which may succeed in re-routing data to a benign node, or instead choose another wormhole node.

6.7.4.1 Discussion

Each of the measures suggest the game theoretic IDS provides better performance than the baseline IDS in this scenario. The game theoretic IDS offers a similar reduction in the intercepted throughput whilst providing a lower number of false responses and maintaining a lower overall overhead compared to the baseline IDS. The performance of

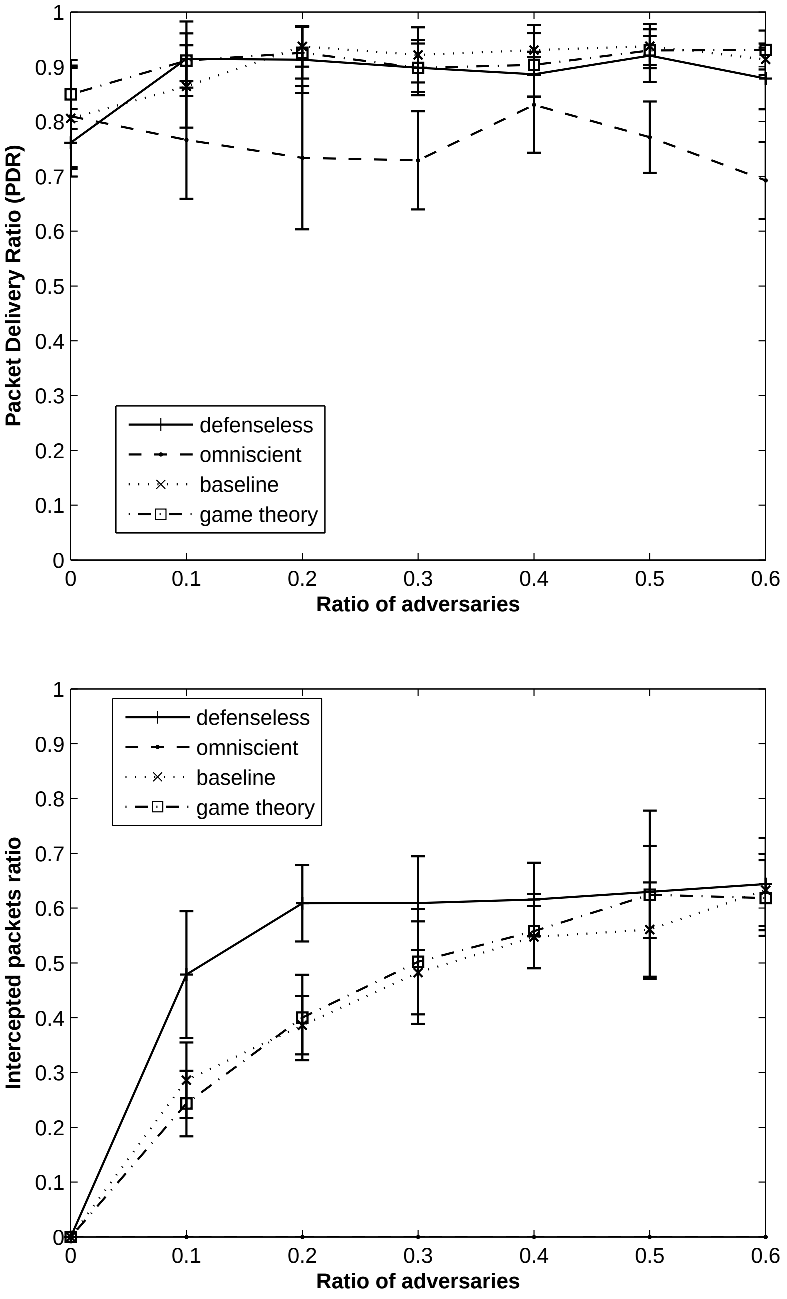


Figure 6.24: PDR and intercepted packet ratio achieved in wormhole scenario with an increasing number of adversaries

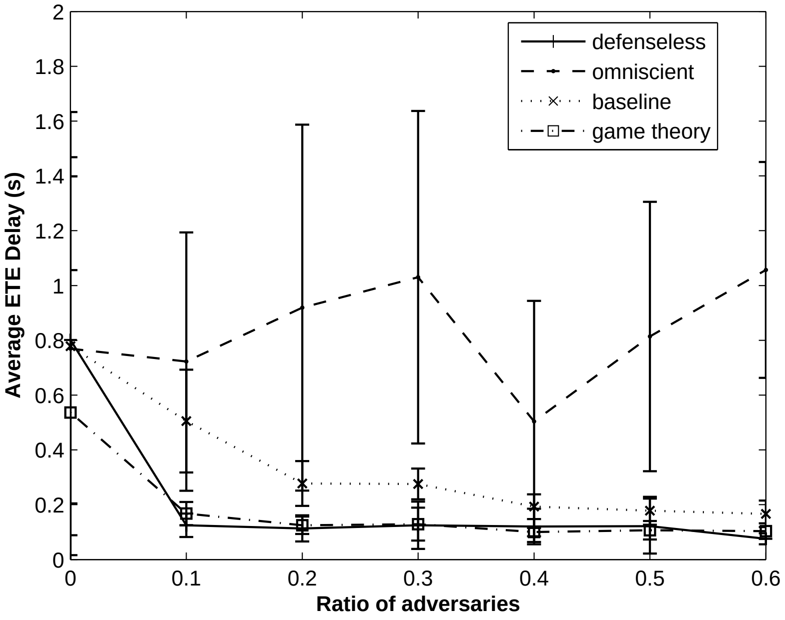


Figure 6.25: Average end-to-end (ETE) delay achieved in wormhole scenario with an increasing number of adversaries

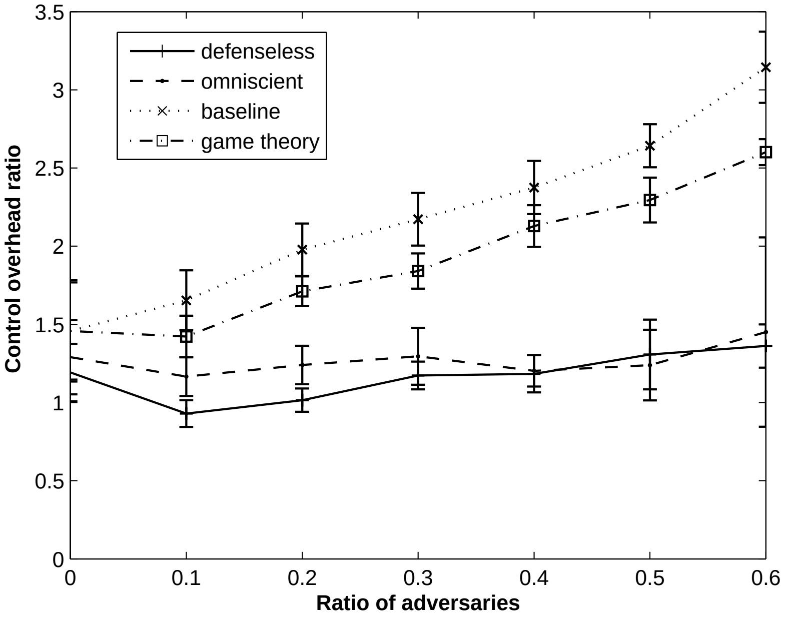


Figure 6.26: Control overhead in wormhole scenario with an increasing number of adversaries

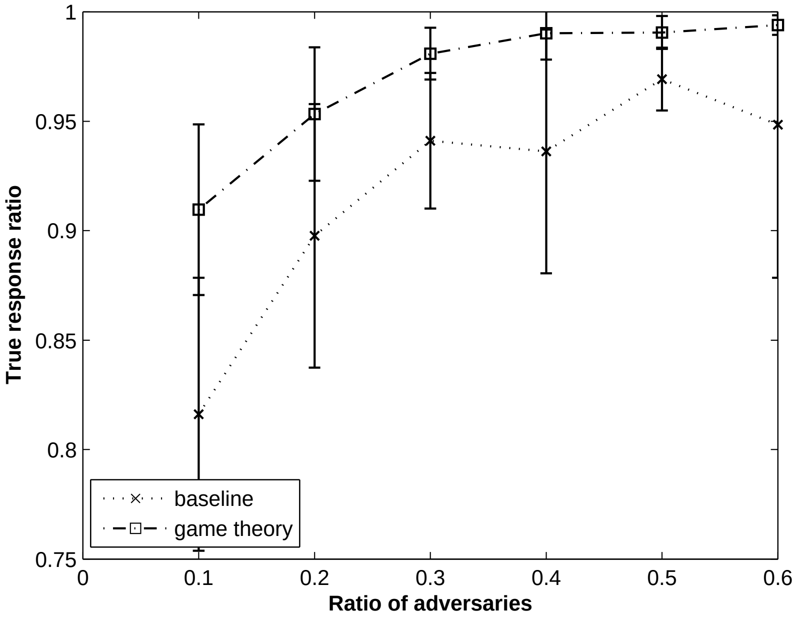


Figure 6.27: True response rate achieved in wormhole scenario with an increasing number of adversaries

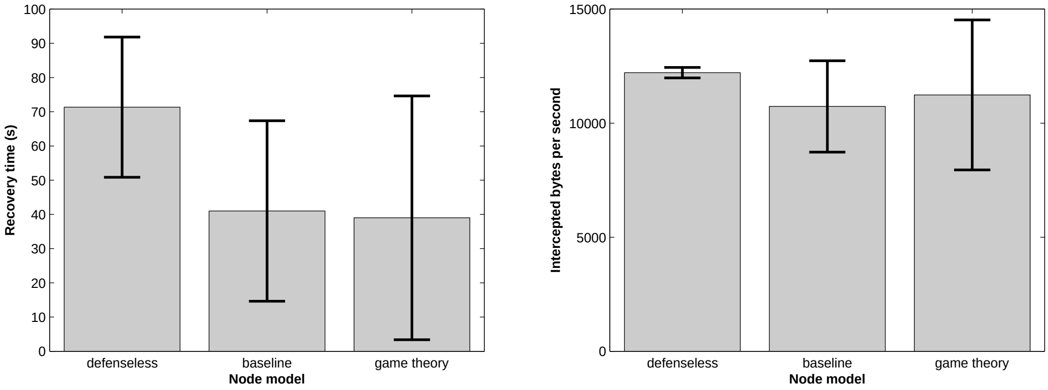


Figure 6.28: Recovery times and minimum intercepted bytes achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in wormhole scenario

Misbehaviour	Frequency of occurrence	Duration
Data modification	0.225	20s
Unassisted black hole	0.225	20s
Black hole and passive route invasion	0.225	20s
Wormhole	0.225	20s
RREQ flooding	0.1	1s

Table 6.20: Set of misbehaviours, their relative weightings and duration

both suffers when there is a moderate numbers of adversaries and converges with the defenceless network's performance for high numbers of adversaries.

6.7.5 Random misbehaviour

In the final scenario, a one-time random distribution of misbehaviour was generated over the entire simulation time from the set of misbehaviours and their respective frequency of occurrence as shown in Table 6.20. Of these, flooding has a lower weighting due to the previous trace file explosion problem described in §6.7.1. The output from `RandomMisbehaviourGenerator` was applied to each adversary at simulation initialisation. The scenario evaluated the context switching ability of each of the IDS schemes.

Packet Delivery Ratio

Figure 6.29 shows the PDR preserved over each network type for an increasing ratio of adversaries in the network. The defenceless network shows a 45.3% reduction in PDR from the network average at 0% adversaries to the lowest PDR at 60% adversaries. The rate at which the attack reduces the PDR diminishes for larger numbers of adversaries. The omniscient network shows a slight reduction in the PDR, from the flooding component of the random scenario.

For low ratios of adversaries, the baseline IDS preserves 95.6% of the omniscient PDR, which decreases to between 81.9% and 85.8% when larger numbers of adversaries are present. The game theoretic IDS performs similarly to the baseline for 10% adversaries, but thereafter offers little protection.

Average end-to-end (ETE) delay

ETE delay for the random scenario across all network types is shown in Figure 6.30. The defenceless network shows a variable delay as the ratio of adversaries in the network increases, due to the extra contention induced by the flooding and assisted black hole

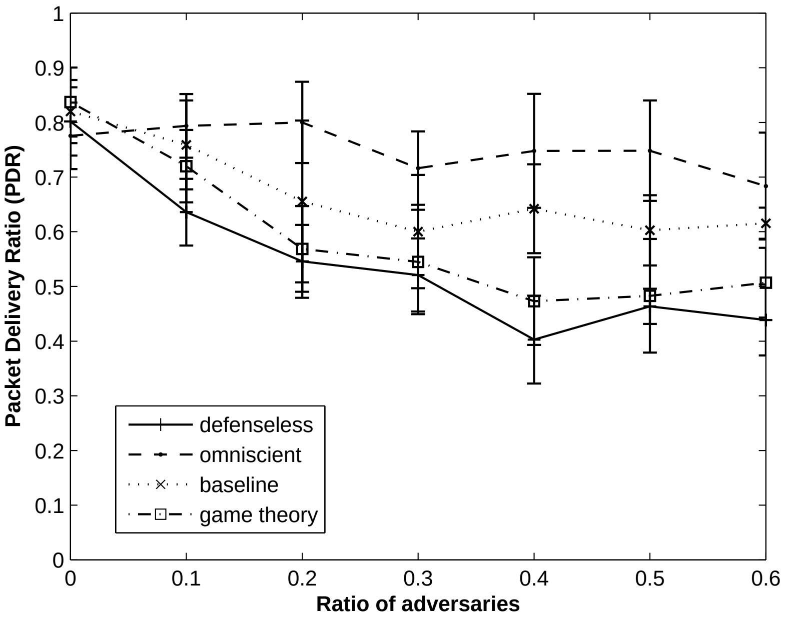


Figure 6.29: PDR achieved in random misbehaviour scenario with an increasing number of adversaries

components of the attack. The omniscient network exhibits an ETE delay response similar to the flooding scenario (§6.7.1) due to the local congestion hot spots from flooding adversaries. The delay from the baseline network increases at a slower rate than the game theoretic protected network, reaching their maximum ETE delay at 20% adversaries and remaining at an average of 2.01s and 2.46s for the remaining ratios of adversaries respectively. The improved response provided by the baseline IDS corresponds to a 33% reduction in ETE delay when compared to the game theoretic network.

Control overheads

The omniscient protocol's control overheads remain at the network average, as illustrated in Figure 6.31. The overheads for the defenceless protocol rise as a result of the fake RREP and flooding causing additional contention, but this increase is offset by the wormhole component, and the dropped data resulting in less latency for the remaining data.

Of the IDS's, the baseline leads to better preservation of network quality whilst minimising the number of extra overheads; the average overhead induced at any time is a maximum of 36.5% more than the defenceless network. The game theoretic IDS performs much more poorly, resulting in a maximum 75% overhead over the defenceless network.

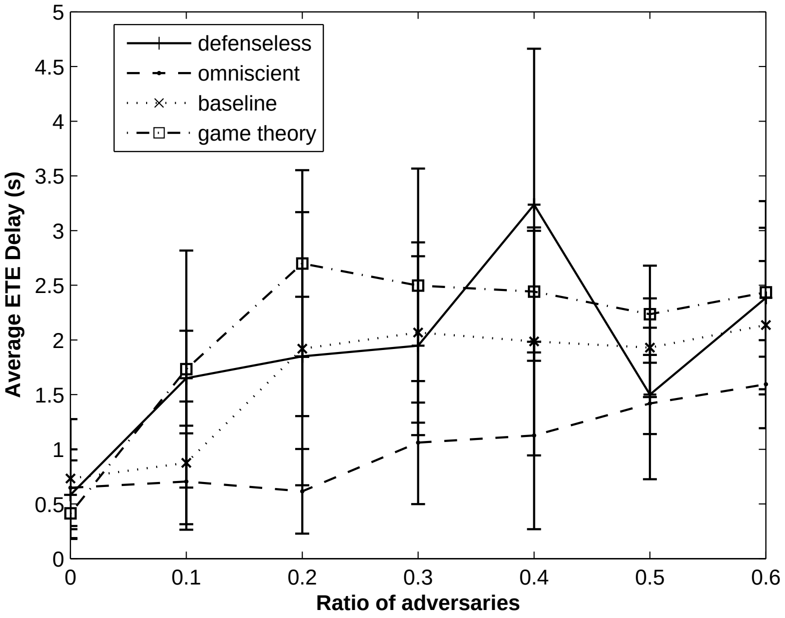


Figure 6.30: Average end-to-end (ETE) delay achieved in random misbehaviour scenario with an increasing number of adversaries

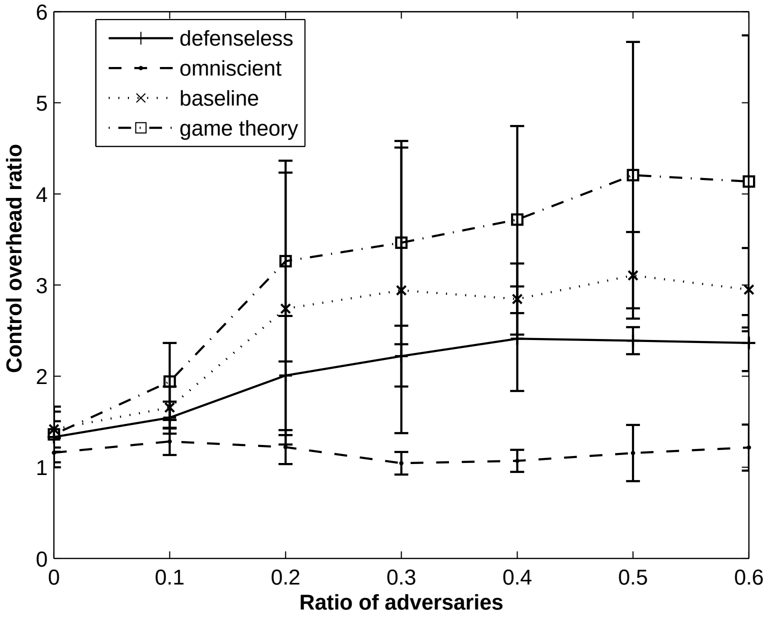


Figure 6.31: Control overhead in random scenario with an increasing number of adversaries

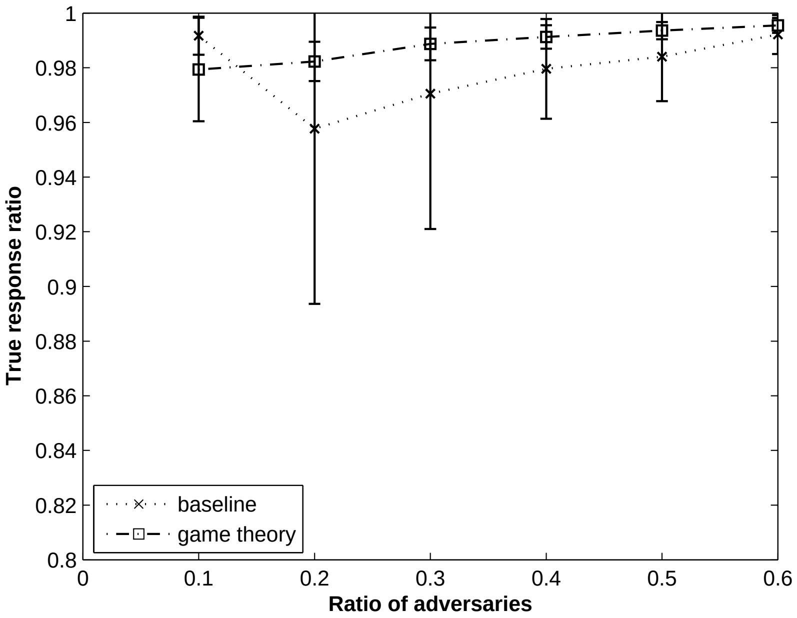


Figure 6.32: True response rate in random scenario with an increasing number of adversaries

Response Rate

The true response rate for the baseline and game theoretic IDS is shown in Figure 6.32, indicating that whilst the baseline protocol incurs more false responses than the game theoretic protocol and has greater variability for a lower proportions of adversaries, it converges towards the performance of the game theoretic IDS for higher ratios.

6.7.5.1 Discussion

The metrics suggest that the baseline IDS performs better in this scenario as it preserves a larger portion of network PDR and simultaneously minimises the overhead required for the extra responses. Although the true response ratio for the baseline IDS is slightly lower than the equivalent for the game theoretic IDS for lower numbers of adversaries, its performance converges with that of the game theoretic protocol for moderate to high numbers of adversaries. The results show that the baseline IDS is as effective with moderate numbers of adversaries as it is when there is a majority of adversaries, illustrated by the difference in PDR between itself and the resulting PDR from the omniscient network.

