An Intrusion Detection Scheme for Wireless Mobile Ad hoc Networks based on DSDV Protocol

by

Ketan Nadkarni

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Dr. Amitabh Mishra, Chairman
Dr. Srinidhi Varadarajan, Co-Chairman
Dr. Eunice Santos, Committee Member

Keywords: Wireless, Ad hoc, Security, Intrusion Detection, Self-Adjusting Threshold

August 29, 2003
Blacksburg, Virginia

An Intrusion Detection Scheme for Wireless Mobile Ad hoc Networks based on DSDV Protocol

Ketan Nadkarni

(ABSTRACT)

Wireless mobile ad-hoc networks (MANETs) have come into prominence due to potentially rapid and infrastructure-less deployment in military operations and also in emergency and disaster-relief situations. However, the unreliability of wireless links between nodes, possibility of mobile nodes being captured or compromised, break down of cooperative algorithms, all lead to increased vulnerability. No matter how supposedly secure a system is, unrelenting attackers eventually succeed in infiltrating it. This underscores the need to monitor what is taking place in a system and look for suspicious behavior. An Intrusion Detection System (IDS) does just that: monitors audit data, looks for intrusions in the system, and initiates a proper response. Bandwidth constraints of MANETs necessitate the need for efficiency of any security scheme in order to prevent the overloading of the network.

In this thesis, we have proposed an effective and efficient IDS for MANETs that aims to combine misuse detection with anomaly detection. Experimental validation has provided significant results about not only the accuracy and robustness of the scheme but also the non-degradability of network performance upon induction of our security scheme. It is not affected by factors such as node density, node mobility, traffic load and percentage of malicious nodes. On an average, our IDS, implemented using Destination-Sequenced Distance-Vector (DSDV) protocol, detects intrusions with an accuracy of over 90% and is generally insensitive to false alarms. Moreover, performance metrics such as end-to-end delay, packet delivery ratio and normalized routing load are only marginally affected (about 2% decrease in performance).

Acknowledgements

I take this opportunity to express my deepest gratitude to my parents back home in Goa, India for their moral support through the course of my studies. Without them, this would not have been possible for they have positively influenced my life and have shaped the person that I am today.

This research would not have been possible without the support of many people. I would like to thank my thesis advisor Dr. Amitabh Mishra for guiding me through my thesis. Dr. Mishra spent several hours reviewing this document for accuracy and for providing critical comments about good writing during the several iterations that this document went through. I thank him for his constructive comments on the thesis; it has helped me improve my writing skills and present my research in a much better way.

A special word of thanks goes to Dr. Srinidhi Varadarajan for agreeing to be the co-chairman of my thesis committee. I would also like to thank Dr. Eunice Santos for being on my committee and giving invaluable suggestions on my work.

Finally, I would like to thank my friends for all the good times we have had together.

Table of Contents

List of Figures

List of Tables

1. Introduction

Despite considerable advancement in wireless networks over the last few years, the realm of wireless mobile ad-hoc networks, in particular, is a relatively new one. Essentially, these are networks without any underlying infrastructure. The tremendous potential for rapid deployment of such networks is offset by their inherently vulnerable characteristics. Security in ad hoc networks, thus, assumes paramount importance. This chapter provides an insight into wireless mobile ad hoc networks and forms the basis of our problem statement.

1.1 Ad hoc Networks

An ad-hoc network, by definition, is a collection of autonomous nodes that form a dynamic, purpose-specific, multi-hop radio network in a decentralized fashion. Centralized network machinery seen in traditional wireless networks such as mobile switching centers, base stations, access points is conspicuous by its absence in wireless mobile ad hoc networks. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multihop topologies which are most likely composed of relatively bandwidth-constrained wireless links.

1.2 Applications

From industrial to tactical usage, wireless mobile ad hoc networks have stimulated a lot of interest in the research community. The set of applications is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks. Some applications of MANET technology could include industrial and commercial applications involving cooperative mobile data exchange. The developing technologies of "wearable" computing and communications may provide applications for MANET technology [RFC 2501]. When properly combined with satellite-based information delivery, MANET technology can provide an extremely flexible method for establishing communications for fire/safety/rescue operations or other scenarios requiring rapidly-deployable communications with survivable, efficient dynamic networking.

Wireless mobile ad hoc networks are extremely useful in situations where there is a need to rapidly deploy a network for communication such as emergency situations. These networks also find application in certain sensitive situations where one cannot have a fixed infrastructure such as military battlefield. Applications encompass various areas including home-area wireless networking, personal area networking, on-the-fly conferencing, collaborative peer-to-peer computing, wireless sensor networks, disaster recovery, search-and-rescue operations, battlefield communication and GSM service extension to dead spots. These networks are now being envisioned to support more and more applications than just forming a communication medium.

1.3 Characteristics

Wireless mobile ad hoc networks do not have an underlying fixed infrastructure. Some of the salient features of such networks are summarized below:

1. An ad hoc network can be created or deployed at the spur of the moment.
2. Mobile hosts can “join” in and move out of the network at any time.
3. Mobile nodes within one another’s radio range can communicate directly via radio links whereas nodes not in direct range use other mobile nodes as relays (this is multi-hop communication).
4. The network topology is constantly changing as a result of nodes joining in and moving out.
5. Owing to the lack of an underlying infrastructure with dedicated routers and gateways, the individual mobile nodes carry out the packet forwarding, routing and other network operations themselves.
6. Nodes are battery and bandwidth constrained, which makes energy saving a critical issue.

1.4 Motivation

The very set of characteristics that make these networks unique and rapidly deployable in certain situations can also lead to their downfall. This can be particularly damaging in sensitive scenarios such as a battlefield. The unreliability of wireless links between nodes can lead to passive eavesdropping or even active interfering. Mobile nodes, being independently capable of roaming, can be captured or compromised. Without the support of a fixed underlying infrastructure, cooperative algorithms are critical for the correct functioning of such networks. It is entirely possible for attackers with malicious intent to break the cooperative algorithms upon taking over a compromised node. Moreover, one cannot make any *a priori* assumptions about the trust relationships among the nodes. In effect, there is an inherent lack of security leading to an increased vulnerability and susceptibility to attacks.

In general, no matter how supposedly secure a system is, unrelenting attackers will eventually manage to find a loophole and succeed in infiltrating the system. This underscores the need to constantly (or at least periodically) monitor a system for suspicious behavior. An Intrusion Detection System (IDS) does precisely that: monitors audit data, looks for intrusions in the system and initiates a proper response.

When one thinks of network security, the first thing that comes to mind is preventing unauthorized access to the system and this is accomplished by intrusion prevention measures. Intrusion prevention is not guaranteed to work all the time and, in order to build a high-survivability network, it is necessary to complement traditional intrusion prevention mechanisms such as encryption and authentication with efficient intrusion detection and response techniques. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data is compromised. Moreover, an effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions. Intrusion detection enables the collection of information about intrusions that can be used to strengthen the intrusion prevention facility. Intrusion detection is really the second wall of defense. This research has focused on developing an intrusion detection scheme for wireless mobile ad hoc networks.

1.5 Thesis Contribution

Security features already in place for traditional wireless networks such as IEEE 802.11 WLANs cannot be easily applied to their ad hoc counterparts primarily due to the lack of a similar underlying fixed infrastructure. There is, therefore, a need to develop effective security mechanisms for ad hoc networks. As mentioned earlier, unrelenting attackers eventually manage to find a loophole in the intrusion prevention setup and this can lead to a damaging effect. Intrusion detection is needed not just as a damage-control operation but also for acting as a deterrent for potential attackers.

The following research goals were identified for the proposed intrusion detection scheme and are accomplished in this thesis:

- Identify the needs of an intrusion detection scheme for ad hoc networks.
- Understand the constraints, complexities and design issues.
- Analyze various network simulation tools for suitability.
- Study the routing protocol extensively in order to implement the proposed approach.
- Design an intrusion detection scheme that effectively detects deterministic attacks.
- Analyze the possible and critical attacks (which can have highest severity of negative impact).
- Inject attacks into the system.
- Validate the results experimentally by gathering simulation data.
- Analyze the accuracy, efficiency and robustness of the model.

1.6 Thesis Organization

The rest of the thesis is organized as follows:

- **Chapter 2** discusses wireless mobile ad hoc networks in detail and addresses some of their inherently vulnerable characteristics, which bring out the need for intrusion detection.

- **Chapter 3** provides a brief background on intrusion detection and addresses some of the complexities and design issues. This chapter also touches upon related work in the field of intrusion detection in ad hoc networks.

- **Chapter 4** elaborates on the proposed research and presents our intrusion detection scheme in detail.

- **Chapter 5** presents the experimental model and discusses the simulation methodology.

- **Chapter 6** validates the simulation results with performance evaluation and discusses our findings on the accuracy and robustness of our intrusion detection scheme.

- **Chapter 7** summarizes the research work and presents a few directions for future work.

1.7 Chapter Summary

In this chapter, we have provided an insight into the research work embarked upon. We have also touched upon some of the core research issues, which are discussed in more detail in the following chapters. In the next chapter, we discuss wireless mobile ad hoc networks in detail, including characteristics, routing protocols, and vulnerabilities.

2. Wireless Mobile Ad hoc Networks

This chapter provides an in-depth insight into the realm of wireless mobile ad hoc networks. Popularly known in research circles as MANETs (short for **M**obile **A**d hoc **Net**works), these networks have recently come into the limelight primarily because of tremendous potential for rapid on-the-fly deployment. The Internet Engineering Task Force (IETF) even has a special working group [ietf], which focuses on exploring a broad range of MANET problems, performance issues, and related candidate protocols.