6.8 Summary

This chapter has shown the application of a threshold and game theoretic inspired MANET IDS to several scenarios employing both local and distributed misbehaviour of varying subtlety in order to determine the effectiveness and the scalability of both approaches with increasing number of adversaries. The threshold IDS was used as a baseline in order to determine the efficacy of the game theoretic approach.

The core of the game theoretic implementation was the design of a game with appropriate regions that responds or processes packets from neighbours based on their observed behaviour. The output of the game is a mixed strategy solution that determines this outcome and applies it probabilistically until it is updated in the next interval. The baseline and game theoretic IDS employ the same method of computing a node's behaviour profile in two measures, m_p and m_i, which were comprised of independent metrics that measured the ability of a node to perform a given function, such as forwarding data, from local observations. What differentiates the approaches is how the measures are evaluated. The threshold IDS applied a strict threshold on m_p and m_i whilst the game theoretic approach employs further reasoning (termed a "benefit of doubt"), by placing less importance on m_i (constituting weak evidence) without an associated reduction of m_p (stronger evidence), to reduce the number of retaliations against benign nodes in imperfect wireless environments, such as fading ones. In these environments, taking the output of detectors that operate via eavesdropping is unreliable and leads to false positives as often the notification that a neighbour has fulfilled a given function (e.g. packet forwarding), is missed.

Overall, the game theoretic protocol demonstrates a significant improvement in true response rate compared to the baseline IDS for most of the scenarios presented in this Chapter, though this does not always translate into an improvement in network metrics. For example, the baseline provides consistently better network performance in the assisted black hole scenario and when subjected to random misbehaviour. This may be the result of the longer convergence time from the additional reasoning employed in the game theoretic IDS when it is either unnecessary (as in the assisted black hole scenario) or when it prevents quick context switching (as in the random misbehaviour scenario). Both approaches are unsatisfactory when subjected to flooding and for high numbers of wormhole adversaries, where they failed to preserve any network performance over the defenceless network. Flooding is a special case where the core detection strategies are compromised by the additional contention, resulting in self-reactivity that is illustrated by large false positive rates in both the game theoretic and baseline IDS.

Both the game theoretic and baseline IDS employ local detection and response, and do not share their intrusion knowledge with their neighbours. This is advantageous in that the IDS is less exposed to misleading from false reports by adversaries. Negative false reports could cause IDS nodes to deny a benign node access to the network, whilst positive false reports disseminated by colluding adversaries could reduce the likelihood of a retaliation, even if that adversary is actively attacking the network. This approach also means, however, that if nodes do not have prior experience of an adversary they will be susceptible to its misbehaviour they detect it themselves, even if this adversary is known to others. The leads to a slower network-wide response, and imposes an upper limit on its effectiveness. In addition, these techniques respond to misbehaviour over a single timescale (refreshed each *interval* seconds), whilst a near-instantaneous response would be appropriate for misbehaviour that can be detected perfectly.

CHAPTER 7

SIDS+AIS: AN IMMUNE INSPIRED APPROACH TO DETECTION AND RESPONSE

7.1 Introduction

Chapter 6 showed that incorporating game theoretic reasoning encouraged IDS nodes to give neighbours that triggered imperfect detectors with little corresponding evidence from perfect detectors a benefit of the doubt. The contribution of *sIDS* is a reduction in the rate by which benign nodes are retaliated against and the application of more appropriate responses.

The objective of the Human Immune System (HIS) is to protect the host by employing distributed mechanisms that work in tandem to orchestrate an immune response. This Chapter provides an overview of these mechanisms that served as an immune-inspired component to *sIDS*, referred to as *sIDS+AIS*. The scheme was run in parallel to the game theoretic component over each of the intrusion scenarios detailed in §6.6.3, and end to end network and intrusion detection metrics were extracted to enable the effectiveness of the approach to be evaluated.

7.2 From the Human Immune System to MANET IDS

The properties of the immune system and challenges it faces were discussed in previously in §2.4. These challenges are not too dissimilar of the security challenges faced in MANETs. A number of the features of the immune system emerged as being desirable within a MANET IDS:

- **multiple layers**: the innate response deals with *a priori* known threats that can be detected perfectly. The adaptive response deals with more subtle threats or those where the pathogen has subverted the innate response.
- **over different timescales**: the innate response is instantaneous whilst the adaptive response occurs much later and acts over a much longer time period.

- **agent signalling**: cells of the immune system use chemical signals to orchestrate a response.

In Chapter 6 we showed that fortifying a MANET with *sIDS* that detects and responds locally results in an increase in end-to-end network performance over a defenceless network when both are subjected to malicious behaviour. Even so, there is a limit to the effectiveness of this purely local approach, shown by its ineffectiveness to deal with more subtle types of misbehaviour, or those where a co-ordinated response is required to prevent the spread of misbehaviour. This was illustrated in §6.6.3 where the best performing IDS is still far from the performance of the ideal omniscient network that has a shared global knowledge of adversaries.

The local detection paradigm employed so far means that if a node identifies a suspect as misbehaving, even by perfect schemes, this knowledge is not shared with its neighbours. In addition, detection activities operate over a single timescale, with no discrimination made between perfectly or imperfectly detected misbehaviour. Thus, when the IDS retaliates to possible misbehaviour from an imperfect source, it often results in a false positive. The following sub-sections describe the above properties in the context of a MANET IDS.

7.2.1 From innate TLRs to perfect detectors

On breaching the epithelial surfaces, a major part of the innate response involves the recognition of PAMPs by TLRs present on the surface of immune cells. As stated earlier, PAMPs exist only on pathogens, thus are a definitive marker of *non-self.* The activated immune cell reacts near instantaneously, by digesting the pathogen (a local response) and/or releasing cytokines to recruit neighbouring immune cells that are in transit (an extended response).

Applying this to the MANET environment, *sIDS+AIS* offers a near-instantaneous local response for misuses that can be classified **perfectly** (with a zero false positive rate) to short-circuit the long convergence time of the current IDS architecture (Figure 6.4). Secondly, *sIDS+AIS* disseminates this knowledge to its neighbours to co-ordinate an extended response should other neighbours corroborate with these findings. If the majority of nodes corroborate, distant neighbours with no first-hand experience of the misbehaving node can also contribute to the extended response against it.

7.2.2 Adaptive response and imperfect detectors

The adaptive response takes over a few days after the original infection (Janeway *et al.*, 2004, p. 21) if the innate response has failed to rid the pathogen from the host. The cells involved are tasked with classifying more subtle markers of infection, such as recognising the altered expression of Major Histocompatibility Complex (MHC) on infected host cells

(Janeway *et al.*, 2004, p. 28). Cells of the adaptive response often require activation by multiple co-stimulatory signals before being able to carry out their effector function.

In *sIDS+AIS* signals are used to reduce the uncertainty of the imperfect detection schemes. In particular, a node's observation radius is limited by unidirectional links and contention, such that two nodes that are both neighbours of the suspect may have a slightly different vantage point and could classify the suspect differently. When consistent imperfect misbehaviour is observed with no corresponding evidence from perfect detectors, IDS nodes cross-correlate alerts generated from the imperfect schemes, to reduce the likelihood of falsely responding. Like the above, should the majority of reports cross-correlate, neighbours with no first-hand experience of the misbehaving node can also contribute to an extended response.

7.3 An implementation of an AIS component for sIDS.

The implementation of the AIS component was inspired by the three desirable properties of the immune system discussed earlier: multiple layers, response over multiple timescales, and the organisation of extended responses, classified as innate (§7.3.2) and adaptive (§7.3.3), via immune cell signalling. The shared services used by both are described first.

7.3.1 Reporting and aggregating intrusion data

Majority voting is a simple mechanism to determine intrusions with or without strong evidence in a distributed manner. The technique described by Zhang & Lee (2000) was applied to compute the extended response in both the innate and adaptive components. Rather than trust and act on reports from individual neighbours implicitly, majority voting aggregates a number of them, reducing the effect of falsified or erroneous data from neighbours. It is robust to a minority of compromised nodes sending falsified data.

7.3.1.1 Disseminating reports

The dissemination of intrusion reports is central to the AIS' extended response via the innate or the adaptive component. To disseminate these reports, gossiping is used, in which messages are forwarded with a given probability to reduce the redundancy of flooding (Li *et al.*, 2002; Ni *et al.*, 1999; Haas *et al.*, 2006). In flooding, a node transmits a message to all its neighbours, who in turn transmit the message to their neighbours until the message reaches the desired set of nodes. It is routinely used by ad-hoc routing protocols during route discovery and is the source of large overheads that reduce the capacity of MANETs (Yi *et al.*, 2003). Haas *et al.* (2006) applied gossiping to AODV route discovery, showing its ability to reduce routing load and improve network metrics whilst maintaining high delivery ratios of control messages.

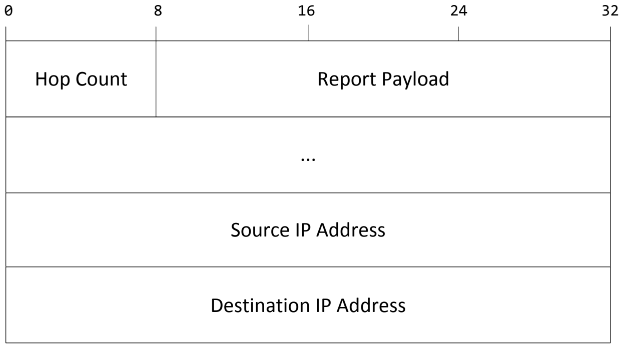


Figure 7.1: General structure of report IP packet

The approach we use is most similar to GOSSIP1(p, k) in Haas *et al.* (2006). A node originating a report about a given suspect node sends it with $p = 1$, and waits for $REPORT\_TIMEOUT$ seconds to acquire reports from other nodes about the suspect (as shown in Algorithm 17). Neighbours increment the hop count field (h) and compute p as follows:

$$p = \begin{cases} 1 & h \leq k \\ \frac{1}{h-k} & h > k \end{cases} \tag{7.1}$$

The value $k = 3$ was determined empirically by running several simulations and was a good trade-off between the radius of report dissemination and control overhead. In this scheme, neighbours forward a given report once at most, by storing recently received reports in a buffer; the reporting node's identity ('Source IP Address') and the report contents are stored in a data structure keyed by the target node ('Destination IP Address'). The structure of the report payload is illustrated in Figure 7.1. It defines the *target* (the 'Destination IP Address'), the *reporting node* (the 'Source IP Address'), the report *payload* and the *hop count* ('Hop Count'), which is incremented by intermediate nodes during its dissemination.

When a neighbour receives a new report, it also originates its own report about the reported suspect node if it has any first-hand experience about it. If the same report is received again, it is silently discarded. Broadcast based gossip is employed to ensure the functioning of the extended response independently from the routing protocol (which may be under attack). The reporting procedure is throttled such that each neighbour that participates in the reporting procedure for a suspect cannot initiate another for that node until at least $MIN\_REPORT\_INTERVAL$.

7.3.1.2 Data aggregation

Once *REPORT_TIMEOUT* has elapsed, the n received reports about the suspect are processed to determine a consensus. Whilst the conditions that determines an intrusion vary between the innate and adaptive component, if the computation concludes that the suspect has intruded, the node responds.

7.3.1.3 Managing active responses

For simplicity, the same data structure is used by the innate and adaptive component when initiating and managing the state of responses. Algorithm 15 illustrates how this data structure is updated when either component computes that the node with identity *macIpPair* has intruded, and shows for each detected misuse the length of the response increases exponentially.

Algorithm 15 `respond(`*macIpPair*`)`

```
if actively responding to $macIpPair$ then
    entry ← bufResponse[$macIpPair$]
    $entry.bo$ ← 2*$buf.bo$
else
    create new $entry$
end if
```

To achieve this, backoff values for monitored nodes are cached in a finite buffer with the oldest entries from out-of-range neighbours removed first. Then, the length of response for *macIpPair* is computed from *bo* as follows:

$$response\_time = bo * RESPONSE\_BASE \tag{7.2}$$

RESPONSE_BASE is a parameter that determines the smallest length of time a response can be active.

7.3.2 The innate component

The innate component of the AIS consists of a set of artificial TLRs, which comprise the **perfect** detectors shown in §6.2. This is accomplished by feeding the notifications directly into the innate component, as shown in Figure 7.2 (extending the architecture of the IDS shown in Figure 6.4).

7.3.2.1 Local response

Should one or more artificial TLRs be activated by a suspect's audit data, the IDS responds locally and near-instantaneously to the detected threat by blocking further communication

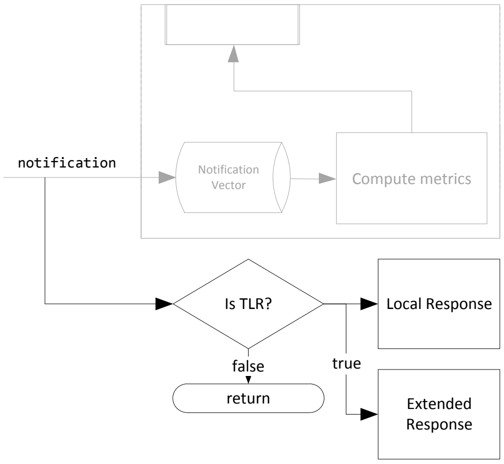


Figure 7.2: Flow diagram of innate response when notification is generated by one or more detectors. If the notification is from a perfect detector the innate component responds locally and then initiates an extended response.

between itself and the suspect (§7.3.1.3). Any messages that are received by the suspect during this active response are delivered only to the IDS component in order to monitor its future behaviour.

7.3.2.2 Extended response

After the application of the local response, an extended response is co-ordinated by sharing knowledge of the intrusion to neighbouring nodes by invoking a reporting procedure. The initiator of the extended response constructs a report, adds its observation to it and transmits it onto its neighbours. When a neighbour receives a new report, it forwards the original report to its neighbours and, if it has recent first-hand experience, disseminates its own report about the target. The strategy for disseminating reports was described in §7.3.1, and aside from the payload and majority voting criteria, is common to both the adaptive and innate components of the AIS. An example of the reporting procedure and corresponding extended response is shown in Figure 7.3.

The innate report IP packet is illustrated in Figure 7.4, and has a payload that consists of a flag (T) if one or more TLRs was activated by the target node specified in the Destination IP Address field.

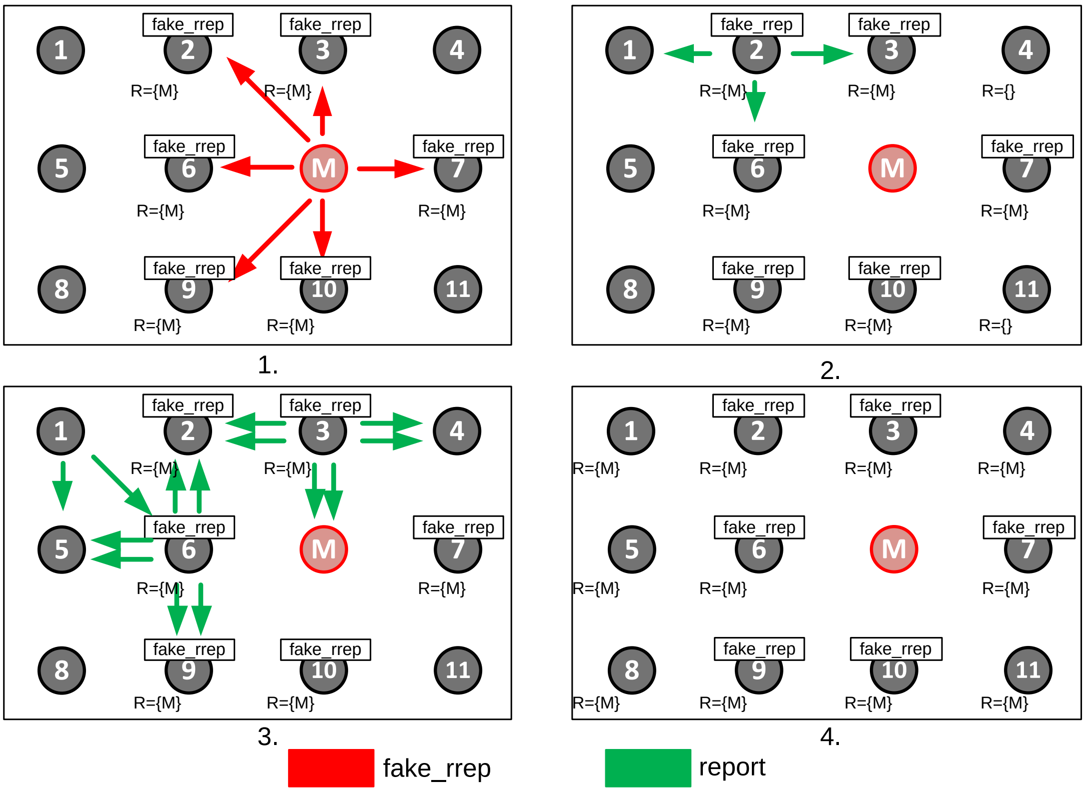


Figure 7.3: Report procedure for innate component of AIS. 1. An adversary misbehaves by sending fake route replies, which is detected by most, but not all of its immediate neighbours. 2. After a random jitter period, one of the immediate neighbours, node 2, initiates the innate reporting procedure by constructing and sending a report containing the new intrusion state to its immediate neighbours. 3. Neighbours forward reports and if the receiver is a neighbour of the adversary it sends its own report out (indicated by the double arrows). This process continues. 4. After a timeout, nodes determine a consensus, in this case, isolating M.

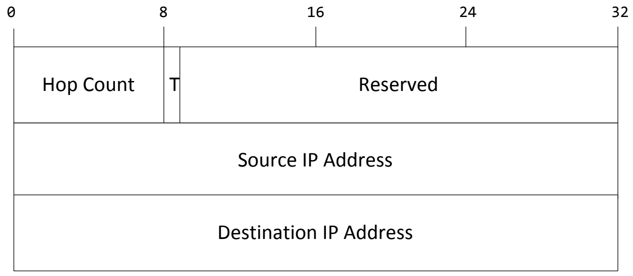


Figure 7.4: Structure of innate report

Once the reporting procedure times out, the reports for the suspect are aggregated if at least *MIN_REPORTS* reports have been received. A majority vote is computed if the ratio of reports which have the TLR flag (T) set is greater to those that are unset. If more nodes report that at least one TLR was fired for the suspect in the near past, then this node invokes the local response described earlier.

Pseudo code for initiating the local and extended innate responses is shown in Algorithm 16. The sending of an innate report and the management of the innate report buffer is shown in Algorithm 17 whilst the approach for determining the consensus report timeout is listed in Algorithm 18.

Algorithm 16 `IdsInterface.alert(`*type*`)` method for generating innate intrusion reports

```
if tlrFired(type,macIpPair) then
  if macIpPair not in report buffer then
    initiateInnateReportProcedure(macIpPair)
  end if
  armResponse(macIpPair)
else
  return
end if
```

7.3.3 The adaptive component

The adaptive component of the AIS is used to cross-correlate a node's own experience about a suspected intrusion with neighbours that may also have first-hand experience to report, however, it is weak evidence characterised by a low value of m_i in the absence of m_p. The intuition is that the cross-correlated outcome should more accurately reflect the

Algorithm 17 `initiateInnateReportProcedure(`*macIpPair*`)`

```
$netAddr \leftarrow$ node ip address
if $macIpPair$ not in $innateReportBuffer$ then
    $entry \leftarrow$ new innate report buffer entry
    $msg \leftarrow$ new innate report
    $msg.tlrFired =$ true
    $msg.src = netAddr$
    $msg.destIp = macIpPair.ip$
    send($msg$)
    $entry$.addReport($msg.tlrFired, msg.src$)
    queue innateReportTimeout($entry$)
end if
```

Algorithm 18 `innateReportTimeout(`*entry*`)`

```
if $|entry.reports| > MIN\_REPORTS$ then
    compute majority($entry.reports$)
    if intrusion detected then
        armResponse($macIpPair$);
    end if
end if
```

behaviour of the suspect than just that from local observations.

An adaptive report procedure is initiated when m_i and m_p are re-computed by the IDS for each known neighbour (§6.2) each *interval*. If any node is assigned a low value of m_i from the computation of the individual metrics, the adaptive component can initiate a reporting procedure for that node if it hasn't done so for *MIN_REPORT_INTERVAL*, as illustrated in Figure 7.5.

The reporting procedure uses gossip to push reports to neighbours and majority voting to compute a consensus similar to the innate component. The adaptive report IP packet is illustrated in Figure 7.7. The payload contains the most recent computed value of m_i for the target node (specified in the Destination IP Address field). Nodes only reply if they have had recent first-hand experience of the node. This prevents stale values of m_i being disseminated and contributing to the consensus.

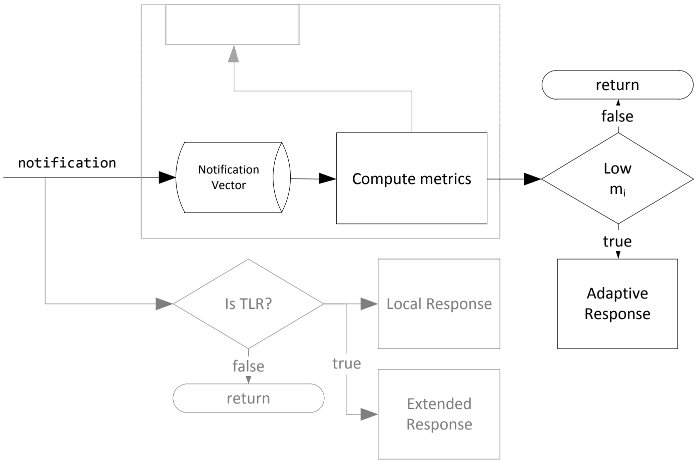


Figure 7.5: Flow diagram of adaptive response, invoked when any suspect has a low m_i.

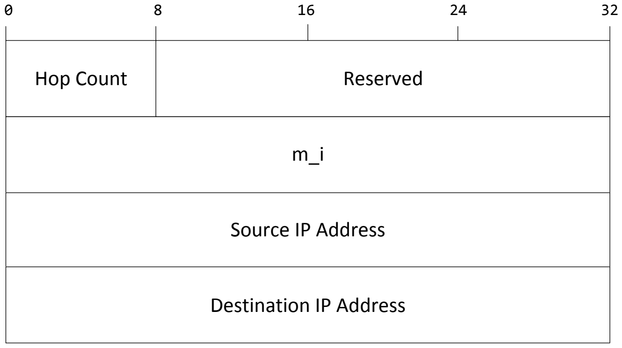


Figure 7.7: Structure of adaptive report

Once the procedure times out, the consensus is computed by applying an *IMPERFECT_THRESHOLD* to each of the samples of m_i reported by neighbours about the suspect. Neighbours compute a consensus, which if it indicates an intrusion, invoke a response, as detailed in §7.3.1.3.

7.4 Results

sIDS+AIS was run over the range of local and distributed attacks listed in §6.6.3 and network and IDS metrics extracted in order to evaluate the performance of the immune-

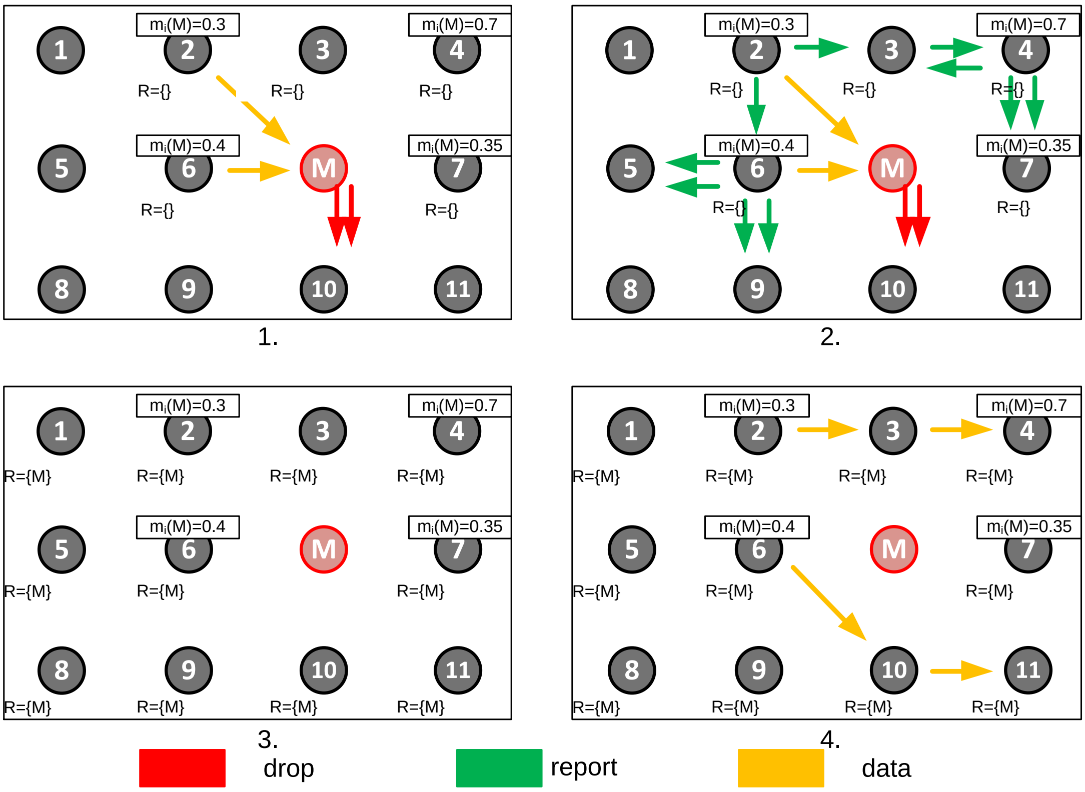


Figure 7.6: Report procedure for adaptive component of AIS. 1. An adversary misbehaves by dropping packets. 2. After a random jitter period, one of the immediate neighbours, node 2, initiates the reporting procedure by forwarding a report containing its evaluation of the suspect's m_i to its neighbours. Nodes forward the reports on and if they have recent experience they transmit a report with their evaluation of m_i (indicated by double arrows). 3. After a timeout, nodes apply a threshold to the reported m_i's, and find the consensus - in this case to respond to M. The empty set R now contains M 4. Nodes find an alternative route around M. Note that neighbours with no experience of M also contribute in the extended response against M.

Table 7.1: AIS IDS parameters

Parameter	Value
REPORT_TIMEOUT	2s
MIN_REPORT_INTERVAL	50s
RESPONSE_BASE	100s
MIN_REPORTS	3
IMPERFECT_THRESHOLD	0.5
REPORT_DIAMETER	3

inspired approaches employed. The results for each of the metrics are superimposed onto the graphs previously obtained to highlight the performance of *sIDS+AIS* versus the baseline and game theoretic networks. The overall response taken by *sIDS+AIS* is formed of a Boolean OR operation on the responses independently initiated by the game theoretic and IDS components.

Each of the parameters for the AIS component were set as shown in Table 7.1, and their values determined empirically.

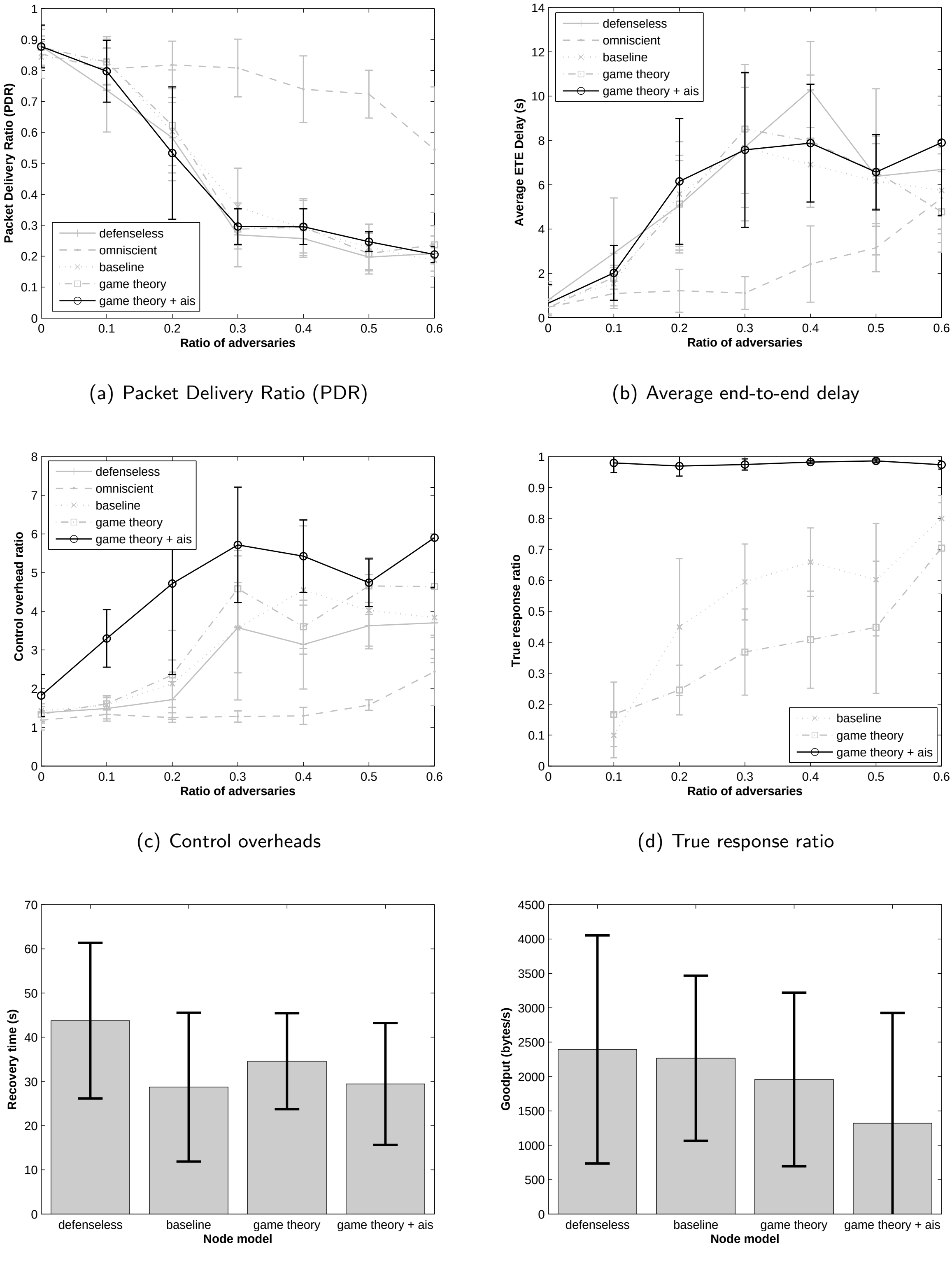
7.4.1 Flooding

The flood scenario used was identical to the description provided in §6.7.1.

Packet Delivery Ratio

Figure 7.8(a) shows the PDR achieved by *sIDS+AIS* when subjected to flooding by an increasing number of adversaries in the network. This attack tests the effectiveness of the innate component since flood detection belongs to the set of perfect detectors that activate a TLR once an intrusion has occurred. Whilst the local response provided by the innate component drops more flood packets than either the baseline or game theoretic IDS due to a near-instantaneous and asynchronous response, queues are still regularly exceeded because a number of flooding packets are still forwarded by IDS nodes until *RREQ_RATE_LIMIT* is reached.

The contention and congestion introduced by the remaining flood packets limits the usefulness of the extended response provided by the innate component as the report packets that are crucial to the initiation of the extended response are frequently dropped at by a neighbour's queue or MAC layer. The result of this is that extended responses are suppressed; neighbours may not observe reports and thus are unable to contribute to the reporting procedure for a given suspect. Moreover, this limits the effectiveness of the synchronisation procedure and reduces overheads by causing all nodes that contribute to the report procedure for a given suspect to back off for *MIN_REPORT_INTERVAL* .



(a) Packet Delivery Ratio (PDR)

(b) Average end-to-end delay

(c) Control overheads

(d) True response ratio

(e) Recovery time (30% adversaries)

(f) Minimum goodput (30% adversaries)

Figure 7.8: Results from flooding scenario with increasing numbers of adversaries

Overall, these data show that there are no significant gains to PDR in this scenario by employing *sIDS+AIS* over the baseline or game theoretic IDS.

Average end-to-end (ETE) delay

Figure 7.8(b) depicts the ETE delay achieved by *sIDS+AIS*. The data show that the rate of increase of ETE delay decreases until 30-40% adversaries, after which the attack saturates, resulting in no further ETE delay or loss in PDR. This characteristic of the ETE delay in *sIDS+IDS* is similar to that of the game theoretic and baseline networks. The attempts to initiate an extended response introduce further overheads, which can be up to twice that of the baseline IDS, yet result in no appreciable performance gain, as shown in Figure 7.8(c).

Response Rate

The high true response rate of *sIDS+AIS* compared to the baseline and game theoretic IDS's (shown in Figure 7.8(d)) is purely a result of the local response provided by the innate component of the AIS. This blacklists a node that is detected for flooding for an exponentially increasing period of time (§7.3.1.3) near-instantaneously, thus dropping any further traffic from this node. This response is in dependant from the slower computation of metrics m_p and m_i used in the pure game theoretic and baseline networks.

Recovery Time and Minimum Goodput

The recovery time and minimum goodput for *sIDS+AIS* is reported in Figure 7.8(e) and Figure 7.8(f) respectively. The data indicate only a slight decrease in the time taken to restore goodput to pre-intrusion levels, but a reduction in the average minimum goodput compared to the pure game theoretic approach.

7.4.1.1 Discussion

As discussed in §6.7.1, flooding leads to large amounts of network congestion and medium contention, which negates one of the main advantages of the AIS approach by suppressing the extended detection and response by neighbours of the node detecting the original intrusion. This often results in the minimum number of reports required to invoke an extended response (*MIN_REPORTS*) not being met or the production of false negative reports from neighbours due to collisions of multiple flooding attacks that result in IDS nodes not observing the full number of flooding packets sent by each adversary. The extra overhead introduced by the extended response tends not to produce retaliatory effect, and is evidenced by the lack of an appreciable gain in any network metric in this scenario.

7.4.2 Unassisted Black hole

Packet Delivery and Dropped Packet Ratio

Figures 7.9(a) and 7.9(b) show the PDR and dropped packet ratio of *sIDS+AIS* respectively, illustrating that there is very little difference in either metric compared to the baseline or pure game theoretic approach to recommend the use of the AIS component in this scenario.

Whilst in the flooding scenario the similar performance to the baseline and game theoretic IDS was caused by a suppressed extended response, in this scenario, although the adaptive component of the AIS enables cross-correlation, few neighbouring nodes are in a position to corroborate on first-hand experience of an intrusion, thus cannot take advantage of this feature. Once more, this results in very few extended responses in the network.

Average end-to-end (ETE) delay and control overheads

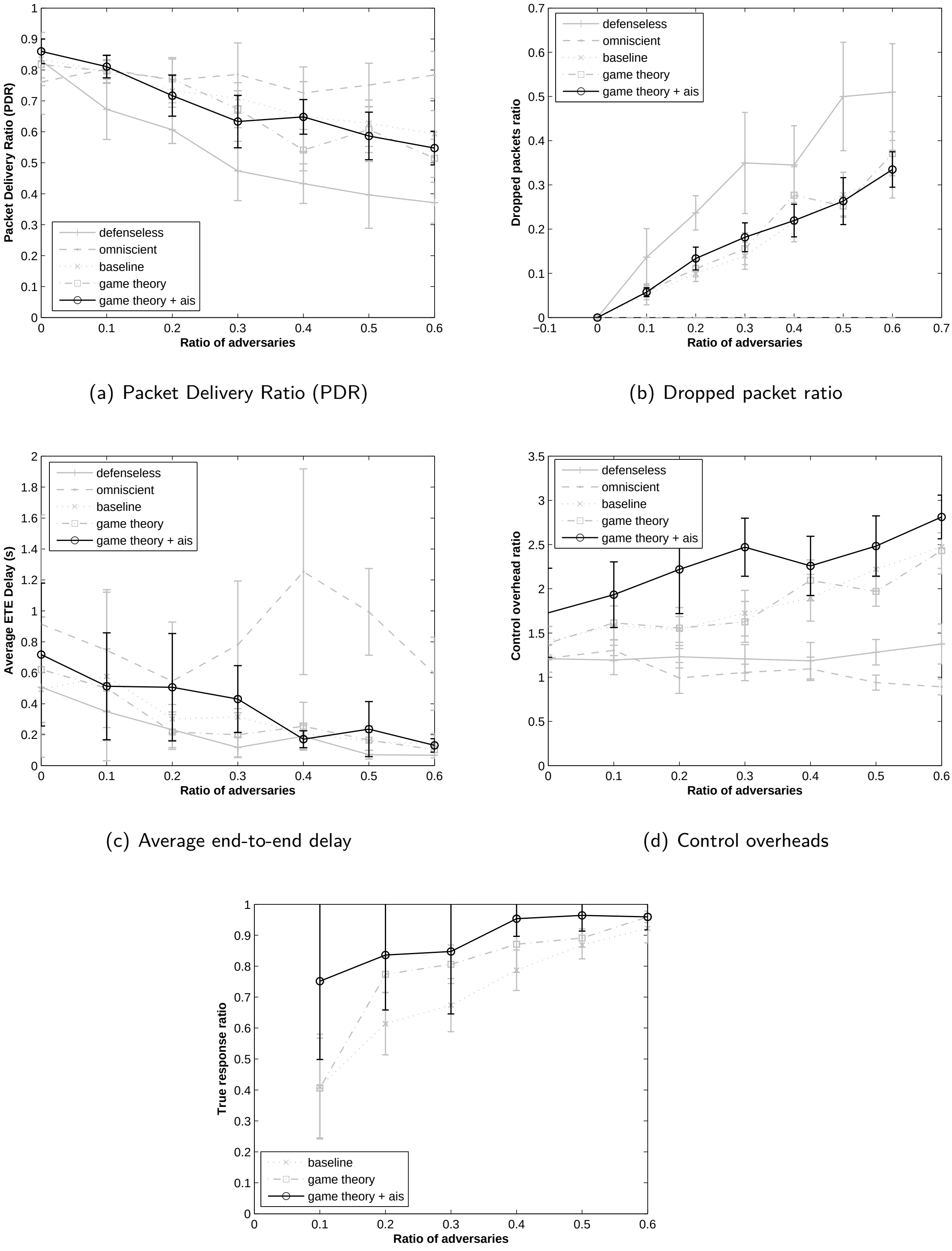
Figure 7.9(c) shows sIDS+AIS introduces a slightly larger ETE delay than the baseline and game theoretic networks. This is as a result of the increased overhead from the extended response of the adaptive component, as shown in Figure 7.9(d). The ETE delay returns to the levels of the baseline and game theoretic networks for greater than 40% adversaries, due to a similar level of overheads.

Response Rate

The response rate of *sIDS+AIS* offers a 5% improvement over the use of a pure game theoretic approach, which showed the best performance previously. This improvement results from the few long-lasting extended responses initiated by the adaptive component which cross-correlates the neighbours evaluations of a suspect's imperfect metrics to reduce the uncertainty, and emerges as less false responses being directed at benign nodes.

Recovery Time and Minimum Goodput

The recovery time and minimum goodput experienced by each of the node models for the unassisted black hole scenario is shown in Figure 7.12. The figure shows *sIDS+AIS* has marginally inferior performance for both metrics compared to the baseline or pure game theoretic approach. The recovery time reported by *sIDS+AIS* is not surprising, however, since the extended response from the adaptive component is only invoked when m_i decreases sufficiently, which would cause a local response in the baseline IDS, thus we are comparing responses that occur on the same timescale.