2.1 Introduction

In contrast to infrastructure based wireless networks, all nodes in ad hoc networks are mobile and can be connected dynamically in an arbitrary manner. This may be done either because it may not be economically practical or physically possible to provide the necessary infrastructure or because the situation does not permit its installation. Some classic examples would be situations where friends or business associates would run into each other in an airport terminal and wish to exchange business cards, or in case of an emergency, a group of rescue workers may need to quickly set up a communication medium.

A wireless mobile ad hoc network is, by definition, a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. Limitations of the wireless environment such as transmission range necessitate that mobile nodes make use of other nodes in forwarding a packet to its destination. Thus, multi-hop radio communication is used to transfer data between nodes in the network and this is accomplished by a routing protocol that discovers routes between these nodes.

Nodes usually share the same physical media; they transmit and acquire signals at the same frequency band, and follow the same hopping sequence or spreading code [ZhLe00]. The data link layer functions manage the wireless link resources and coordinate medium access among neighboring nodes. The medium access control (MAC) protocol is essential to a wireless ad-hoc network because it allows mobile nodes to share a common broadcast channel efficiently. The network-layer functions maintain the multi-hop communication paths across the network; all nodes must function as routers that discover and maintain routes to other nodes in the network. Mobility and volatility are hidden from the applications so that any node can communicate with any other node as if everyone were in a fixed wired network.

In the case where only two hosts within the transmission range are involved in the ad hoc network, no real routing protocol or routing decisions are necessary. But, in many practical ad hoc networks, two hosts that wish to correspond may not be close enough to be within wireless transmission range of each other. These hosts could communicate if other hosts between them, which are also participating in the ad hoc network, are willing to forward packets for them.

As an example, consider the Figure 2.1 shown below. Mobile hosts A and *C* are not within the each other's radio range [Jo94]. Now, if *A* and *C* wish to communicate with each other, they may do so by utilizing host *B* to forward packets for them (host *B* lies within the transmission range of both *A* and *C*). A real ad hoc network is more complicated than this example due to the possibility that any or all of the hosts involved may move at any time in any direction.

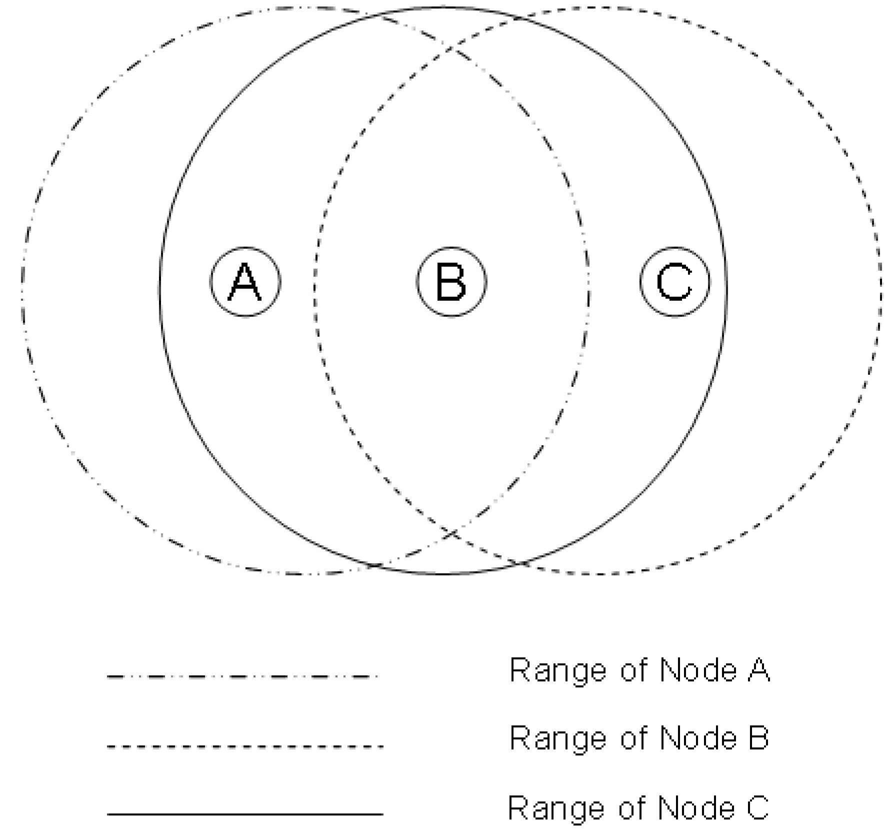


Figure 2.1: Nodes A and C are using Node B as a relay in multi-hop communication [JoMa96]

Using graph theoretic notations, we can view an ad hoc network as a graph, G(A, E(t)), where A denotes a set of nodes representing mobile hosts and E(t) denotes a set of time-varying edges. Thus, each mobile host can be denoted by a node and an edge can be drawn between the two nodes if they are within the wireless communication range of each other. The set of edges (E(t)), is denoted as a function of time, as it keeps changing as the node in the ad hoc networks move around. The topology can be arbitrary since there are no restrictions on where the nodes can be located and how they move with respect to each other.