(a) Packet Delivery Ratio (PDR)

(b) Dropped packet ratio

(c) Average end-to-end delay

(d) Control overheads

(e) True response ratio

Figure 7.9: Results from unassisted black hole scenario with increasing numbers of adversaries

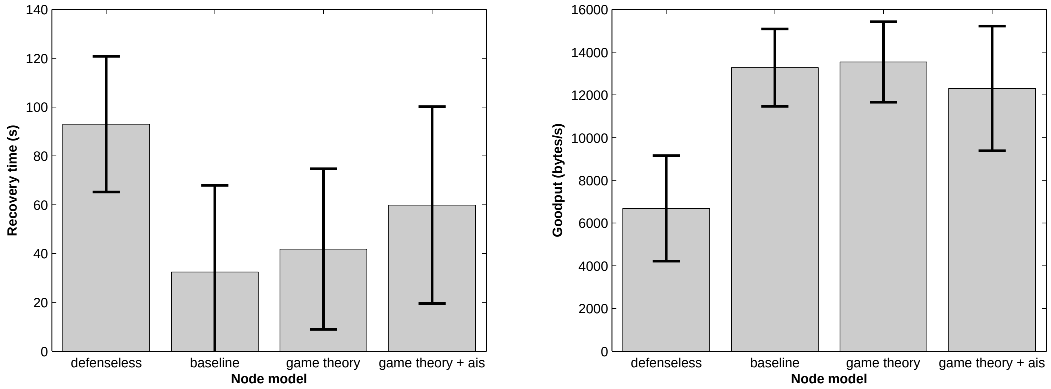


Figure 7.10: Recovery times and minimum goodputs achieved in unassisted black hole scenario with 30% adversaries over range of node models

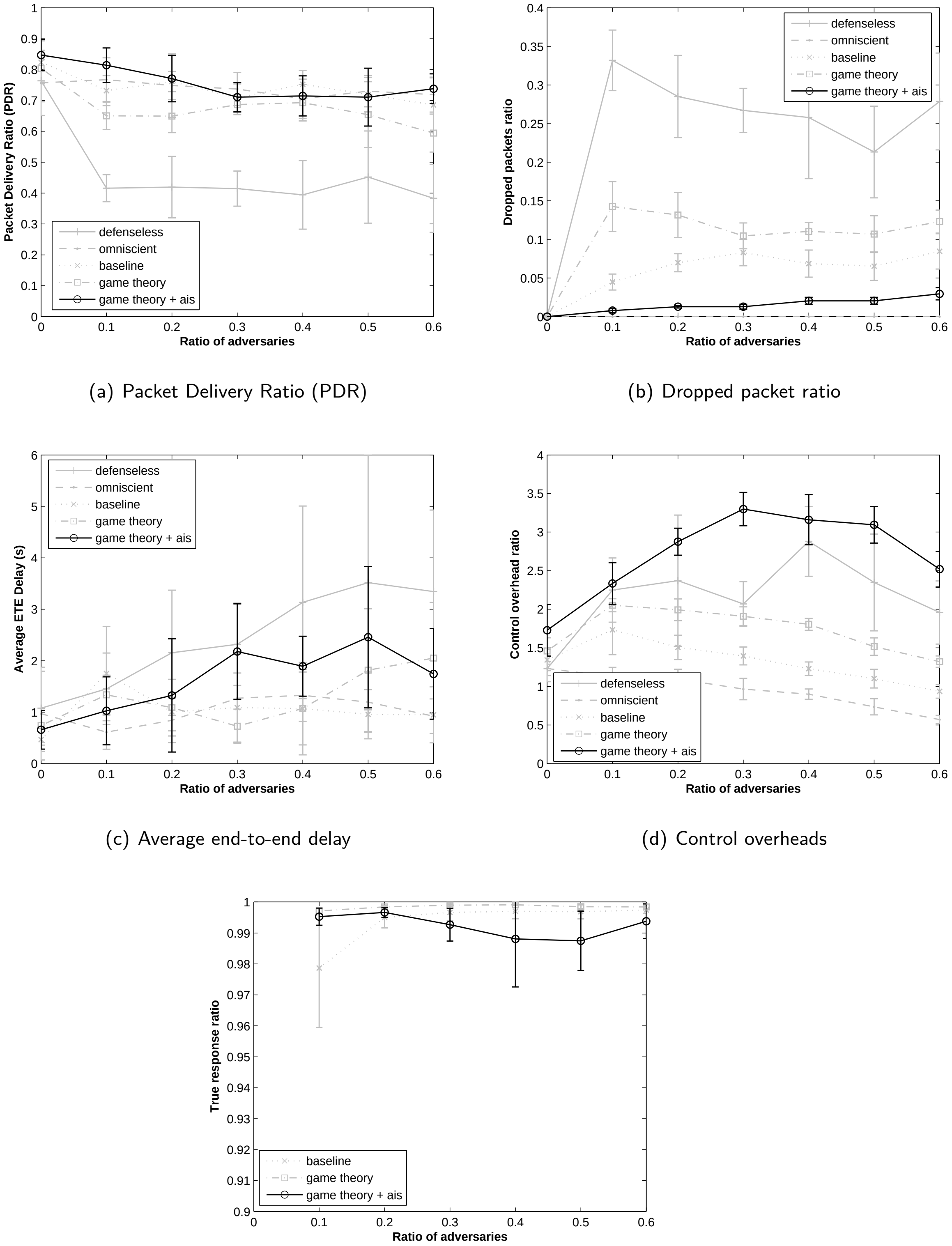
7.4.2.1 Discussion

The forwarding behaviour of a neighbour is captured by the imperfect metric m_i alone, thus the innate component of the AIS does not play a part in the overall response. Although the adaptive response is regularly initiated, as shown by the increased overheads over the baseline and pure game theoretic approach, nodes often fail to reach a consensus decision regarding possible suspects.. This may be the result of weaknesses in the traffic distribution employed in these simulations, which is further compounded by the inability of the drop detector to verify the forwarding of packets on indirect flows (§5.4.1), possible with other ad-hoc routing protocols. Despite this, the approach shows a further advantage in true response rate over the pure game theoretic approach, resulting from appropriate extended responses.

7.4.3 Black hole assisted with passive route invasion

Packet Delivery Ratio

Figures 7.11(a) and 7.11(b) illustrate the PDR and dropped packet ratio for *sIDS+AIS*, indicating a significant reduction (approximately 5%) in the number of dropped packets compared to the leading IDS in this scenario, the baseline IDS. *sIDS+AIS* tracks the PDR achieved by the omniscient and baseline networks, showing a real advantage over the pure game theoretic approach for small ratios of adversaries. That the reduction in drop rate does not result in a proportionate increase in the overall PDR is an interesting result, which could be due to additional queue or MAC drops from the overheads imposed by the reporting procedure, or the routing overheads required to support the additional 5% data preserved.



(a) Packet Delivery Ratio (PDR)

(b) Dropped packet ratio

(c) Average end-to-end delay

(d) Control overheads

(e) True response ratio

Figure 7.11: Results from assisted black hole scenario with increasing numbers of adversaries

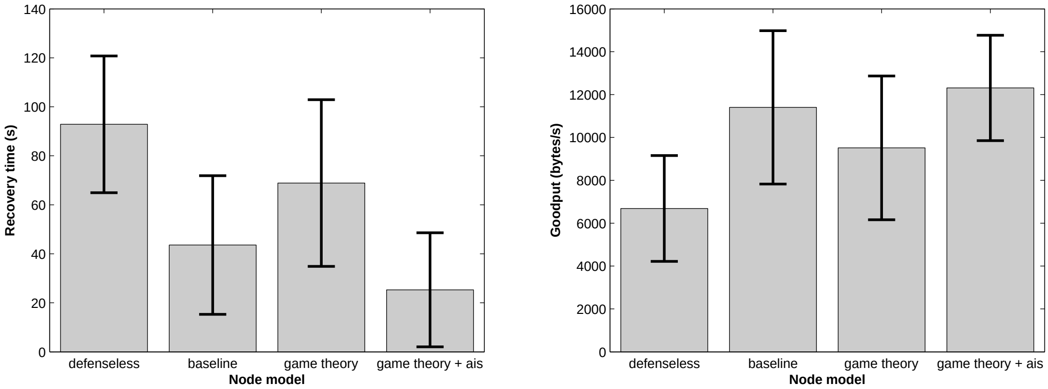


Figure 7.12: Recovery times and minimum goodputs achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in assisted black hole scenario

Average end-to-end (ETE) delay and control overheads

The additional extended responses applied by the innate and adaptive components of the AIS in this scenario account for the larger ETE delays (Figure 7.11(c)) and doubling of the overheads compared to the baseline network (Figure 7.11(d)).

Response Rate

The response rate achieved by *sIDS+AIS* is shown in Figure 7.11(e), indicating a slight reduction in average true response ratio over the pure game theoretic approach. This reduction may be due to the extended detection from the adaptive component of *sIDS+AIS*, in which a percentage of the consensuses reached may be incorrect. These extended false detections lead to more far reaching and longer lasting effects than if a benign node was determined as misbehaving locally.

Recovery Time and Minimum Goodput

sIDS+AIS results in a 42% reduction of recovery time compared to the baseline IDS, which was the previous leader in this scenario (Figure 7.12). This reduction arises from the near-instantaneous response to the strong evidence component of this attack (the passive route invasion). The advantage of employing the AIS framework in *sIDS+AIS* is reinforced by the preservation of a minimum goodput similar to the baseline.

7.4.3.1 Discussion

sIDS+AIS results in a near eradication of the assisted black hole attack at a cost of additional overheads over the pure game theoretic approach and a slight reduction in the true response ratio for larger ratios of adversaries. The extended response proactively identifies adversaries to neighbours that have no prior experience of them, helping to maintain a low drop rate by adversaries. This does not translate into an advantage in PDR, suggesting the overheads from the extended response and delivery of the extra data results in larger MAC and queuing losses.

7.4.4 Wormhole

Packet Delivery and Intercepted Packet Ratio

Figures 7.13(a) and 7.13(b) depict the PDR and intercepted packet ratio for *sIDS+AIS* applied to the wormhole scenario. These data show that the IDS is still susceptible to a large number of intercepted packets for moderate numbers of adversaries, resulting in an artificially high PDR due to intercepted packets taking low latency paths and freeing the shared medium.

At moderate to high numbers of adversaries, employing *sIDS+AIS* does reduce the average number of intercepted packets by between 8.9 and 17% compared to the baseline (previously the most effective approach).

Average end-to-end (ETE) delay and control overheads

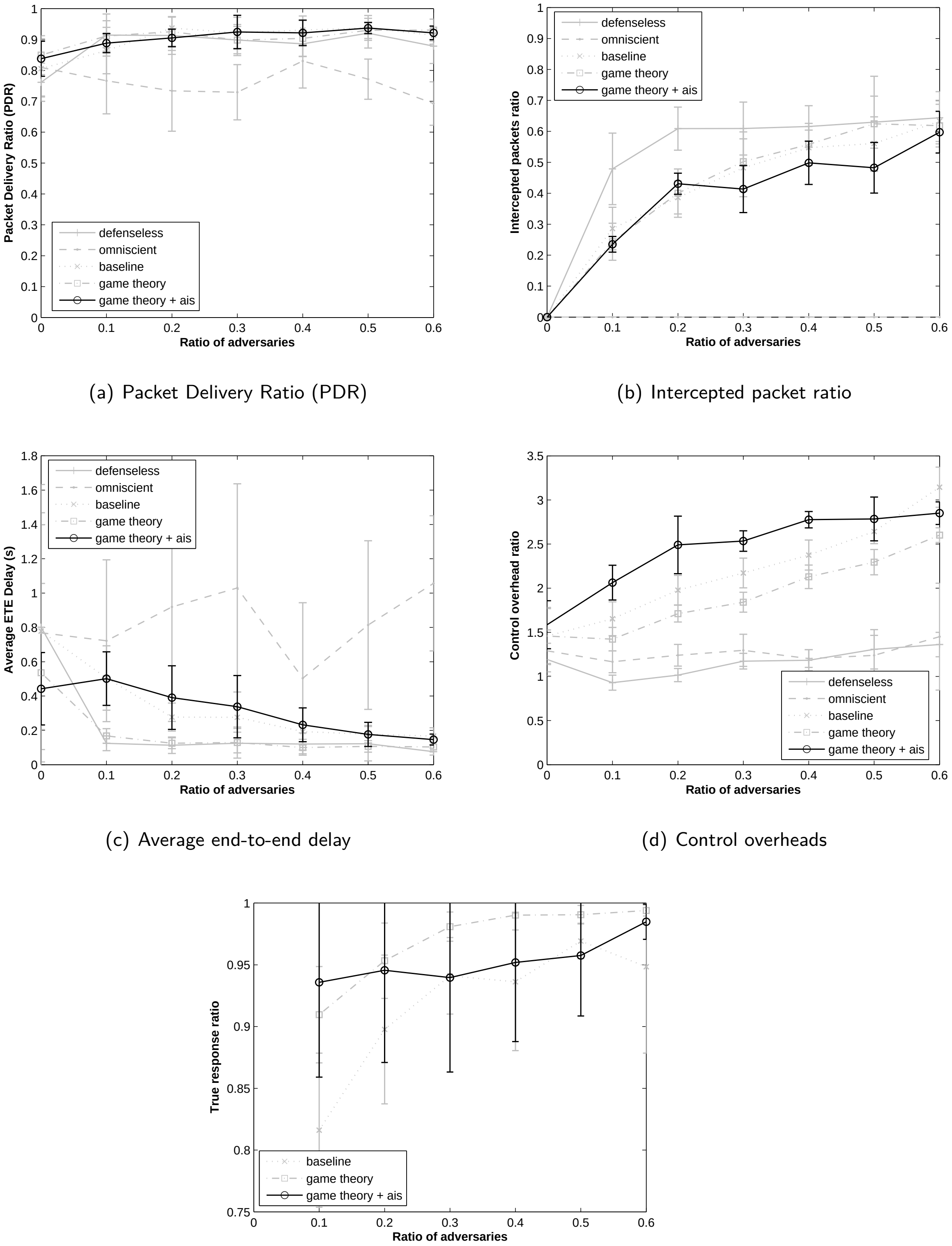
The general trend for a reduction in ETE delay with greater number of adversaries in the wormhole scenario continues with *sIDS+AIS* (Figure 7.13(c)). The decreased number of intercepted packets at moderate levels of adversaries gives rise to a larger ETE delay that reduces to baseline levels at 50% adversaries and above, as the wormhole attack becomes more effective. Extended responses contribute to a 30% increase in overheads at 20% adversaries compared to the baseline IDS, which decreases to a 5% increase for 50% adversaries.

Response Rate

The true response rate achieved by *sIDS+AIS* is shown in Figure 7.13(e), decreasing from the pure game theoretic approach towards the baseline for 30% adversaries and over. This is the result of some consensuses that override the game theoretic reasoning being incorrect. The worst case scenario is a 5% increase in false response rate compared to the pure game theoretic approach for the large advantage in reducing the number of intercepted packets.

Recovery Time and Minimum Intercepted Throughput

sIDS+AIS reduces the recovery time only very slightly compared to the baseline and *sIDS*, which is expected as the innate component does not play as part in the detection of this attack. There is also no noticeable reduction in the maximum bytes intercepted



(a) Packet Delivery Ratio (PDR)

(b) Intercepted packet ratio

(c) Average end-to-end delay

(d) Control overheads

(e) True response ratio

Figure 7.13: Results from wormhole scenario with increasing numbers of adversaries

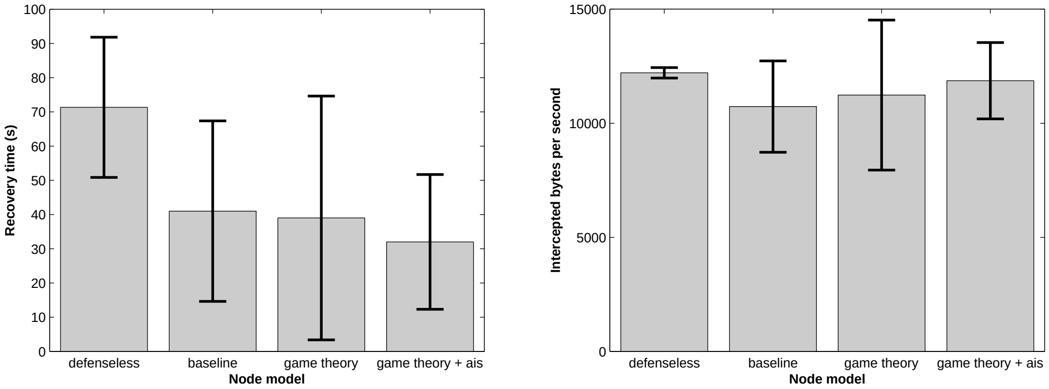


Figure 7.14: Recovery times and minimum intercepted bytes achieved by defenceless, baseline and game theoretic networks with moderate misbehaviour (30% adversaries) in wormhole scenario

by wormhole nodes through employing *sIDS+AIS*, as a consequence of a similar recovery time (Figure 7.14) to the other IDS's.

7.4.4.1 Discussion

There is a clear advantage to running *sIDS+AIS* for large ratios of adversaries as it reduces the number of packets intercepted by between 8.9 and 17% of those intercepted when running the baseline IDS for equivalent numbers of adversaries.

7.4.5 Random misbehaviour

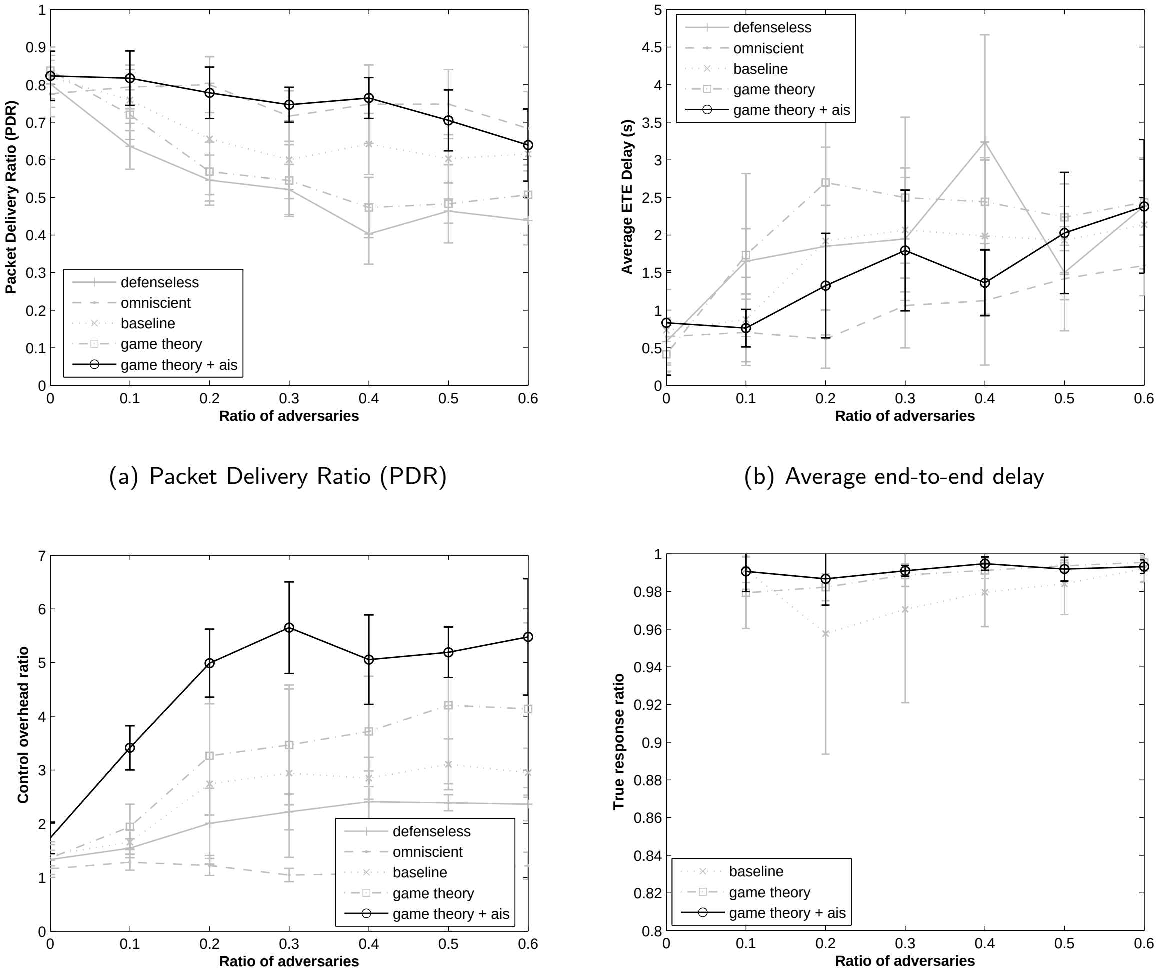
The random misbehaviour scenario was constructed using the misbehaviour distribution constructed in Section 6.7.5.

Packet Delivery Ratio

sIDS+AIS tracks the omniscient network PDR for up to 40% adversaries, and then appears to diminish after 50% adversaries (Figure 7.15(a)) representing a 7-25% advantage over the baseline IDS. This is due to the innate and adaptive components working in tandem to produce long-lasting responses to audit data identified as strong evidence of an intrusion, including passive route invasion and flooding.

Average end-to-end (ETE) delay and control overheads

The ETE delay produced by each network type for the random scenario is shown in Figure 7.15(b). *sIDS+AIS* results in a slightly decreased ETE delay compared to the baseline IDS, on average, as a result of effective blocking of adversaries that flood or attempt to invade routes that would otherwise cause benign nodes to indirectly generate additional



(a) Packet Delivery Ratio (PDR)

(b) Average end-to-end delay

(c) Control overheads

(d) True response ratio

Figure 7.15: Results from random scenario with increasing numbers of adversaries

routing overheads through retransmissions. To support these responses, Figure 7.15(c) illustrates *sIDS+AIS* doubles the control overheads compared to the baseline IDS, a basic result reflected in all previous scenarios.

Response Rate

The true response rate of *sIDS+AIS* is depicted in Figure 7.15(d), and shows similar characteristics to the pure game theoretic approach, responding to adversaries in 98.7-99.5% of cases across all ratios of adversaries tested.

7.4.5.1 Discussion

The metrics shown in Figure 7.15 indicate the benefits of the near-instantaneous innate response and co-ordinated response provided by the AIS in a scenario where adversaries produce audit data with a mixture of strong and weak evidence of an intrusion. Near omniscient level performance was provided by the *sIDS+AIS* across the full range of adversaries, at a cost of greater overheads.