In a practical ad hoc network, all nodes behave as routers and take part in discovery and maintenance of routes to other nodes in the network. Route construction should be done with a minimum of overhead and bandwidth consumption. Many different protocols have been proposed to solve the multi-hop routing problem in such networks.

2.2 Existing Ad Hoc Routing Protocols

Numerous protocols have been developed for ad hoc networks to deal with the typical limitations of these networks, which include high power consumption, low bandwidth, and high error rates [RoTo99]. As shown in Figure 2.2 below, these unicast routing protocols may generally be categorized as:

- Table-driven
- Source-initiated (demand-driven)

Solid lines in this figure represent direct descendants, while dotted lines depict logical descendants. Despite being designed for the same type of underlying network, the characteristics of each of these protocols are quite distinct.

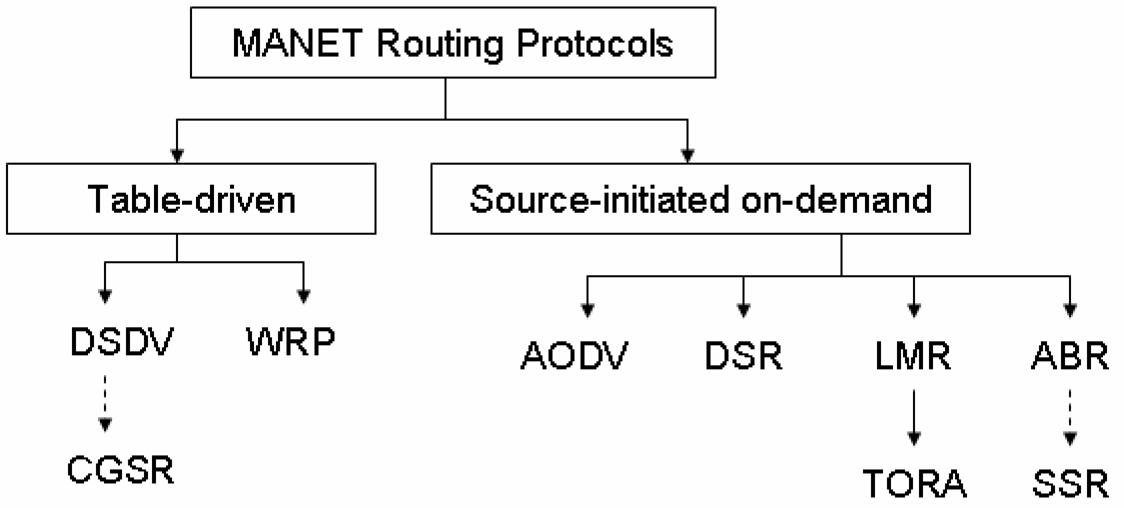


Figure 2.2: Categorization of MANET Routing Protocols [RoTo99].

2.2.1 Table-Driven Routing Protocols

Table-driven routing protocols attempt to maintain consistent, up-to-date routing information from each node to every other node in the network. These protocols require each node to maintain one or more tables to store routing information, and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. The areas in which they differ are the number of necessary routing-related tables and the methods by which changes in network structure are broadcast. Examples of table-driven routing protocols include Destination-Sequenced Distance-Vector (DSDV) protocol and Wireless Routing Protocol (WRP) [ChGe98]. Clusterhead Gateway Switch Routing (CGSR) protocol [MuGa96] is direct descendant of DSDV protocol.

2.2.2 Source-Initiated On-Demand Routing

A different approach from table-driven routing is source-initiated on-demand routing. This type of routing creates routes only when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Examples of source-initiated on-demand routing protocols include Ad-hoc On-Demand Distance Vector (AODV) routing protocol [PeRoDa02], Dynamic Source Routing (DSR) protocol [JoMaHuJe01], Lightweight Mobile Routing (LMR) protocol [CoEp95] and Associativity-Based Routing (ABR) protocol [Toh97]. Signal Stability Routing (SSR) protocol [DuRaWaTr97] makes use of signal stability based adaptive routing. Temporally-Ordered Routing Algorithm (TORA) [PaCo] is a direct descendant of LMR protocol.

2.3 Vulnerabilities of Wireless Ad-Hoc Networks

A wireless ad-hoc network is particularly vulnerable, due to its fundamental characteristics such as open medium, dynamic topology, distributed cooperation, and constrained capability. Ironically, most of the features contribute to usefulness and popularity of ad-hoc networks. In this section, we discuss the characteristics, which make these networks vulnerable to attacks.