7.5 Summary

This chapter has shown the design and implementation of immune system inspired mechanisms suitable for MANET IDS and their application to *sIDS*, resulting in a multi-layered IDS entitled *sIDS+AIS*. The performance of *sIDS+AIS* was compared to the pure game theoretic and baseline IDS in scenarios employing a range of local and distributed misbehaviour.

The immune inspired approach employed several key aspects of the HIS, including responding over multiple timescales (additionally asynchronously) and the recruitment of nearby agents (in this case, nodes rather than immune cells) to facilitate the overall response against the adversary. The benefit of employing multiple timescales is that intrusions with strong evidence can be responded to near-instantaneously, thereby preventing the attacks spread and reducing further susceptibility, whilst the recruitment of nearby agents enables quicker isolation of the node, reducing the false positives associated with detection schemes that rely on eavesdropping or accurate neighbour lists by cross-correlation of m_i.

The innate response is a long-term response to any misbehaviour that activates a TLR from the perfect detectors described in §5.4. This initiates a cascade in which the detecting node blocks any further traffic from the adversary immediately and an extended response where a consensus decision is made by all neighbours within a radius of the original intrusion leading to a further extended response over a larger set of nodes in the locality of the intrusion. The extended response of the adaptive component occurs over a longer timescale and is a reaction to weak evidence of an intrusion encapsulated in low values of m_i, where intrusion reports are cross-correlated with other neighbouring nodes to reduce the overall imperfection in m_i.

The strength of the multi-layered approach employed by *sIDS+AIS* is shown by the simulation results of the wormhole attack, where the number of intercepted packets was reduced compared to the baseline or pure game theoretic approach, due to the extended response from the adaptive component of the AIS. *sIDS+AIS* also showed a marked reduction in the ratio of packets dropped by adversaries in the assisted black hole scenario. Another significant result is in the random scenario where the overall PDR is vastly improved compared to any of the previous approaches, and tracks the performance of the omniscient network for up to 50% adversaries. The cost associated with this approach that much larger control overheads are required to support the extended reporting procedure that is responsible for some of these performance gains. In the majority of simulations, the total overheads were doubled compared to the baseline. Of the scenarios that remain, the extreme contention present in a flooding environment causes the extended response to be suppressed, whilst in the unassisted black hole, there is not enough corroborative evidence to allow nodes to cross-correlate, which when taken into account with the overheads, makes the system unsuitable for such attacks.

Overall, this chapter has proved the utility of an immune system inspired approach in conjunction with the pure game theoretic approach. A clear advantage in the reduction of attack efficacy resulted in improved end-to-end metrics observed in most studied scenarios over the baseline IDS and delivered high true response rates throughout. The modular design that this IDS is based on means there is the potential for further performance gains by adding new detectors for new forms of attack, which contribute toward the overall behavioural profile of the monitored node.

CHAPTER 8

CONCLUSIONS

Security remains a fundamental research challenge for MANET and distributed systems in general. Pushing the responsibility of core networking functions, such as routing and addressing onto individual agents relies on the co-operation of participants. Full co-operation is rarely the case in practice: nodes in shared environments that belong to multiple authorities may have competing objectives and may not be willing to contribute to the service without the risk of being penalised for poor behaviour or the promise of incentives for good behaviour. Secondly, nodes with a lack of physical protection in hostile environments risk capture and reprogramming by an adversary. These two scenarios illustrate just two cases where adversaries could launch *insider* attacks, which was the main focus of this thesis.

8.1 Summary and Research Contributions

The research question under investigation in this thesis was whether game theoretic and immune system inspired approaches could be applied to improve the quality of intrusion detection in MANET, with a focus on the sources of uncertainty in detection arising from the subtlety of misbehaviour or in the detection strategies themselves. This is of interest and importance because current approaches of misbehaviour detection are not comprehensive enough for application in *imperfect* wireless environments. The uncertainty implicit in eavesdropping leads to a large number of false detections and consequent responses to benign nodes, an effect which worsens with degradation of wireless environment. In addition, IDS' using only local detection techniques are unable to detect distributed attacks, leading to insensitivity to subtle misbehaviour.

The first hypothesis under investigation in this thesis was that employing a game theoretic framework within a distributed IDS would help provide reasoning within these *imperfect* wireless environments. The second was that the application of an immune-inspired approach could be used to improve network end-to-end metrics when applied in conjunction with the pure game theoretic approach.

To meet the aims of the thesis, a simulation-based approach was adopted using a robust methodology that employed Monte-Carlo sampling as discussed in Chapter 3. This required significant development and validation of the Jist/SWANS simulation kernel and flexible adversary models to provide a range of attacking scenarios that could be used to evaluate the performance of the proposed IDS (described in Chapter 4).

A modular intrusion detection system employing distributed detection and response was developed in Chapter 5, using a number of perfect and imperfect detectors. Chapter 6 studied their ability to generate false positives in the absence of misbehaviour in fading environments. Detectors that could generate false positives formed part of the evaluation of m_i, whist detectors that did not formed m_p, the combination of which formed a node's behavioural profile. A baseline threshold-IDS was simulated and the PDR fraction obtained corroborated mostly with the findings of a similar local detection scheme published in Marti *et al.* (2000) in which the authors showed a 20% reduction in PDR between 0% and 40% adversaries in the black hole scenario. Defenseless and omniscient networks provided the upper and lower bounds of each of the studied metrics when applied to each of the scenarios. These results were used to assess the efficacy of the game theoretic and AIS approaches when applied to the developed attacking scenarios over the studied metrics.

The results in Chapter 6 go some way to addressing the first hypothesis under investigation in this thesis. A game theoretic inspired IDS, *sIDS*, was developed, which applied reasoning between the detection and response components of a typical IDS. Unlike the threshold IDS, in which an equal importance was placed on m_p (the perfect metrics) and m_i (the imperfect metrics), the game theoretic IDS employed a 2-player game between the IDS and suspect, designed to give the suspect a 'benefit of doubt' that reduced the ability of m_i (constituting weak evidence) to invoke a response without being accompanied with a reduction of m_p (consisting stronger evidence). As such, these terms formed part of a payoff function, which when the LH algorithm was applied, generated a mixed strategy solution each monitored neighbour. This mixed strategy solution was applied consequently to any packets received from neighbours, to determine whether to drop or process them. The results show that most scenarios studied this approach delivered significant improvements in true response rates. In particular, there was a 5-10% improvement in the wormhole scenario and a 10-15% improvement in the unassisted black hole scenario over the baseline IDS. The use of game theory to improve discrimination in IDS scenarios has been shown in (Otrok *et al.*, 2008a), where the closest measure, false positive rates, was also reduced significantly in their environment. The approach we took deviated from the typical approach of applying game theory to intrusion detection scenarios (Patcha & Park, 2004; Liu *et al.*, 2006) where false positive rates are as encoded as a feature of the game; instead the computation of strategy profiles did not depend on *a priori* known false positive rates of a IDS. An interesting feature of the game theoretic solution is that this advantage is almost always lost beyond 50% adversaries, but it represents an important result for applications where benign nodes are in the majority, where the additional discrimination can be used to bias the operation of the network toward the favour of these nodes. These improvements do come at a cost of additional routing overheads due to a component of the local response that attempts to re-route flows as misbehaving outgoing neighbours are detected. The game theoretic approach also incurs a computational overhead from finding the mixed strategy solution to a game by invoking the LH algorithm on each game instance. To reduce these, the mixed strategy solution is only updated when the neighbour's behavioural profile is recomputed, thus the IDS adopts a cached version of the mixed strategy solution until then.

To address the second hypothesis under investigation in this thesis *sIDS+AIS* was developed, presented in Chapter 7, which further extended the pure game theoretic approach with a number of immune system inspired features. These were a response over multiple timescales as employed by the innate and adaptive components of the immune system, and the recruitment of neighbouring immune cells to participate in a co-ordinated response to an infection. In *sIDS+AIS*, the innate component was comprised of the arrangement of the perfect detectors into TLRs that initiated long-lasting near-instantaneous responses when an intrusion was detected. This was followed by an extended, consensus-based response by neighbours in the locality of the known intrusion. The adaptive component of *sIDS+AIS* monitored long term reductions in m_i and initiated an extended response where nodes with recent, first-hand experience exchanged reports that led to a long-lasting, consensus-based response if the majority of the reports indicated a low m_i, taking advantage of the cross-correlation of observations about a given suspect from the monitors in the locality. These we hypothesised would both improve the end-to-end metrics, and further increase the true response performance because neighbours unaware of the detected intrusion would be able to avoid the affected area entirely and the cross-correlation would reduce the uncertainty implied in the imperfect detection techniques encapsulated in m_i.

The results show that in most scenarios studied, this approach delivered significant improvements in end-to-end network metrics over the pure game theoretic approach. Moreover, some improvements were observed for true response rates, especially in the wormhole and flooding scenarios where the discrimination performance of the baseline or game theoretic approach was unsatisfactory. Notable is the near elimination of packets dropped by adversaries in the assisted black hole scenario, resulting in a PDR that tracks the omniscient protocol. The best result by far is in the random scenario, where the system performs well for a number of intrusions with strong and weak evidence, once again tracking the omniscient protocol. There is a slight reduction in true response ratios for the wormhole and assisted black hole scenario following the application of *sIDS+AIS*, which could be a result of incorrect consensuses occasionally being reached. The consequence of this is that there is a larger reduction on the resulting true response ratio. The main contribution to overheads in the AIS approach is the communication overheads required to initiate an extended response. Computational overheads required to compute a consensus are trivial. To disseminate intrusion reports, data push via intelligent flooding of broadcast packets was chosen. Thus, reports are shared voluntarily without a node specifically requesting them, but the use of broadcast leads to unacceptable performance when conditions are severely degraded (as in the case of flooding). The reduction of false positives by adopting a consensus based approach to weak evidence (encapsulated in m_i) so long as the majority of nodes are not malicious is a well-known result (Zhang & Lee, 2000), but this is the first attempt to the best of our knowledge of employing game theory and features of the immune system within a MANET IDS.

In summary, the research contributions presented in this thesis are as follows: the development of a generic framework intrusion detection MANET in *imperfect* wireless environments, the implementation of a pure game theoretic approach using this generic IDS framework to enhance reasoning and better discriminate benign nodes from adversaries, and the further development of this to include immune-inspired approaches employed to allow co-operation of responses over an extended network area. These contributions have gone some way to meeting to original research aims set out in Chapter 1. It has been shown that the game theoretic and immune system inspired approaches do improve the quality of intrusion detection in MANETs. The pure game-theoretic approach showed improvements in the true response ratio of IDS nodes to better discriminate benign nodes from adversaries, whilst the fusion of game theory and immune system features resulted in improved network metrics, at least when applied to certain attacking scenarios. The additional strength of this work is the modular approach employed to the design of the intrusion detection system that allows new detectors to be added (§5.4) and contribute directly to a node's behavioural profile, once the susceptibility of the detector to cause false positives is understood.

8.2 Future Work

Although we have shown several advantages to the use of a game theoretic and immune system inspired approach to intrusion detection, there remains a great deal of work to fully quantify the bounds of the performance advantages of the corresponding protocols - *sIDS* and *sIDS+AIS*. It was previously noted that the performance of some detectors, and therefore the effectiveness of the extended response, may be dependent on the traffic patterns used in the simulations. The use of more realistic traffic patterns, such as bursty traffic, where the number of flows active at one time is fixed but source-destination pairs vary, may help to provide more instances of weak evidence that can be cross-correlated to improve the overall effectiveness of the extended response. This highlights the general limitation of employing simulations to prove the efficacy of the approach, in which it is desirable to run a greater number of simulations over a wider range of network parameters in order to build confidence in the results presented. Although Monte-Carlo sampling over a wide range of free parameters was employed, varying other network measures such as node density may warrant an investigation to present further rigorous proof of the advantages of the approach. Furthermore, as was noted in the individual discussions, the improvements in true response rate provided by *sIDS+AIS* does not always translate to an advantage in the network metrics chosen to evaluate it. Thus, further evaluation of the efficacy of *sIDS+AIS* at the application layer, where the advantage in true response rate could be applied to bias network operation to favour benign nodes, merits further research.

Additionally, it would also be worth investigating the applicability of *sIDS* to other data centric ad-hoc routing protocols, such as Dynamic Source Routing (DSR), where it is easier to verify the monitored neighbours ability to perform a given function from additional audit data provided by the DSR packet header. An example of this is more complete verification of packet forwarding, where DSR would permit neighbours to indirectly monitor other flows.

Whilst minimising overheads was not the primary focus of this research, it is clear that communication overheads impose an upper limit on the performance gains, especially of *sIDS+AIS*. Computational overheads on the other hand are not quantified during the simulation. Further work is needed to gain an appreciation of the total overheads, and a course of optimisation prescribed. For example, in the game theoretic approach, although the application of LH provides a general solution to NE that is guaranteed to terminate, more specific approaches that compute mixed-strategy solutions only (e.g. described in Rasmusen (2001)) may be more appropriate and reduce the computational complexity significantly.

In *sIDS+AIS*, the main overheads are communication overheads stemming from the extended responses, which use a naive limited flood/gossip technique. This results in scalability problems as the number of adversaries in the network increase. Instead, alternative data dissemination techniques could be evaluated, such as epidemic routing where summary vectors are exchanged instead (Vahdat & Becker, 2000). This would require an investigation into the time-sensitivity of the report procedure and the overall impact this would have on the efficacy of extended responses. If these approaches were to result in little reduction of overheads, it may be worth adapting to the current network conditions, which could be used to switch between the lightweight and more demanding IDS mechanism(s) (taking inspiration from the approach by Liu *et al.* (2006)). This would make the most of the advantage offered by the approaches when they best suit the current observed network conditions.

Furthermore, we only considered one of many ways that immune techniques could be fused with the game theoretic approach. We ran them in parallel, forming the overall response from a Boolean OR operation on the independent responses initiated by the game theoretic and IDS components, but other approaches could be considered. For example, the expensive adaptive response could be initiated only when the mixed strategy solution invokes as response. Alternatively, similarity measures such as the sum of absolute differences on the cross-correlated reports could be used to automatically set the benefit of doubt term used in the game theoretic component, which may result in a tighter integration of the two approaches. Lastly, we considered rational, but non-adaptive malicious behaviour. It would be interesting to see how the performance of *sIDS* deviates when adversaries behave more intelligently by scavenging evidence of responses applied by neighbours to adapt their overall strategy.

In conclusion, it has been shown that employing a reasoned approach through the use of game theory in intrusion detection within *imperfect* wireless environments results in improved discrimination, and together with an immune inspired approach results in increased performance of network metrics in a variety of attack scenarios.

APPENDIX A

COMMON PARAMETERS USED FOR JIST/SWANS SIMULATIONS

Table A.1: Simulation parameters used common to all Jist/SWANS simulations

Simulation Property	Class	Property	Value
Application (Client)	`pkiddie.app. AppUdp. Server`		
Application (Server)	`pkiddie.app. AppUdp. Client`		
Transport	`pkiddie.trans. TransUdp`		
Network	`pkiddie.net.NetIp`		
MessageQueue	`pkiddie.net. MessageQueue. DropMessageQueue`	`capacity`	`50`
Route	`pkiddie.route. RouteAodvMultiInterface`	`RouteAodvBase. AODV_TIMEOUT`	`1*Constants.SECOND`
		`RouteAodvBase. HELLO_INTERVAL`	`1*Constants.SECOND`
		`priority`	`Constants. NET_PRIORITY_CONTROL`
Mac	`pkiddie.mac. Mac802_11`	`promisc`	`true`
Radio	`pkiddie.net. RadioNoiseAdditive`		

PacketLoss	jist.swans.net. PacketLoss.Zero		
Spatial	jist.swans.field. Spatial.Grid	divisions	5
Mobility	jist.swans.field. Mobility. RandomWaypoint	pauseTime	50
		granularity	10
		minSpeed	0
		maxSpeed	10
Placement	pkiddie.field. Placement.Random		
FlowPlacement	pkiddie.flow. FlowPlacement.Random		
Constraint	pkiddie.constraints. Constraint. FlowConnectedConstraint		
FlowProperties	pkiddie.flow. FlowPropertiesHolder .Random		
flows	n/a	n/a	10
endTime	n/a	n/a	500*Constants.SECOND

APPENDIX B

FIXES TO JIST/SWANS

RouteAodv

Fixes applied during validation with ns2

- `RouteAodvBase.HELLO_INTERVAL = 1*Constants.SECOND` - Default of 30s lead to poor performance.
- `RouteAodvBase.AODV_TIMEOUT = 1*Constants.SECOND` - Required for `HELLO_INTERVAL` to be fired each second.

Fixed applied during comparision with published works

- `RouteAodvMultiInterface.MAX_RREQ_BUFFER = 1000` - Default was 10. In a loaded network drops route requests in progress too readily, leading to duplicated RREQs and an increased likelihood of a broadcast storm in highly loaded network with many data flows. Inspired by no limit on the number of RREQs that can be held in the broadcast buffer of the ns2 AODV agent.
- `PrecursorSet.sendRERR` - Previously this method iteratively unicast RRER to each precursor. Now this method broadcasts a RRER if there are more than 5 precursors to be informed, else unicasts to each precursor. As per §6.11 of the AODV RFC (Perkins, 1997):

  > "A Route Error (RERR) message MAY be either broadcast (if there are many precursors), unicast (if there is only 1 precursor), or iteratively unicast to all precursors (if broadcast is inappropriate). Even when the RERR message is iteratively unicast to several precursors, it is considered to be a single control message for the purposes of the description in the text that follows."

- Added `sentBroadcastMessage` boolean, which is set to true if any broadcast messages have been sent to a precursor in a `HELLO_INTERVAL`. This boolean is reset to *false* each `HELLO_INTERVAL`.

- `RouteAodvBase.helloSendEvent()` - Previously this method blindly broadcasted hello messages to each precursor, regardless of whether broadcast messages were sent. Now this method checks to see if a broadcast message was sent in the last `HELLO_INTERVAL` seconds (captured in `sentBroadcastMessage` boolean) to the precursor. Sends a HELLO message to the precursor if `sentBroadcastMessage` is false. As per §6.9 of AODV RFCPerkins (1997):

  > "Every HELLO_INTERVAL milliseconds, the node checks whether it has sent a broadcast (e.g., a RREQ or an appropriate layer 2 message) within the last HELLO_INTERVAL. If it has not, it MAY broadcast a RREP with TTL = 1, called a Hello message ..."

- Added a `bo` (backoff) field to `RouteRequest` instances, used in `RreqBuffer` during route discovery which holds a multiplication factor used to calculate backoff value on repeated attempts to find a RREP for a single destination.

- `RouteAodvMultiInterface.RouteRequest.getBo` - gets the current backoff for the `RouteRequest`

- `RouteAodvMultiInterface.RouteRequest.incBo` - increment the backoff for the `RouteRequest`, when unsuccessful.

- `RouteAodvMultiInterface.computeRREQTimeout()` - Previously, this method did not use binary exponential backoff; instead the timeout was linear with the increased TTL applied to repeated RREQs for a single destination. Now this method performs exponential backoff based on the `NET_TRAVERSAL_TIME`. §6.3. of AODV RFC Perkins (1997) states:

  > "To reduce congestion in a network, repeated attempts by a source node at route discovery for a single destination MUST utilize a binary exponential backoff. The first time a source node broadcasts a RREQ, it waits NET_TRAVERSAL_TIME milliseconds for the reception of a RREP. If a RREP is not received within that time, the source node sends a new RREQ. When calculating the time to wait for the RREP after sending the second RREQ, the source node MUST use a binary exponential backoff."

- Any `RouteErrorMessages` are created with TTL=1, rather than 64 as used by default in `DEFAULT_TTL`, reducing the likelihood of a broadcast storm of RRERs. §6.11 of AODV RFC Perkins (1997) states:

  > "...the RERR is typically sent to the local broadcast address (Destination IP == 255.255.255.255, TTL == 1) with the unreachable destinations, and their corresponding destination sequence numbers, included in the packet..."

- `RouteRequest.broadcast()` - Previously, this method used an expanding ring technique, until `TTL_THRESHOLD` was reached, when the destination was returned as unreachable. Now, route Requests are disseminated using a expanding ring search technique, until `TTL_THRESHOLD` reached. Then, for `RREQ_RETRIES` times, attempts are generated with TTL = `NETWORK_DIAMETER`. §6.4 of the AODV RFC Perkins (1997) states:

  > "To prevent unnecessary network-wide dissemination of RREQs, the originating node SHOULD use an expanding ring search technique...
  > *... This continues until the TTL set in the RREQ reaches TTL_THRESHOLD, beyond which a TTL = NET_DIAMETER is used for each attempt.*"

- `RouteAodvMultiInterface.generateRouteReplyMessage(...) -` If an intermediate node was to generate an advertisement for a destination, the destination sequence number of the node (and not the destination) was incorrectly being copied into the advertisement. Updated this method such that if a route reply message is being generated by an intermediate node, then the destination sequence number in the route entry for the destination is used. §6.6.2 of the AODV RFC Perkins (1997) states:

  > "If the node generating the RREP is not the destination node, but instead is an intermediate hop along the path from the originator to the destination, it copies its known sequence number for the destination into the Destination Sequence Number field in the RREP message..."

- `RouteAodvMultiInterface.generateRouteReplyMessage(...) -` Nodes own sequence number was incorrectly being updated. Updated method such that the nodes own sequence number is ONLY incremented if it is responding to the RREQ as the destination and the nodes own sequence number is less than or equal to the one in the RREQ. §6.6.1 of the AODV RFC Perkins (1997) states:

> "If the generating node is the destination itself, it MUST increment its own sequence number by one if the sequence number in the RREQ packet is equal to that incremented value. Otherwise, the destination does not change its sequence number before generating the RREP message."

`RadioNoiseAdditive`

- `endTransmit` - This method is overridden from `RadioNoise` (the base class of each radio implementation). Now, when the radio has signals after transmitting, it goes into sensing mode, and not into receiving mode. Credit to: http://www.vanet.info/node/14.

`Mac802_11`

- `pauseBackoff` - This method was updated to adhere to 802.11 standard, §9.2.5.2 - Backoff procedure. The current implementation caused more collisions resulting in decreased throughput. The result of this fix and `RadioNoiseAdditive` fix is that a further bug where nodes are permanently suppressed in a backoff state was fixed. Credit to: http://www.cs.technion.ac.il/~gabik/Jist-Swans/mac/BUG-NonDiscrete-Backoff.html.

- `sendDataUnicast` - This method was updated to supress duplicate messages, adhering to the 802.11 standard. The result of this fix is that there are no more duplicated messages at the CBR application layer. Credit to: http://www.cs.technion.ac.il/~ gabik/Jist−Swans/mac/BUG−Duplicate−Elimination.html.

`NetIp`

- `protocolHandlers` now grows dynamically to prevent unnecessary exceptions. Credit to: http://www.vanet.info/node/14.

`NetMessage.Ip`

- added `logId` field as integer type. This is output as `pid` field in log file and prevents wrap around of id in large simulations. id, the ip identifier field, is left unchanged as short type, but output as `iid` in trace files.

`MessageQueue.DropMessageQueue`

- `insert` - Drop and log the newest message (drop tail queue) instead of terminating the simulation.

APPENDIX C

EXTENSIONS TO JIST/SWANS

C.1 Trace file generation

Jist/SWANS v1.0.6 had no default mechanism to output simulation statistics so post-processing for the purpose of collecting network metrics was impossible. A number of community projects were available, such as Javis for Jist/SWANS (Kliot, 2009) and XTend-SWANS (no longer available online), but they logged a small number of the possible simulation events, which was too restrictive for our purposes. The log4j library (Apache Software Foundation, 2010), included within Jist/SWANS, was re-purposed for trace output. A trace format was defined suitable for easy post-processing as a set of tab delimited fields and key-value pairs, corresponding to the following format:

```
log level (INFO,DEBUG...)| event source class | simulation time (ns)|
event source (ip address)| event | event_details
```

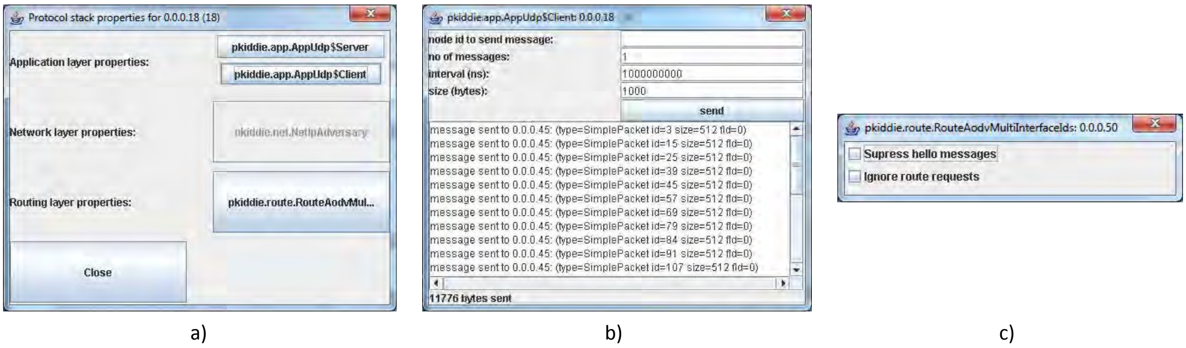
This trace format was applied to all events across all existing layer implementations. The following shows a trace from the network layer and corresponds to the node with IP address 0.0.0.1 receiving a route request from a node with mac address 68 at 10009562861ns:

```
INFO [NetIpBase]: 10009562861 0.0.0.11 receive: from=68 on=1 data=(src
=0.0.0.68 dst=ANY size=44 type=ip prot=123 ttl=5 pid=13 iid=13 route=[]
src_route=null data=(hopCount=0 rreqId=0 destIp=0.0.0.54 destSeqNum=0
origIp=0.0.0.68 origSeqNum=1 type=rreq))promisc=false
```

C.2 Simulation user interface

`GuiLog` is a user interface for Jist/SWANS that enabled the user to view events in the simulation event queue as they were processed (Fong, 2004). This type of output failed to provide an 'at a glance' overview of the simulated MANET, so `GuiLog` was extended to provide a graphical overview of the MANET topology and events of interest as they

Figure C.2: a) Node protocol stack in simulation user interface, b) AppUdp application layer panel, c) Routing layer panel


Protocol stack properties for 0.0.0.18 (18)

Application layer properties: pkiddie.app.AppUdp$Server pkiddie.app.AppUdp$Client

Network layer properties: pkiddie.net.NetIpAdversary

Routing layer properties: pkiddie.route.RouteAodvMul...

Close

a)

pkiddie.app.AppUdp$Client: 0.0.0.18

node id to send message:
no of messages: 1
interval (ns): 1000000000
size (bytes): 1000

send

```
message sent to 0.0.0.45: (type=SimplePacket id=3 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=15 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=25 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=39 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=45 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=57 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=69 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=79 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=84 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=91 size=512 fld=0)
message sent to 0.0.0.45: (type=SimplePacket id=107 size=512 fld=0)
```

11776 bytes sent

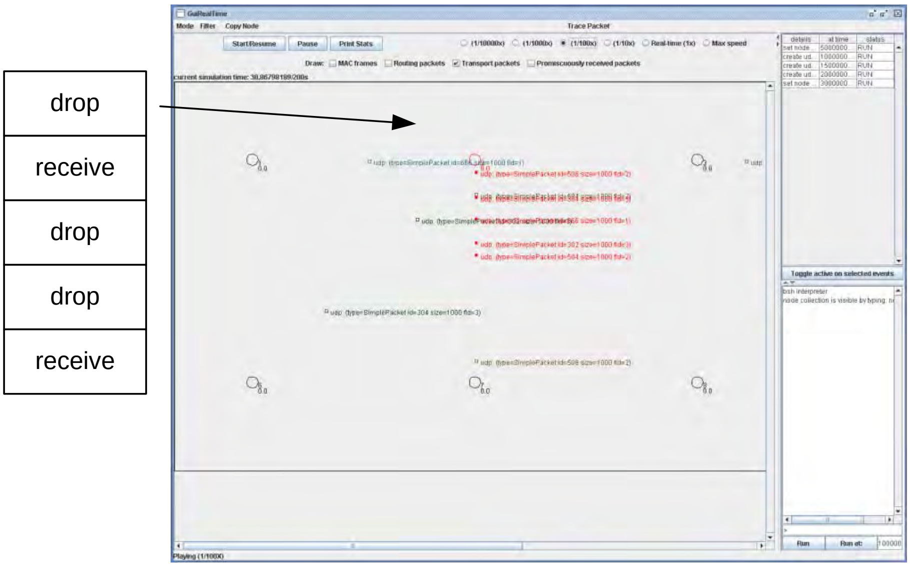
b)

pkiddie.route.RouteAodvMultiInterfaceIds: 0.0.0.50

□ Supress hello messages
□ Ignore route requests

c)

were processed by the simulator, including received packets, dropped packets, and node movement (Figure C.1), with options to increase and decrease the speed of simulation execution. The ad-hoc network user interface aided debugging of the Jist/SWANS simulation models as and eased future development of the misbehaviour models and protocols.



GuiRealTime

Mode Filter Copy Node Trace Packet

Start/Resume Pause Print Stats ○ (1/10000x) ○ (1/1000x) ● (1/100x) ○ (1/10x) ○ Real-time (1x) ○ Max speed

Draw: □ MAC frames □ Routing packets ☑ Transport packets □ Promiscuously received packets

current simulation time: 38.86798189/200s

details	at time	status
set node ...	5000000	RUN
create ud...	1000000	RUN
create ud...	1500000	RUN
create ud...	2000000	RUN
set node ...	3000000	RUN

```
udp: (type=SimplePacket id=686 size=1000 fld=1)
udp: (type=SimplePacket id=506 size=1000 fld=2)
udp: (type=SimplePacket id=302 size=1000 fld=3)
udp: (type=SimplePacket id=504 size=1000 fld=2)
udp: (type=SimplePacket id=304 size=1000 fld=3)
udp: (type=SimplePacket id=508 size=1000 fld=2)
```

Toggle active on selected events

bsh interpreter
node collection is visible by typing: n

Run Run at: 100000

Playing (1/100X)

Figure C.1: Simulation GUI, showing SWANS event stream.

Furthermore, protocol layer classes could contribute a panel presenting a number of options available on the given protocol layer. Figure C.2 shows the protocol stack for a simulated node at simulation initialisation. For example, an UDP application layer panel was developed to force the selected instance to send network traffic to a given destination, and an AODV routing layer panel developed to disable routing behaviour on-demand.

C.3 Driver base class

The majority of code in simulation drivers (§??) was common 'boilerplate' code that tied adjacent protocol layer entities together for each node in the simulation. For every driver, this code had to be repeated, thus to reduce code maintenance, a `DriverBase` driver containing the common code was created. Future drivers were created from `DriverBase`, which also provided a number of other common data structures that assisted in simulation development. For example, a `Node` data structure kept references of the simulated protocol layer entities for each node. This was important for allowing protocol layers to be reconfigured during simulation time, to simulate node capture or invoke traffic generation.

C.4 Jist/SWANS kernel

The Jist/SWANS rewriter prohibited derived classes from being defined as proxy entities for use in network simulation. This reduced the advantages that the simulator had over competing simulators through the use of Java object oriented code: inheritance allows code reuse and specific functionality to be overridden as necessary. Thus, the kernel was extended to permit derived classes, with an aim to simplify the creation of adversary models in the future by requiring them to override only the necessary methods. The aim was for `RouteAodvMultiInterfaceAdversary`, the adversarial router, to be composed of the `RouteAodvMultiInterface` class that defined the AODV protocol, itself composed from `RouteAodvBase`, which defined the common fields and data structures (route table, route table entries, message queues) used in all AODV implementations. Then, to create any number of attacks, `RouteAodvMultiInterfaceAdversary` just overrode appropriate methods and the could still benefit from any bug fixes applied to its base classes.

Most other layers were refactored to derive from a base class, which helped to simplify logging code and the binding of adjacent layers by `DriverBase`. Figure C.3a) shows the broken inheritance model of unmodified Jist/SWANS, using `RouteAodv` as an example, whilst the revised hierarchy is presented in Figure C.3b).

C.5 Test framework

In network simulations, a standard testing practise is to run a simulation and compute the relevant metrics to determine if it is in line with expectations. If it is not, then the user investigates the cause, which can be labour intensive as it can often resort to manually iterating through a network trace. In this thesis, unit testing was used to repeatedly test small parts of code during the development of the adversary models and intrusion detection schemes. This proactively highlighted regressions during the development process.

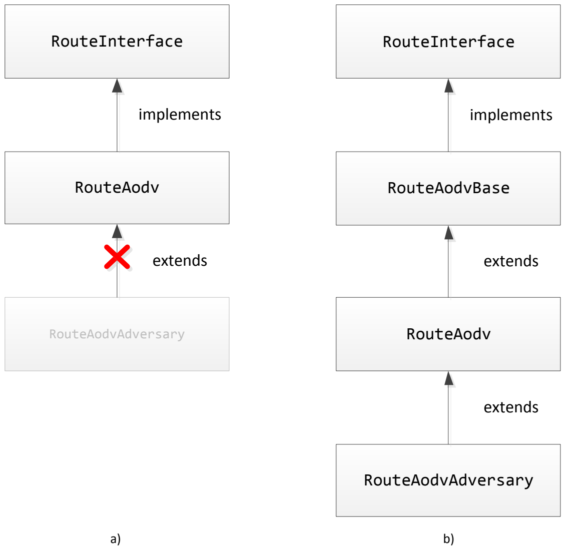


Figure C.3: a) Jist/SWANS v1.0.6 class hierarchy; Jist rewriter specifically prohibits derived classes. b) After modification of Jist/SWANS kernel, and extraction of basic functionality into `RouteAodvBase`. Functionality in `RouteAodv` is overridden in `RouteAodvAdversary` only where necessary to implement misbehaviour.

In Java, unit tests are typically run in a unit testing framework such as JUnit, but JUnit and Jist/SWANS are incompatible due to the bytecode rewriting Jist/SWANS does. Thus, inspiration was taken from JUnit and a simple unit test framework entitled `ReallySimpleSwansUnitTestingFramework` was developed to enable unit testing of Jist/SWANS simulation entities. A number of test methods, written in Arrange, Act, Assert syntax (AAA) (Hamill, 2004) were produced. These methods are annotated with `@Test` optionally specifying the simulation time which the test should be executed in, by specifying a *time* parameter, and a *timeout* parameter, after which the test fails if the assertions have not evaluated to true in the specified interval. Simulation time can be incremented by calling `JistAPI.sleepBlock()`, whilst `Assert.assertTrue(bool)` asserts that simulation entities under test have an expected state at the new simulation time. When a number of tests are run under the test framework, the framework produces a 'pass' or 'fail' along with the name of test, which can be used to identify problems with the method of the simulation entity that is under test. A simple example is shown in Listing C.1.

The test `testRouteToSelf()`, described in Listing C.1, checks that a route entry to self is created when an instance of `RouteAodvMultiInterface` is created. The test begins at the simulation time *t=1s* by creating an instance of the router and starting it. Simulation

Listing C.1: A unit test to check an route entry to self is created when an instance of *RouteAodvMultiInterface* is created.

```
@Test(time=1*Constants.SECOND)
public void testRouteToSelf() {
   NetAddress netAddr = new NetAddress(1);
   RouteAodvMultiInterface routeAodv = new RouteAodvMultiInterface(netAddr);
   routeAodv.getProxy().start();
   JistAPI.sleepBlock(1*Constants.SECOND);
   RouteTableEntry entry = routeAodv.getRouteTable().lookup(netAddr);
   Assert.assertNotNull(entry);
}
```

time is incremented to *t=2s*, by which time it is expected that there should be a single entry in the route table with the network address of the node (`netAddr`) known as the *self* entry. The test framework produced the following output:

```
Method: testRouteToSelf, Passed: true
```

C.6 Source code changes

Source code wide changes

- numerous debug/info logging statements added using built in log4j library,
- `LogUtil.formatLog` for standardised log formatting across all the layers of the protocol stack.

`RouteAodvMultiInterface`

- common functionality factored into `RouteAodvBase` and shared by all `RouteAodv` types.
- router is multiple interface aware. `interface_id` field added to routing table entries.
- allow setting of network layer (IP) priority that routing packets are sent as.

`jist.Runtime.Rewriter`

- `ignoredPackages` - added "org.w3c.", "au.com.bytecode.opencsv", "org.netbeans.", "org.apache.commons", "edu.uci.ics.jung". All are external libraries used in research that should not be rewritten by the simulator.
- constructor - `rewriterTime` set to 0 if `clRewriter` returns null. Prevents `NullReferenceException` being thrown on occasion.

- `doClass()` - removed `Constants.ACC_FINAL` from generated method `mg` (both times), which permits derived, rewritten classes, with no noticeable side-effects.

`jist.Runtime.Controller`

- manages gui creation, should the user require it.
- `eventLoop()` - at end of simulation (i.e. when `JistException.JistSimulationEnd Exception` is thrown), close gui instance.

APPENDIX D

SIMULATION SOURCE CODE

Full source code to the updated Jist/SWANS environment (including bug fixes), node models and post-processing scripts are available at http://www.eee.bham.ac.uk/com_test /dsnl/kiddiep/index.html as a zip file. In particular:

- `src\pkiddie` contains a mirror of most of the Jist/SWANS simulation engine with the fixes and additions listed in Appendix B and Appendix C respectively.
- post-processing scripts can be found in `src\pkiddie\tools\perl`.
- ns2 source code to create a simulation from a Jist/SWANS trace can be found in `src\pkiddie\tools\ns2`
- simulation drivers are located in `src\pkiddie\drivers`.

To compile, an Ant build script has been provided in the root of the zip file entitled `build_pkiddie.xml`, and for simplicity, the whole distribution can be compiled under Eclipse by setting the builder to use `build_pkiddie.xml` in the project properties dialog. Detailed instructions provided on the website. **Please ensure you are compiling with the Java 1.5 class format, as there are known issues with the BCEL library and Java 1.6+ class format.**

The source code for the baseline IDS can be found at `src\pkiddie\ids\IdsBaseline .java`.

The source code for the game theoretic IDS includes `src\pkiddie\ids\IdsGameTheory .java` and the Lemke-Howson solver at `src\pkiddie\gametheory\lemke\*`.

The source code for the game theoretic/AIS IDS are all located in `\src\pkiddie\ids` and include `IdsAisGameTheory.java, Ais2.java, GameTheory.java, InnateReport*. java, Report*.java` and `Buffer*.java`.

Each of the IDS depend on the set of detectors implemented in `src\pkiddie\ids \detection`, the custom network layer `\src\pkiddie\net\NetIpIds.java` and routing layer `src\pkiddie\route\RouteAdovMultiInterfaceIds.java`.

Finally, source code for implementing the types of misbehaviour described in Chapter 4 may be found in `src\pkiddie\route\RouteAodvMultiInterfaceAdversary.java`.

APPENDIX E

USING THE UNIT TEST FRAMEWORK

To run a test class, e.g. xxxTest.java, under the `ReallySimpleSwansUnitTestingFramework` test framework:

- Add a static main method to the test class and insert the following code:

  ```
  public static void main(String[] args) {
     ReallySimpleSwansUnitTestingFramework tester = new
     ReallySimpleSwansUnitTestingFramework();              tester.
     run(xxxTest.class);      }
  ```

- Then invoke the Jist/SWANS runtime with the argument xxxTest.java (assuming it resides in the tests package)

  ```
  java jist.runtime.Main jist.swans.Main tests.xxxTest
  ```

`ReallySimpleSwansUnitTestingFramework` analyses the test class, and runs each test method (a method that was annotated `@Test`) sequentially on the Jist/SWANS event queue (providing access to simulation time).

The results are output on the console in the following format:

```
Method: testRouteToSelf , Passed: true
```

APPENDIX F

VALIDATING JIST/SWANS

Two validation studies were completed by comparing results from published simulation studies of AODV and employing the same scenario in Jist/SWANS using the final version of `RouteAodvMultiInterface` with the additions and bug-fixes listed in Appendix B.

F.1 Validation I

The network environment used in Perkins *et al.* (2001) was chosen to validate the Jist/SWANS AODV router and implemented in the driver `Perkins2001.java`. Table F.2 lists the parameters that were used. The independent variables were `flows` and `pauseTime`, and were driven by the Monte-Carlo runner, which enumerated over each possible combination of $flows = \{10, 20, 30, 40\}$ and $pauseTime =$ 0-900 with step size of 100, to produce the mean PDR and end-to-end delay plots shown in Figures F.1 and F.2 respectively. Table F.1 details the flow pattern used in this simulation.