The wireless links between nodes are highly susceptible to link attacks, which include passive eavesdropping, active interfering, leakage of secret information, data tampering, impersonation, message replay, message distortion and denial-of-service [MiNa02]. Eavesdropping might give an adversary access to secret information, violating confidentiality [ZhHa99]. Active attacks might allow the adversary to delete messages, to inject erroneous messages, to modify messages, and to impersonate a node, thus violating availability, integrity, authentication, and non-repudiation. All these mean that a wireless ad-hoc network does not have a clear line of defense and every node must be prepared for encounters with an adversary directly or indirectly [ZhLe00].

Unlike wired networks where an adversary must gain physical access to the network wires or pass through several lines of defense at firewalls and gateways, attacks on a wireless ad-hoc network can come from all directions and target any node [ZhLe00]. Mobile nodes are autonomous units that are capable of roaming independently. This means that nodes with inadequate physical protection are susceptible to being captured, compromised, and hijacked.

The absence of infrastructure and subsequently, the absence of centralized authorization facilities impede the usual practice of establishing a line of defense, distinguishing nodes as trusted and non-trusted. There may be no ground for an *a priori* classification, since all nodes are required to cooperate in supporting the network operation, while no prior security association (SA) can be assumed for all the network nodes. Freely roaming nodes form transient associations with their neighbors; join and leave sub-domains independently and without notice.

Decision-making in ad-hoc networks being usually decentralized, many ad-hoc network algorithms rely on the cooperative participation of all nodes. The lack of centralized authority means that the adversaries can exploit this vulnerability for new types of attacks designed to break the cooperative algorithms. Ad-hoc routing presents such vulnerability. Most ad-hoc routing protocols are inherently cooperative. Unlike a wired network where extra protection can be placed on routers and gateways, an adversary who hijacks an ad-hoc node could paralyze the entire wireless network by disseminating false routing information. Such false routing information could also result in messages from all nodes being fed to the compromised node, which is more hidden yet usually more dangerous.

An additional problem related to the compromised nodes is the potential *byzantine failures* encountered within MANET routing protocols, wherein a set of nodes could be compromised in such a way that the incorrect and malicious behavior cannot be directly noted at all. Such malicious nodes can also create new routing messages and advertise non-existent links, provide incorrect link state information and flood other nodes with routing traffic, thus inflicting byzantine failures to the system.

The presence of even a small number of adversarial nodes could result in repeatedly compromised routes, and, as a result, the network nodes would have to rely on cycles of timeout and new route discoveries for communication. This would incur arbitrary delays before the establishment of a non-corrupted path, while successive broadcasts of route requests would impose excessive transmission overhead. In particular, intentionally falsified routing messages would result in a denial-of-service (DoS) experienced by the end nodes.

For nodes relying on battery for power supply, energy conservation is also an important issue, which makes some computation-intensive security measures infeasible. The battery-powered operation of ad-hoc networks gives attackers ample opportunity to launch a denial-of-service attack by creating additional transmissions or expensive computations to be carried out by a node in an attempt to exhaust its batteries.

The above discussion makes it clear that ad-hoc networks are inherently insecure, more so than their traditional wireline counterparts and, need security schemes before it is too late. If there are attacks on a system, one would like to detect them as soon as possible (ideally in real-time) and take appropriate action. This is essentially what an IDS does.

2.4 Need for Intrusion Detection

Although techniques designed for protecting wired networks are in place for years, the vast difference between the two networks makes it very difficult to apply them to an ad-hoc wireless network directly. Intrusion prevention measures, such as encryption and authentication, can be used in ad-hoc networks to reduce intrusions, but cannot eliminate them. For example, encryption and authentication cannot defend against compromised mobile nodes, which carry the private keys. Integrity validation using redundant information (from different nodes), such as those being used in secure routing [SmMuGa97, ZhHa99], also relies on the trustworthiness of other nodes, which could likewise be a weak link for sophisticated attacks. Therefore, we need a second wall of defense that provides the capability to detect intrusions and to alert users. Intrusion detection techniques can provide a second level of defense for securing wireless ad-hoc networks.

Where exactly does an intrusion detection system fit in the design? To put it in simpler terms, an IDS can be compared with a burglar alarm. For example, the lock system in a car protects the car from theft. But if somebody breaks the lock system and tries to steal the car, it is the burglar alarm that detects that the lock has been broken and alerts the owner by raising an alarm.

Thus, an IDS is really a security scheme that monitors a system (or network traffic) and complements intrusion prevention techniques to provide a secure high-survivability system. It analyzes the network traffic for possible hostile attacks originating from outside the system and also for system misuse or attacks originating from inside the system.

2.5 Chapter Summary

Whether the motivation is financial gain, intellectual challenge, espionage, political, or simply trouble-making, one is exposed to a variety of intruder threats. Obviously it is just common sense to guard against these, but the potential for loss of supposedly secure data in mission-critical operations is unfathomable and makes this imperative. Intrusion Detection Systems are like a burglar alarm for your computer network; they detect unauthorized access attempts. With hackers becoming smarter by the day, the safest of intrusion prevention techniques are at significant risk and there is a strong need to provide a secure architecture for a high-survivability system and by this, we mean, a solid Intrusion Detection model. And this is especially important in case of wireless mobile ad hoc networks, which are inherently insecure. The goal of this research is to develop an intrusion detection model. The technical challenge is to overcome the chaotic appearance in operations of wireless ad-hoc networks and discover intrinsic signs of intrusions. The next chapter gives the reader an insight into the components, requirements and types of intrusion detection systems.

3. Background on Intrusion Detection

This chapter provides a background on intrusion detection and addresses some of the complexities and design issues involved in wireless ad hoc environments. Intrusion detection, by definition, is the automated detection and generation of alarms for any situation where an intrusion has taken, or is about to take place.

3.1 Components of an intrusion detection system

Based on the type of audit data used, intrusion detection systems (IDSs) can be categorized as network-based or host-based. A network-based IDS normally runs at the gateway of a network and "captures" and examines network packets that go through the network hardware interface. A host-based IDS relies on operating system audit data to monitor and analyze the events generated by programs or users on the host.

A basic model of an intrusion detection system is likely to include two elements. First element is the audit data that needs to be analyzed in order to determine if any intrusion is taking place. Second element is the detection algorithm that does the actual analysis of audit data and determines if an intrusion is taking place [ZhLe00].

3.2 Taxonomy

Intrusion detection, at a high level, can be classified into three broad categories: Anomaly Detection, Misuse Detection, and Compound Detection [Ax00].

3.2.1 Anomaly Detection

In anomaly detection, the focus is on abnormalities in traffic rather than any specific intrusion signal. The elementary way of looking at this would be to consider anything that is abnormal as suspicious. However, this begs the question as to what can be considered as normal and then deciding what is abnormal based on that. This detection principle thus flags behavior that is unlikely to originate from the normal process, without regard to actual intrusion scenarios.

3.2.2 Misuse Detection

In misuse detection, decisions are made on the basis of knowledge of a model of the intrusive process and what traces it ought to leave in the observed system. *Legal* or *illegal* behavior can be defined, and observed behavior can be compared accordingly. Such a system tries to detect evidence of intrusive activity (also called a signature of an attack) irrespective of any idea of what the background traffic, i.e. normal behavior, of the system looks like. Moreover, such a system has the ability to operate no matter what constitutes the normal behavior of the system.

3.2.3 Compound Detection

This form of detection considers both the normal behavior of the system and the intrusive behavior of the intruder. It operates by detecting the intrusion against the background of the normal traffic in the system. This approach has a much better chance of correctly detecting truly interesting events in the supervised system, since it knows the patterns of intrusive behavior and can relate them to the normal behavior of the system. Compound detection would, at the very least, be able to qualify its decisions better, i.e. give us an improved indication of the quality of the alarm.

3.3 Attributes of an IDS

In this section, we discuss the attributes or elements or constituent parts of a typical IDS. More often than not, an IDS is likely to have at least some of these attributes [Ax00]:

- **Audit collection:** Audit data from which to make intrusion detection decisions must be collected. Source for audit data collection could be keyboard input, command based logs, application based logs etc. However, typically, network activity, or host based security logs (or both) are used.

- **Audit storage:** Typically, the audit data is stored somewhere, either indefinitely for later reference, or temporarily awaiting processing. The volume of data often being exceedingly large, audit storage is a crucial element in any IDS, and this has led some researchers in the field to view intrusion detection as a problem in audit data reduction.

- **Processing:** The processing block is the heart of the intrusion detection system. It is here that one or many algorithms are executed to find evidence (with some degree of certainty) of suspicious behavior, in the audit trail.

- **Configuration data:** This is the state that affects the operation of the IDS as such (how and where to collect audit data, how to respond to intrusions, etc.) and is the main means of controlling the IDS. This data can grow very large, and complex for a real world intrusion detection deployment. It is furthermore quite sensitive, since access to this data would give the competent intruder information about which avenues of attack are likely to go undetected.

- **Reference data:** The reference data storage stores information about known intrusion signatures and/or profiles of normal behavior. In the latter case, the processing element updates the profiles (as new knowledge about the observed behavior becomes available) at regular intervals in a batch-oriented fashion. Stored intrusion signatures are most often updated as and when new intrusion signatures become known.

- **Active/Processing data:** The processing element frequently must store intermediate results, e.g. information about partially fulfilled intrusion signatures. The space needed to store these active data can grow quite large.

- **Alarm:** This part of the system handles all output from the system, whether that be an automated response to the suspicious activity, or which is most common, the notification of some site security officer.

3.4 Examples of IDSs for Wired Networks

It is not within the scope of this thesis to survey the vast literature on IDSs for wired networks. Nevertheless, in order to give the reader a basic idea about the research work done on IDSs for wired networks, we present a few examples.