Table F.1: Properties of `AppUdp` traffic generator in Validation I scenario

Property	Value
packet size	512 bytes
start time	10s
interval	0.5s
type	udp
burstTime	random(min=40s,max=50s)
pauseTime	random(min=10s,max=20s)
reps	15

Table F.2: Simulation parameters used in Validation I scenario

Simulation Property	**Jist/SWANS**			**ns2**		
	Class	**Property**	**Value**	**TCL Class**	**Property**	**Value**
Application (Client)	pkiddie.app. AppUdp.Server			Agent/ NULL		
Application (Server)	pkiddie.app. AppUdp.Client			Application/ Traffic/ CBR		
Transport	pkiddie.trans TransUdp			Agent/ UDP		
Network	pkiddie.net.NetIp			n/a		
Message Queue	pkiddie.net. MessageQueue. DropMessageQueue	capacity	50	Queue/ DropTail/ PriQueue	ifqlen	50
Route	pkiddie.route. RouteAodv / pkiddie.route. RouteAodvMultiInterface			AODV		

Mac	pkiddie.mac. Mac802_11			Mac/802_11	dataRate_	1Mb
					basicRate_	1Mb
Radio	pkiddie.net. RadioNoiseAdditive			Phy/ WirelessPhy	freq_	2.4e+9
					RXThresh_	7.23131e-11
					bandwidth_	1Mb
Spatial	jist.swans.field. Spatial.Grid	divisions	5	n/a		
Mobility	jist.swans.field. Mobility.RandomWaypoint	pauseTime	0	*driven by Jist/SWANS*		
		granularity	10			
		minSpeed	0			
		maxSpeed	20			
Field		bounds	2000,2000	*driven by Jist/SWANS*		
Pathloss	jist.swans.field. Pathloss.FreeSpace			Propagation/ FreeSpace		

Placement	pkiddie.field. Placement.Random			*driven by Jist/SWANS*		
Flow Placement	pkiddie.flow. FlowPlacement.Random			*driven by Jist/SWANS*		
Constraint	pkiddie.constraints. Constraint. FlowConnected Constraint			n/a		
Flow Properties	pkiddie.flow. FlowPropertiesHolder. Random			n/a		
flows	n/a	n/a	10	*driven by Jist/SWANS*		
endTime	n/a	n/a	500s	*driven by Jist/SWANS*		
nodes	n/a	n/a	50	*driven by Jist/SWANS*		

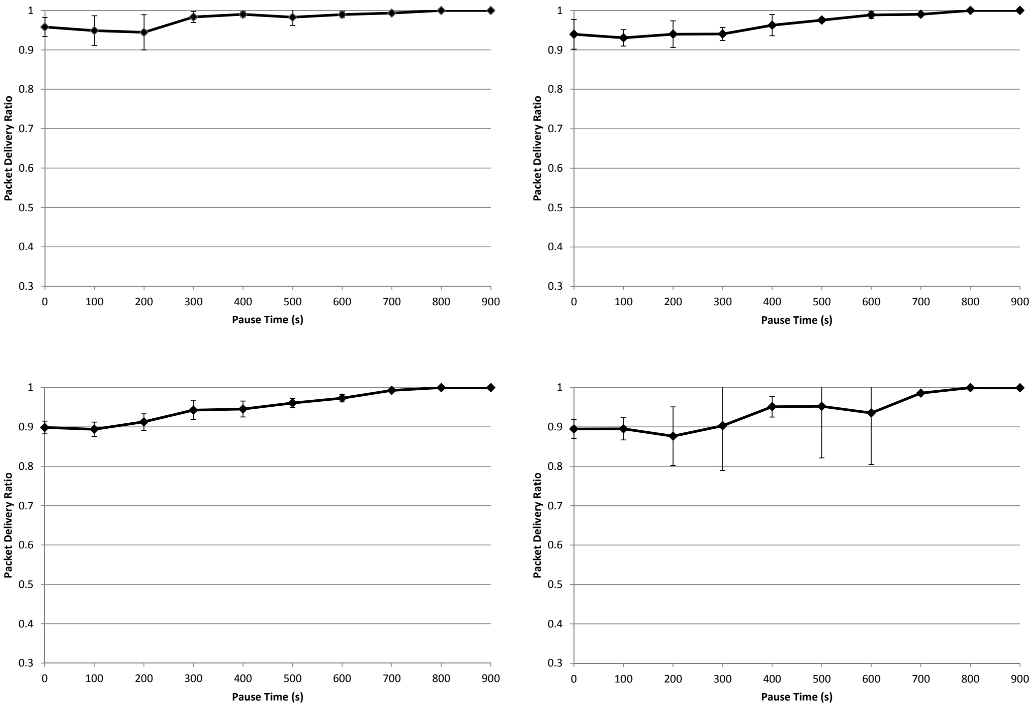


Figure F.1: PDRs achieved in Validation I scenario, using `RouteAodvMultiInterface` (v2).

The resulting network metrics shown in Figures F.1 and F.2 show the resulting metrics obtained in Jist/SWANS were slightly better than the published results for smaller numbers of flows, an effect that increases as more flows are present.

F.2 Validation II

The performance of `RouteAodvMultiInterface` was also compared to Takai *et al.* (2001), who published a study illustrating routing protocol scalability and the corresponding degradation of PDR and end-to-end delay when fading and pathloss propagation models were applied to simulated networks. We were interested how comparable the metrics generated by Jist/SWANS were to this work when applying the same propagation models, and created `Takai2001.java`, set up as shown in Table F.3.

Table F.3: Simulation parameters used in Validation II scenario

Simulation Property	Class	Property	Value
Application (Client)	`pkiddie.app. AppUdp. Server`		

Application (Server)	pkiddie.app. AppUdp. Client		
Transport	pkiddie.trans. TransUdp		
Network	pkiddie.net.NetIp		
MessageQueue	pkiddie.net. MessageQueue. DropMessageQueue	capacity	50
Route	pkiddie.route. RouteAodvMultiInterface	RouteAodvBase. AODV_TIMEOUT	1*Constants.SECOND
		RouteAodvBase. HELLO_INTERVAL	1*Constants.SECOND
		priority	Constants. NET_PRIORITY_CONTROL
Mac	pkiddie.mac. Mac802_11	promisc	true
Radio	pkiddie.net. RadioNoiseAdditive		
PacketLoss	jist.swans.net. PacketLoss.Zero		
Spatial	jist.swans.field. Spatial.Grid	divisions	5
Fading	jist.swans.field. Fading.Rayleigh		
Pathloss	jist.swans.field. Pathloss.TwoRay		
Mobility	jist.swans.field. Mobility. RandomWaypoint	pauseTime	0
		granularity	10
		minSpeed	0
		maxSpeed	20
Field		bounds	new Location. Location2D (1200,1200)

Placement	`pkiddie.field.Placement.Random`		
FlowPlacement	`pkiddie.flow.FlowPlacement.Random`		
Constraint	`pkiddie.constraints.Constraint.FlowConnectedConstraint`		
FlowProperties	`pkiddie.flow.FlowPropertiesHolder.ForAll`		
flows	n/a	n/a	`40`
endTime	n/a	n/a	`500*Constants.SECOND`
nodes	n/a	n/a	`100`

The `RadioNoiseAdditive` model can be created using either Bit Error Rate (BER) or Signal To Noise Ratio Threshold (SNRT). For this study, SNRT mode was chosen. The end time and queue length of 50 was assumed since the original study based the simulation environment from Perkins *et al.* (2001). The traffic pattern used is described in Table F.4.

Table F.4: properties of flows in this scenario

Property	Value
packet size	512 bytes
start time	10s
interval	0.376s
type	udp
burstTime	random(min=40s,max=50s)
pauseTime	random(min=10s,max=20s)
reps	7

Figure F.3 shows the PDR and end-to-end delay achieved by Jist/SWANS and the study by Takai *et al.* (2001), using ns2, illustrating that whilst the PDR obtained by Jist/SWANS is within the margin of error, the end-to-end delay was as much as 40% more on average. It is unclear from the published results whether they are the average of several

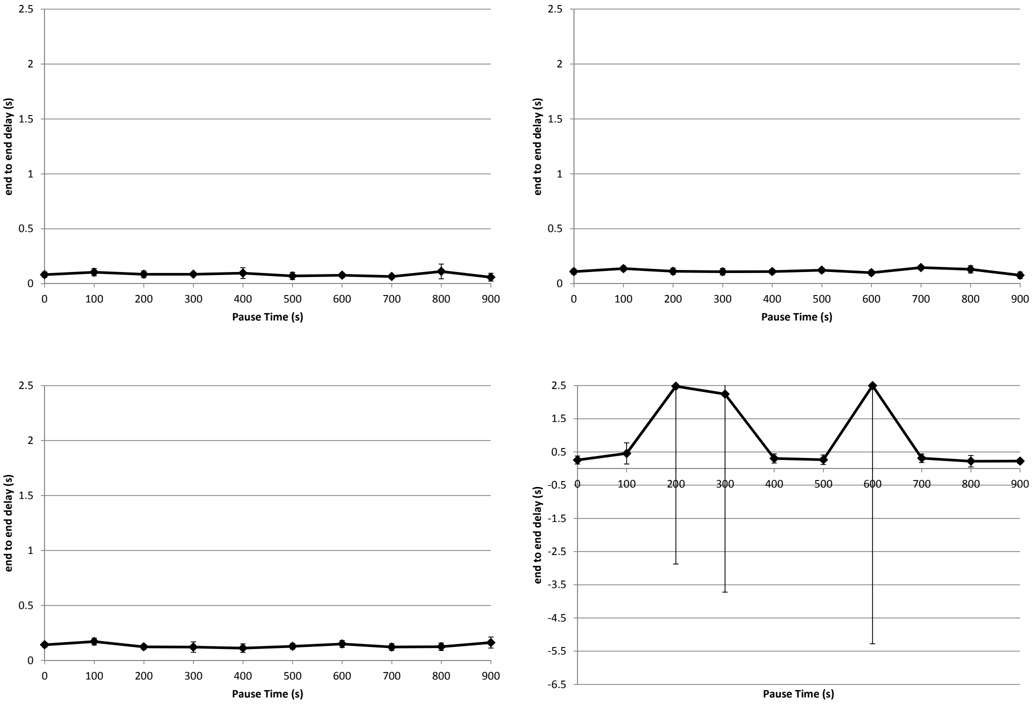


Figure F.2: Average end-to-end delay achieved in Validation I scenario using `RouteAodvMultiInterface` (v2).

simulation runs.

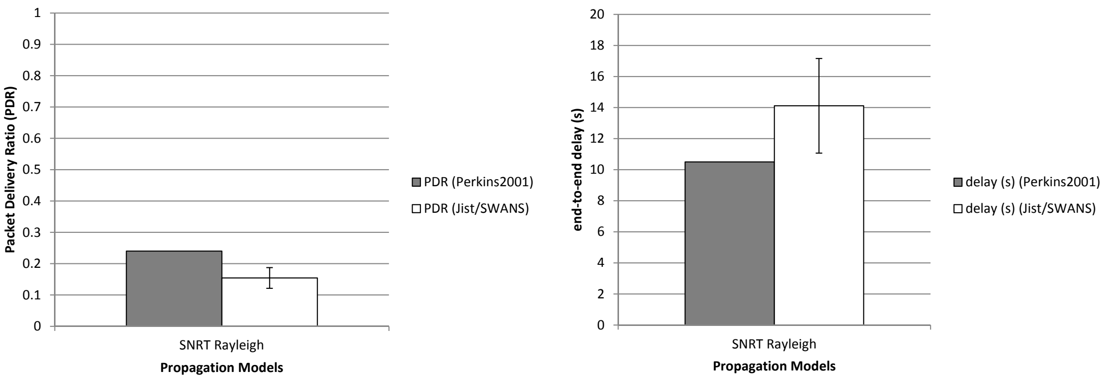


Figure F.3: PDR and end-to-end delay achieved in Validation II using Rayleigh fading and Tworay pathloss within Jist/SWANS and original study for comparison.

APPENDIX G

CONFIGURATION OF ADVERSARIES

The misbehaviour described in Chapter 4 was implemented in the `RouteAodvMultiInterfaceAdversary` router within Jist/SWANS. Adversaries were created with an instance of this router and their behaviour controlled by calling any of the methods shown in Table G.1 at any point in simulation time. An example that causes all adversaries to behave as a black hole at *t=20s* is shown in listing G.1.

Listing G.1: Source code which causes adversary nodes of type `NODE_ADVERSARY` to attack via a black hole at *t=20s*.

```
JistAPI.runAt(new Runnable() {
   public void run() {

      final Vector<Node> adversaries = nodes.getByType(Constants.
         NODE_ADVERSARY);
      for (Node n : adversaries) {
      final RouteAodvMultiInterfaceAdversary advRouter = ((
         RouteAodvMultiInterfaceAdversary)n.getRouter());
                        advRouter.setBlackHole();
      }
   }
},20*Constants.SECOND);
```

Table G.1: Methods on `RouteAodvMultiInterfaceAdversary` to control the adversary during simulation.

Mode of operation	Method	Comments
Normal	`setNormal()`	Resets the adversary

Passive Route Invasion	`setPassiveRouteInvasion( int passiveRouteInvasion)`	Sets the type of passive route invasion to attempt (Table G.3)
Active Route Invasion	`setActiveRouteInvasion( int activeRouteInvasion)`	Sets the type of active route invasion to attempt (Table G.4)
Selfish	`setSelfish()`	Sets the adversary as selfish
Black Hole	`setBlackHole()`	Sets the adversary as a black hole
Gray Hole	`setGrayHole(Vector targets)`	Sets the adversary as a grey hole targeting nodes specified in `targets`
Wormhole	`setWormhole()`	Set the wormhole interface, consequently used to broadcast route requests from

Table G.3: Possible values of `passiveRouteInvasion` field used in `RouteAodvMultiInterfaceAdversary` router.

Attack type	**Value of `passiveRouteInvasion` field**	**Action**
N/A	`NULL`	Do nothing
1	`FAKE_SEQNUM`	For requested destinations that are known to the adversary, advertise route with a fake destination sequence number (known `dsn` + a constant).
2	`FORGE_RREP_AS_DESTINATION`	For any requested destination, forge a route advertisement as the specified destination (`hopCount` = *0*) with false `dsn`.
3	`FORGE_RREP_WITH_ROUTE_TO_DESTINATION`	For any requested destination, forge a route advertisement as if from an intermediate node with a route to the destination, with user defined `hopCount` and false `dsn`.

Table G.4: Possible values of `activeRouteInvasion` field used in `RouteAodvMultiInterfaceAdversary` router.

Attack Type	**Value of `activeRouteInvasion` field**	**Actions**
N/A	`NULL`	Do nothing
1	`FORGE_RREP`	Forge a RREP for the targetted flow
2	`FORGE_RREQ_THEN_RREP`	Send a RREQ to the flow destination in order to retrieve its latest destination sequence number. Then forge a RREP for the targetted flow based on the latest sequence number
3	`FORGE_RERR_SYBIL`	Forge a RERR whose identity (source network address) corresponds to a node on the targetted flow
4	`FORGE_RREP_SYBIL`	Forge a RREP whose identify (source network address) corresponds to a node on the targetted flow

APPENDIX H

TESTING THE LEMKE-HOWSON ALGORITHM IMPLEMENTATION

In order to verify the accuracy of the implementation of LH within Jist/SWANS (`Lemke.java`), the algorithm was applied to general games with well-known results, including the Prisoners Dilemma, coordination and discoordination games. The general forms of these games are specified solely by their inequalities between payoffs. The implementation was also applied to randomly generated general-sum games, with the expected result generated by Gambit McKelvey *et al.* (2010). The results show that the expected Nash Equilibria obtained correlate with the actual Nash Equilibria generated by the LH algorithm. The unit tests forming this validation is distributed with the source, the details of which can be found in Appendix D.

<table>
<tr><th>Type of game</th><th>General Form</th><th>Game Instantiation</th><th>Expected NE</th><th>Actual NE</th></tr>
<tr><td>Prisoners
Dilemma</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>Cooperate</td><td>Defect</td></tr><tr><td rowspan="2">R</td><td>Cooperate</td><td>R,R</td><td>S,T</td></tr><tr><td>Defect</td><td>T,S</td><td>P,P</td></tr></table>$T > R > P > S$</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>Cooperate</td><td>Defect</td></tr><tr><td rowspan="2">R</td><td>Cooperate</td><td>-1,-1</td><td>-10,0</td></tr><tr><td>Defect</td><td>0,-10</td><td>-8,-8</td></tr></table></td><td>[0,1,0,1]
Rasmusen (2001)</td><td>[0,1,0,1]</td></tr>
<tr><td>Coordination</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>a,w</td><td>b,x</td></tr><tr><td>B</td><td>c,y</td><td>d,z</td></tr></table>$a > c, d > b, w > x, z > y$</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>5,4</td><td>2,3</td></tr><tr><td>B</td><td>4,1</td><td>3,2</td></tr></table></td><td>[1,0,1,0]
[0,1,0,1]
[0.5,0.5,0.5,0.5]
Rasmusen (2001)</td><td>[1,0,1,0]
[0,1,0,1]
[0.5,0.5,0.5,0.5]</td></tr>
<tr><td>Dis-coordination</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>a,w</td><td>b,x</td></tr><tr><td>B</td><td>c,y</td><td>d,z</td></tr></table>$a > c, d > b, x > w, y > z$</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>1,-1</td><td>-1,1</td></tr><tr><td>B</td><td>-1,1</td><td>1,-1</td></tr></table></td><td>[0.5,0.5,0.5,0.5]
Rasmusen (2001)</td><td>[0.5,0.5,0.5,0.5]</td></tr>
<tr><td>Dis-coordination</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>a,w</td><td>b,x</td></tr><tr><td>B</td><td>c,y</td><td>d,z</td></tr></table>$c > a, b > d, w > x, z > y$</td><td><table><tr><td colspan="2" rowspan="2"></td><td colspan="2">C</td></tr><tr><td>A</td><td>B</td></tr><tr><td rowspan="2">R</td><td>A</td><td>-1,1</td><td>1,-1</td></tr><tr><td>B</td><td>1,-1</td><td>-1,1</td></tr></table></td><td>[0.5,0.5,0.5,0.5]
Rasmusen (2001)</td><td>[0.5,0.5,0.5,0.5]</td></tr>
</table>

<table>
<tr><th>Type of game</th><th>General Form</th><th>Game Instantiation</th><th>Expected NE</th><th>Actual NE</th></tr>
<tr>
<td>Random General Sum</td>
<td>N/A</td>
<td><table>
<tr><td></td><td></td><td colspan="3"><i>C</i></td></tr>
<tr><td></td><td></td><td>A</td><td>B</td><td>C</td></tr>
<tr><td><i>R</i></td><td>D</td><td>2.740,
1.230</td><td>1.100,
2.197</td><td>2.828,
2.667</td></tr>
<tr><td></td><td>E</td><td>2.925,
2.229</td><td>2.604,
1.895</td><td>1.103,
1.000</td></tr>
<tr><td></td><td>F</td><td>1.914,
2.295</td><td>2.100,
1.150</td><td>2.272,
2.426</td></tr>
</table></td>
<td>[1,0,0,0,0,1]
[0,1,0,1,0,0]
[0.370,0.451,0.180,
0.2310.397,0.371]
McKelvey <i>et al.</i> (2010)</td>
<td>[1,0,0,0,0,1]
[0,1,0,1,0,0]
[0.370,0.451,0.180,
0.231,0.397,0.371]</td>
</tr>
</table>

APPENDIX I

PUBLICATIONS

Position Paper

Kiddie, P. & Constantinou C. (2008) Decentralised Soft-Security in Distributed Systems. Presented at PhD-NOW workshop, co-located with Ad-Hoc NOW 2008, Sophia Antipolis, France.

REFERENCES

Aad, I. & Hubaux, J.-P. (2004) ns-2 for the impatient. Technical report, EPFL, Lausanne, Switzerland.

Abolhasan, M., Wysocki, T. & Dutkiewicz, E. (2004) A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2 (1), 1 – 22. ISSN 1570-8705.

Abraham, A., Jain, R., Thomas, J. & Han, S.-Y. (2007) D-SCIDS: Distributed soft computing intrusion detection system. *Journal of Network and Computer Applications*, 30 (1), 81 – 98. ISSN 1084-8045.

Acs, G., Buttyan, L. & Vajda, I. (2004) Provably Secure On-demand Source Routing in Mobile Ad Hoc Networks. *Mobile Computing, IEEE Transactions on*, 5 (11), 1533 – 1546. ISSN 1536-1233.

Aickelin, U., Bentley, P., Cayzer, S., Kim, J. & McLeod, J. (2003) Danger Theory: The Link between AIS and IDS? *ICARIS-2003, 2nd International Conference on Artificial Immune Systems*, volume 2787. 147 – 155.

Aickelin, U. & Cayzer, S. (2002) The danger theory and its application to artificial immune systems. Technical Report HPL-2002-244, HP Labs, Bristol.

Alpcan, T. & Basar, T. (2003) A game theoretic approach to decision and analysis in network intrusion detection. *42nd IEEE International Conference on Decision and Control*, volume 3. Maui, HI, USA. ISBN 0 7803 7924 1, 2595 – 2600.

Apache Commons (2010) Math - Commons-Math: The Apache Commons Mathematics Library. `http://commons.apache.org/math/`. Last Accessed: 25/08/2010.

Apache Jakarta Project (2010) BCEL: The Byte Code Engineering Library. `http://jakarta.apache.org/bcel/`. Last Accessed: 25/08/2010.

Apache Software Foundation (2010) Apache log4j 1.2 - log4j 1.2. `http://logging.apache.org/log4j/1.2/`. Last Accessed: 15/09/2010.

Bace, R. G. (2000) *Intrusion detection.* MacMillan Technology.

Balachandran, S., Dasgupta, D. & Wang, L. (2006) A Hybrid Approach for Misbehavior Detection in Wireless Ad-Hoc Networks.

Balakrishnan, K., Deng, J. & Varshney, V. (2005) TWOACK: preventing selfishness in mobile ad hoc networks. *Wireless Communications and Networking Conference*, 4, 2137 – 2142. ISSN 1525-3511.

Baras, J. S., Radosavac, S., Theodorakopoulos, G., Sterne, D., Budulas, P. & Gopaul, R. (2007) Intrusion Detection System Resiliency to Byzantine Attacks: The Case Study of Wormholes in OLSR. *Milcom 2007: Proceedings of the 2007 Military Communications Conference.*

Barr, R. (2004a) *JiST - Java In Simulation Time.* Cornell.

Barr, R. (2004b) *SWANS - Scalable Wireless Ad Hoc Network Simulator User Guide.* Cornell.

Barr, R., Haas, Z. J. & van Renesse, R. (2005) JiST: an efficient approach to simulation using virtual machines: Research Articles. *Softw. Pract. Exper.*, 35 (6), 539 – 576. ISSN 0038-0644.

Basile, C., Kalbarczyk, Z. & Iyer, R. (2005) Neutralization of errors and attacks in wireless ad hoc networks. *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on.* 518 – 527.

Billauer, E. (2011) peakdet: Peak detection using MATLAB. `http://www.billauer.co.il/peakdet.html`. Last Accessed: 17/01/2011.

Bokareva, T., Bulusu, N. & Jha, S. (2005) SASHA: Toward a Self-Healing Hybrid Sensor Network Architecture. *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on.* 71 – 78.

Brandes, U. (2001) A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25, 163 – 177.

Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C. & Jetcheva, J. (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking.* ACM, New York, NY, USA. ISBN 1-58113-035-X, 85 – 97.

Buchegger, S. & Le Boudec, J. (2002a) Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on*, 403 – 410.

Buchegger, S. & Le Boudec, J.-Y. (2002b) Performance analysis of the CONFIDANT protocol. *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM, New York, NY, USA. ISBN 1-58113-501-7, 226 – 236.

Buchegger, S. & Le Boudec, J.-Y. (2005) Self Policing Mobile Ad-Hoc Networks by Reputation. *IEEE Communication Magazine*, 43 (7), 101 – 107.

Buttyan, L. & Hubaux, J.-P. (2000) Toward a formal model of fair exchange – a game theoretic approach. Technical report.

Buttyan, L. & Hubaux, J.-P. (2003) Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 8 (5).

Buttyan, L. & Hubaux, J.-P. (2007) *Security and Cooperation in Wireless Networks Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. Cambridge University Press.

Caro, G. A., Ducatelle, F. & Gambardella, L. M. (2008) A Simulation Study of Routing Performance in Realistic Urban Scenarios for MANETs. *ANTS '08: Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, volume 5217/2008. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-87526-0, 211 – 218.

Carruthers, R. & Nikolaidis, I. (2005) Certain limitations of reputation–based schemes in mobile environments. *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM, New York, NY, USA. ISBN 1-59593-188-0, 2 – 11.

Chakeres, I. & Belding-Royer, E. M. (2005) AODV Implementation Design and Performance Evaluation. *International Journal of Wireless and Mobile Computing (IJWMC)*, 2/3.

Chapel, H., Haeney, M., Misbah, S. & Snowden, N. (2006) *Essentials of Clinical Immunology*. Wiley-Blackwell.

Dasgupta, D. & Gonzalez, F. (2002) An immunity-based technique to characterize intrusions in computernetworks. *Evolutionary Computation, IEEE Transactions on*, 6 (3), 281 – 291. ISSN 1089-778X.

Deng, H., Li, W. & Agrawal, D. (2002) Routing security in wireless ad hoc networks. *Communications Magazine, IEEE*, 40 (10), 70 – 75. ISSN 0163-6804.

Dhillon, D., Randhawa, T., Wang, M. & Lamont, L. (2004) Implementing a fully distributed certificate authority in an OLSR MANET. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, 2, 682 – 688. ISSN 1525-3511.

Dijkstra, E. W. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269 – 271. ISSN 0029-599X.

Doan, T., Melvold, R., Viselli, S. & Waltenbaugh, C. (2007) *Lippincott's Illustrated Reviews: Immunology (Lippincott's Illustrated Reviews Series)*. Lippincott Williams & Wilkins.

Douceur, J. R. (2002) The Sybil Attack. *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, volume 2429/2002. Springer-Verlag, London, UK. ISBN 3-540-44179-4, 251 – 260.

Feigenbaum, J. & Shenker, S. (2002) Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. *In Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. ACM Press, 1 – 13.

Felegyhazi, M., Buttyan, L. & Hubaux, J.-P. (2006) Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing (TMC)*, 5 (5). ISSN 1536 – 1233.

Felegyhazi, M., Hubaux, J.-P. & Buttyan, L. (2003) Equilibrium Analysis of Packet Forwarding Strategies in Wireless Ad Hoc Networks - the Dynamic Case. Technical report.

Fong, M. (2004) JiST event viewer project report. Technical report, Cornell University.

Forrest, S. & Hofmeyr, S. (2001) Engineering an Immune System.

Ghica, O. C., Trajcevski, G., Scheuermann, P., Bischof, Z. & Valtchanov, N. (2008) SIDnet-SWANS: a simulator and integrated development platform for sensor networks applications. *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, New York, NY, USA. ISBN 978-1-59593-990-6, 385 – 386.

Gonzalez, F. A. & Dasgupta, D. (2004) Anomaly Detection Using Real-Valued Negative Selection. *Journal of Genetic Programming and Evolvable Machines*. 4 – 383.

Greensmith, J., Aickelin, U. & Twycross, J. (2006) *Articulation and Clarification of the Dendritic Cell Algorithm*.

Haas, Z. J., Halpern, J. Y. & Li, L. (2006) Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14, 479–491. ISSN 1063-6692.

Hamill, P. (2004) *Unit Test Frameworks.* O'Reilly Media.

Hao, L., Li, X., Yang, S. & Lu, S. (2006) Fast Authentication Public Key Infrastructure for Mobile Ad Hoc Networks Based on Trusted Computing. *Wireless Communications, Networking and Mobile Computing, 2006. WiCOM 2006.International Conference on.* 1 – 4.

Heberlein, L., Dias, G., Levitt, K., Mukherjee, B., Wood, J. & Wolber, D. (1990) A network security monitor. *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on.* 296 – 304.

Helsinki, V. K. & Kärpijoki, V. (2000) Security in Ad Hoc Networks.

Hu, Y.-C., Johnson, D. B. & Perrig, A. (2002) SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, 3 – 13.

Hu, Y.-C., Perrig, A. & Johnson, D. (2003a) Packet leashes: a defense against wormhole attacks in wireless networks. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 3, 1976 – 1986. ISSN 0743-166X.

Hu, Y.-C., Perrig, A. & Johnson, D. B. (2003b) Rushing attacks and defense in wireless ad hoc network routing protocols. *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security.* ACM, New York, NY, USA. ISBN 1-58113-769-9, 30 – 40.

Hu, Y.-C., Perrig, A. & Johnson, D. B. (2005) Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11 (1 - 2), 21 – 38.

Huang, Y., Fan, W., Lee, W. & Yu, P. S. (2003) Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies. *Distributed Computing Systems, International Conference on*, 0, 478. ISSN 1063-6927.

Huang, Y.-a. & Lee, W. (2003) A cooperative intrusion detection system for ad hoc networks. *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks.* ACM, New York, NY, USA. ISBN 1-58113-783-4, 135 – 147.

Janeway, C., Travers, P., Walport, M. & Shlomchik, M. (2004) *Immunobiology.* Garland Science.

Jaramillo, J. J. & Srikant, R. (2007) DARWIN: distributed and adaptive reputation mechanism for wireless ad-hoc networks. *MOBICOM.* 87 – 98.

Johnson, D. B. & Maltz, D. A. (1996) Dynamic Source Routing in Ad Hoc Wireless Networks. Imielinski & Korth, eds., *Mobile Computing*, volume 353. Kluwer Academic Publishers.

Jubin, J. & Tornow, J. (1987) The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75 (1), 21 – 32. ISSN 0018-9219.

JUNG (2010) Java Universal Network/Graph Framework. `http://jung.sourceforge.net/doc/index.html`. Last Accessed: 17/10/2010.

Just, M., Kranakis, E. & Wan, T. (2003) Resisting Malicious Packet Dropping in Wireless Ad Hoc Networks. *In Proc. of ADHOCNOW?03*. Springer Verlag, 151 – 163.

Karlof, C. & Wagner, D. (2003) Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1 (2-3), 293 – 315. ISSN 1570-8705.

Kemmerer, R. A. & Vigna, G. (2002) Intrusion Detection: A Brief History and Overview (Supplement to Computer Magazine). *Computer*, 35, 27 – 30. ISSN 0018-9162.

Kim, J. & Bentley, P. (2001) Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2. 1244 – 1252.

Kim, J., Bentley, P., Wallenta, C., Ahmed, M. & Hailes, S. (2006) Danger Is Ubiquitous: Detecting Malicious Activities in Sensor Networks Using the Dendritic Cell Algorithm. *ICARIS*. 390 – 403.

Kliot, G. (2009) Technion Extensions of the JiST/SWANS Simulator.

Kurkowski, S., Camp, T. & Colagrosso, M. (2005a) MANET Simulation Studies: The Current State and New Simulation Tools.

Kurkowski, S., Camp, T. & Colagrosso, M. (2005b) MANET simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9 (4), 50 – 61.

Kyasanur, P. & Vaidya, N. (2003) Detection and handling of MAC layer misbehavior in wireless networks. *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, 173 – 182.

Lemke, C. E. & J. T. Howson, J. (1964) Equilibrium Points of Bimatrix Games. *SIAM Journal on Applied Mathematics*, 12 (2), 413 – 423.

Li, L., Halpern, J. & Haas, Z. (2002) Gossip-based Ad Hoc Routing.

Lin, C. (2004) SWANS AODV project report.

Liu, K., Deng, J., Varshney, P. & Balakrishnan, K. (2007) An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in MANETs. *Mobile Computing, IEEE Transactions on*, 6 (5), 488 – 502. ISSN 1536-1233.

Liu, Y., Comaniciu, C. & Man, H. (2006) A Bayesian game approach for intrusion detection in wireless ad hoc networks. *GameNets '06: Proceeding from the 2006 workshop on Game theory for communications and networks.* ACM, New York, NY, USA. ISBN 1-59593-507-X, 4.

Lundgren, H., Nordstrom, E. & Tschudin, C. (2002) The gray zone problem in IEEE 802.11b based ad hoc networks. *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, volume 6. 104–105.

Luo, J., Hubaux, J.-P. & Eugster, P. T. (2005) DICTATE: DIstributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks. *IEEE Transactions on Dependable and Secure Computing*, 2 (4), 311 – 323.

Marti, S., Giuli, T. J., Lai, K. & Baker, M. (2000) Mitigating routing misbehavior in mobile ad hoc networks. *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM Press, New York, NY, USA. ISBN 1-58113-197-6, 255 – 265.

Matzinger, P. (2002) The Danger Model: A Renewed Sense of Self. *Science*, 296 (5566), 301 – 305.

McKelvey, R. D., McLennan, A. M. & Turocy, T. L. (2010) Gambit: Software Tools for Game Theory, Version 0.2010.09.01.

Michiardi, P. & Molva, R. (2002) CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. *Advanced Communications and Multimedia Security. IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, 107 – 121.

Michiardi, P. & Molva, R. (2005) Analysis of coalition formation and cooperation strategies in mobile ad hoc networks. *Ad Hoc Networks*, 3 (2), 193 – 219. ISSN 1570-8705.

Michiardi, P. & R., M. (2001) Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks.

Mishra, A., Nadkarni, K. & Patcha, A. (2004) Intrusion detection in wireless ad hoc networks. *Wireless Communications, IEEE*, 11 (1), 48 – 60. ISSN 1536-1284.

Nachbar, J. H. (1992) Evolution in the finitely repeated prisoner's dilemma. *Journal of Economic Behavior & Organization*, 19 (3), 307 – 326. ISSN 0167-2681.

Nash, J. F. (1950) Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, 36 (1), 48 – 49. ISSN 00278424.

Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S. & Sheu, J.-P. (1999) The broadcast storm problem in a mobile ad hoc network. *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking.* ACM, New York, NY, USA. ISBN 1-58113-142-9, 151 – 162.

Ning, P. & Sun, K. (2003) How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols. *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society.* 60 – 67.

ns2 (2010) The Network Simulator - ns-2. `http://www.isi.edu/nsnam/ns/`. Last Accessed: 15/03/2010.

OPNET Technologies Inc (2010) Opnet Modeler. `http://www.opnet.com/solutions/network\_rd/modeler.html`. Last Accessed: 1/10/2010.

Otrok, H., Debbabi, M., Assi, C. & Bhattacharya, P. (2007) A Cooperative Approach for Analyzing Intrusions in Mobile Ad hoc Networks. *Distributed Computing Systems Workshops, 2007. ICDCSW '07. 27th International Conference on*, 86 – 86. ISSN 1545-0678.

Otrok, H., Mohammed, N., Wang, L., Debbabi, M. & Bhattacharya, P. (2008a) A game-theoretic intrusion detection model for mobile ad hoc networks. *Computer Communications*, 31 (4), 708 – 721. ISSN 0140-3664.

Otrok, H., Mohammed, N., Wang, L., Debbabi, M. & Bhattacharya, P. (2008b) A Moderate to Robust Game Theoretical Model for Intrusion Detection in MANETs. 608 – 612.

Otto, J. & Choffnes, D. (2007) SWANS++ - Extensions to the Scalable Wireless Ad-hoc Network Simulator.

Patcha, A. & Park, J.-M. (2004) A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop (IEEE Cat. No.04EX879)*, 280 – 284.

Patwardhan, A., Parker, J., Joshi, A., Iorga, M. & Karygiannis, T. (2005) Secure Routing and Intrusion Detection in Ad Hoc Networks. *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on.* 191 – 199.

Paul, K. & Westhoff, D. (2002) Context aware detection of selfish nodes in DSR based ad-hoc networks. *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, volume 4. ISSN 1090-3038, 2424 – 2429.

Perkins, C. (1997) Ad-hoc on-demand distance vector routing.

Perkins, C., Royer, E., Das, S. & Marina, M. (2001) Performance comparison of two on-demand routing protocols for ad hoc networks. *Personal Communications, IEEE*, 8 (1), 16 – 28. ISSN 1070-9916.

Perkins, C. E. & Bhagwat, P. (1994) Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24 (4), 234–244. ISSN 0146-4833.

Pirzada, A. A. & McDonald, C. (2004) Establishing trust in pure ad-hoc networks. *ACSC '04: Proceedings of the 27th Australasian conference on Computer science*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 47 – 54.

Pritchard, D. (2007) The Lemke-Howson Algorithm.

Rasmusen, E. (2001) *Games and Information: An Introduction to Game Theory*. Blackwell Publishing. ISBN 0631210954.

Riedel, I. (2003) Security in Ad-hoc Networks: Protocols and Elliptic Curve Cryptography on an Embedded Platform.

Rivest, R. (1992) The MD5 Message-Digest Algorithm.

Roughgarden, T., Tardos, E. & Vazirani, V. V. (2007) *Algorithmic Game Theory*. Cambridge University Press.

Royer, E. & Toh, C.-K. (1999) A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 6 (2), 46 – 55. ISSN 1070-9916.

Sanzgiri, K., Dahill, B., Levine, B., Shields, C. & Belding-Royer, E. (2002a) A secure routing protocol for ad hoc networks. *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. ISSN 1092-1648, 78 – 87.

Sanzgiri, K., Dahill, B., Levine, B., Shields, C. & Belding-Royer, E. (2002b) A secure routing protocol for ad hoc networks. *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. ISSN 1092-1648, 78 – 87.

Sarafijanovic, S. & Le Boudec, J.-Y. (2005) An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *International Journal of Unconventional Computing*, 1, 221 – 254.

Schmidt, S., Alpcan, T., Albayrak, S., Basar, T. & Mueller, A. (2008) A Malware Detector Placement Game for Intrusion Detection. *Lecture Notes in Computer Science*, volume 5141. Springer Berlin / Heidelberg, 311 – 326.

Schoch, E. (2009) JiST/SWANS Portal.

Schoch, E., Feiri, M., Kargl, F. & Weber, M. (2008) Simulation of ad hoc networks: ns-2 compared to JiST/SWANS. *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops.* ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium. ISBN 978-963-9799-20-2, 1 – 8.

Shapley, L. S. (1953) A Value for n-person Games. *Contributions to the Theory of Games*, II, 307 – 317.

Shi, E. & Perrig, A. (2004) Designing secure sensor networks. *Wireless Communications, IEEE*, 11 (6), 38 – 43. ISSN 1536-1284.

Shoham, Y. & Leyton-Brown, K. (2009) *Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press.

Srinivasan, A., Teitelbaum, J., Wu, J., Cardei, M. & Liang, H. (2008) *Reputation-and-Trust-Based Systems for Ad Hoc Networks*, chapter 13. Algorithms and Protocols for Wireless, Mobile Ad Hoc Networks. Wiley, 375 – 403.

Srinivasan, V., Nuggehalli, P., Chiasserini, C. & Rao, R. (2003) Cooperation in wireless ad hoc networks. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2, 808 – 817. ISSN 0743-166X.

Stajano, F. & Anderson, R. (2000) The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. *Proceedings of the 7th International Workshop on Security Protocols.* Springer-Verlag, London, UK. ISBN 3-540-67381-4, 172–194.

Sundararajan, T. & Shanmugam, A. (2009) Behavior Based Anomaly Detection Technique to Mitigate the Routing Misbehavior in MANET. *International Journal of Computer Science and Security (IJCSS)*, 3.

Takai, M., Martin, J. & Bagrodia, R. (2001) Effects of wireless physical layer modeling in mobile ad hoc networks. *MobiHoc '01: Proceedings of the 2nd ACM international* *symposium on Mobile ad hoc networking & computing.* ACM, New York, NY, USA. ISBN 1-58113-428-2, 87 – 94.

Urpi, A., Bonuccelli, M. & Giordano, S. (2003) Modelling cooperation in mobile ad hoc networks: a formal description of selfishness. *Proc. of WiOpt Workshop, 2003.*

Vahdat, A. & Becker, D. (2000) Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical report.

Vigna, G., Gwalani, S., Srinivasan, K., Belding-Royer, E. & Kemmerer, R. (2004) An intrusion detection tool for AODV-based ad hoc wireless networks. *Computer Security Applications Conference, 2004. 20th Annual.* ISSN 1063-9527, 16 – 27.

Vigna, G., Valeur, F. & Kemmerer, R. A. (2003) Designing and implementing a family of intrusion detection systems. *ESEC/FSE-11: Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering.* ACM, New York, NY, USA. ISBN 1-58113-743-5, 88 – 97.

Von Neumann, J. (1928) *Zur theorie der gesellschaftsspiele.*

Von Neumann, J. & Morgenstern, O. (1970) *Theory of games and economic behavior.* Princeton Princeton Univ. Press.

Wang, F., Vetter, B. & Wu, S. F. (1997) Secure Routing Protocols: Theory and Practice. Technical report, North Carolina State University.

Wang, W., Lu, Y. & Bhargava, B. (2003) On security study of two distance vector routing protocols for mobile ad hoc networks. *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on.* 179 – 186.

Widger, J. & Grosu, D. (2009) Parallel Computation of Nash Equilibria in N-Player Games. volume 1. 209 – 215.

Xu, W., Trappe, W., Zhang, Y. & Wood, T. (2005) The feasibility of launching and detecting jamming attacks in wireless networks. *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing.* ACM, New York, NY, USA. ISBN 1-59593-004-3, 46 – 57.

Yan, Z. (2006) A conceptual architecture of a trusted mobile environment. *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on.* 81.

Yang, H., Luo, H., Ye, F., Lu, S. & Zhang, L. (2004) Security in mobile ad hoc networks: challenges and solutions. *Wireless Communications, IEEE*, 11 (1), 38 – 47. ISSN 1536-1284.

Yi, Y., Gerla, M. & Kwon, T. J. (2003) Efficient flooding in ad hoc networks: a comparative performance study. *Communications, 2003. ICC '03. IEEE International Conference on*, volume 2. 1059 – 1063.

Zhang, Y. & Lee, W. (2000) Intrusion detection in wireless ad-hoc networks. *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM Press, New York, NY, USA. ISBN 1-58113-197-6, 275 – 283.

Zhou, L. & Haas, Z. J. (1999) Securing Ad Hoc Networks. *IEEE Network*, 13 (6), 24 – 30.