3.4.1 EMERALD

The EMERALD architecture consists of monitors (interoperable analysis and response units) that provide localized protection of key assets throughout an enterprise network [Emerald]. These monitors are deployed locally to the analysis targets, thus reducing possible analysis and response delays arising from the spatially distributed topology of the network. EMERALD also introduces a hierarchically composable analysis scheme wherein local analyses are shared and correlated at higher layers of abstraction.

EMERALD's scheme begins from the network interface layer of individual administrative domains. Each domain has monitors for analyzing the operation of network services and other externally accessible domain components. Each monitor includes an analysis target-specific set of response handlers that it invokes as it detects possible misuse. The distributed analyses of these service-layer monitors is shared with other EMERALD monitors that perform domain-wide correlation. Domain monitors provide a more global perspective to the profiling and modeling of vulnerabilities. EMERALD also implements an enterprise-wide analysis to correlate the activity reports produced across the set of monitored domains. Enterprise-layer monitors focus on network-wide threats. Reports of problems found by one monitor are propagated throughout the network. The monitor architecture is intended to be very small, very fast, and general enough to be deployed at any layer in EMERALD's hierarchical analysis scheme. The initial design of the EMERALD monitor architecture is illustrated in Figure 3.1.

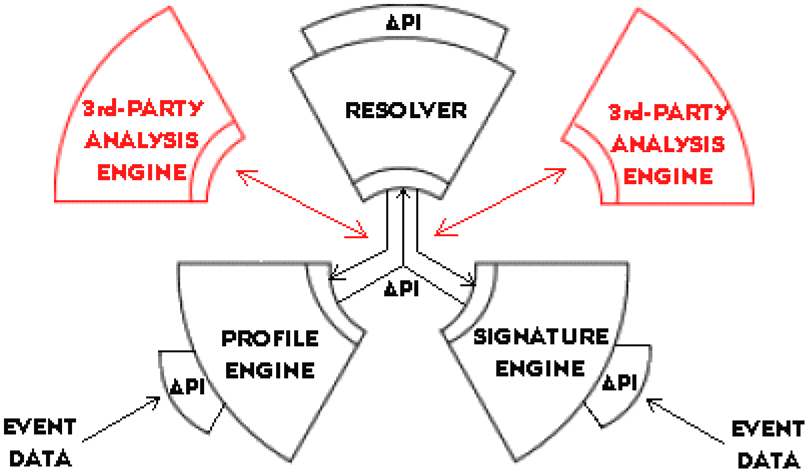


Figure 3.1: EMERALD Monitor Architecture.

EMERALD monitors provide a streamlined, decentralized intrusion detection design. It aims to combine signature analysis with statistical profiling to provide localized real-time protection of network services and infrastructure. The monitor has three computational parts: a signature-based engine, a statistical profiling engine, and countermeasure unit called the resolver. Monitors have a versatile application programmers' interface leading to interoperability with the analysis target, and with other third-party intrusion-detection tool suites.

3.4.2 AID

The development of AID (Adaptive Intrusion Detection system) is ongoing at the Brandenburg University of Technology at Cottbus [Aid]. The system is designed for network audit based monitoring of local area networks and used for investigating network and privacy oriented auditing.

The system's client-server architecture, as shown in Figure 3.2, consists of a central monitoring station (hosts a manager which is the client and an expert system) and several agents (servers) on the monitored hosts. The agents utilize the audit data that is collected by the local audit functions and convert them into an operating system independent data format. The audit data is then transferred to the central monitoring station, buffered in a cache and analysed by an RTworks based real-time expert system. The manager maintains security administration of the monitored hosts and also controls their audit functions, requests new audit data by controlled polling and returns the decisions of the expert system to the agents. Communication between the manager and the agents is accomplished by secure RPC.

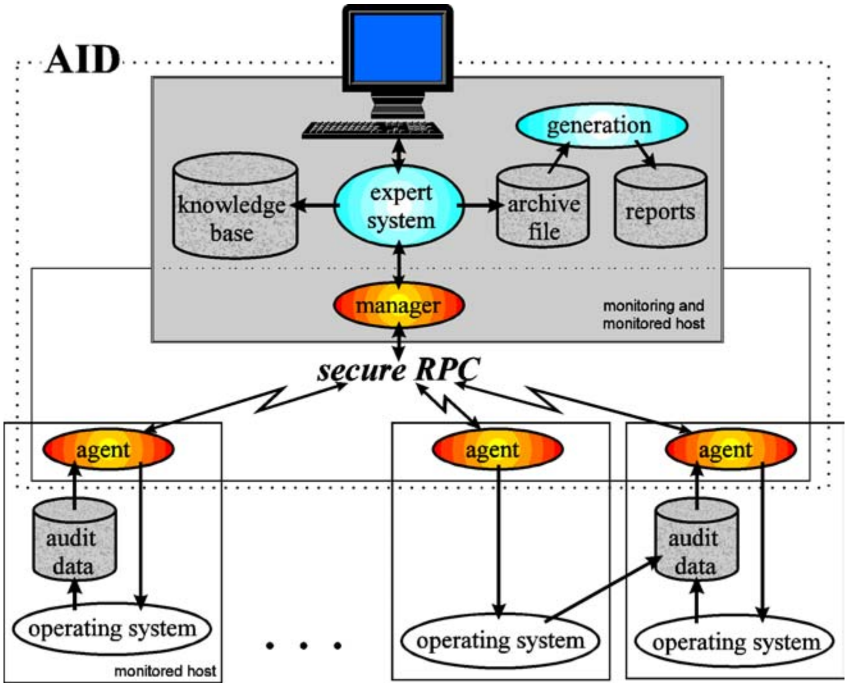


Figure 3.2: AID Architecture

The expert system utilizes a knowledge base with state-oriented attack signatures. These signatures are modeled by deterministic finite state machines and implemented as rule sequences. A security officer can access the relevant monitoring capabilities via a graphical user interface. The expert system also archives data on finished and cancelled attacks, involved users, and creates security reports.

AID was tested in a LAN environment consisting of Sun SPARC stations running Solaris 2.x and TCP/IP. Proposed future work included host-oriented network audit, integration of functions for active defense of detected attacks and adaptions/extensions for monitoring of Windows NT environments.

3.4.3 NID

NID [Nid] is a suite of software tools that helps detect, analyze, and gather evidence of intrusive behavior occurring on an Internet Protocol (IP) – based Ethernet or Fiber Distributed Data Interface (FDDI) network. NID operates passively on a stand-alone host (rather than residing on the hosts it is monitoring), and is responsible for collecting data and/or statistics about network traffic. NID operates within a security domain - a collection of hosts and/or sub-networks that need to be monitored. A security domain can consist of either a subset of a network or the entire network to which NID is directly connected. The security domain can be further refined by only looking at traffic from particular Internet services.

Figure 3.3 illustrates a simple scenario in which NID is used to monitor a collection of hosts. In the example, NID resides on a single host within subnet 2. The security domain is defined to consist of hosts C and D. Assuming that NID is only monitoring traffic entering the security domain, NID can detect, analyze, and gather evidence of intrusive behavior on all network traffic originating from hosts A, B, or E and destined for hosts C or D. Traffic between C and D is not monitored nor is any traffic specific to subnet 1.

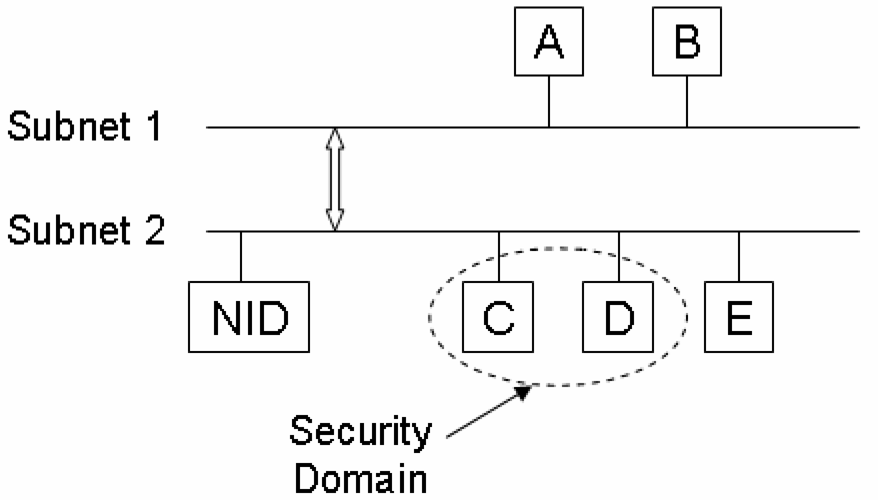


Figure 3.3: NID Illustration

NID utilizes the following techniques for detecting intrusive behavior:

- **Attack Signature Recognition:** NID can examine data packets for strings (or patterns) associated with known attacks and whose presence suggests the possibility of malicious behavior. Upon a pattern match, NID can signal an alarm, display the context in which the pattern was found, and begin saving the session's network packets to an output file. Analysis of these files can determine intrusive behavior.

- **Vulnerability Risk Model:** NID has a built-in vulnerability risk model that computes warning values based on: 1) a host's security level 2) authentication requirements of the service used and 3) recent transaction history of a host. These warning values are used to indicate connections having the greatest vulnerability risks.

- **Anomaly Detection:** NID can monitor and report certain anomalies as they occur. The two primary sets of anomalies that NID is able to detect are: 1) activities associated with untrusted or unexpected hosts and 2) known network attacks such as port scanning and/or SYN flood.

3.5 Requirements of an IDS

Even though, intrusion detection has existed as a research field for a number of years, several fundamental factors remain unaddressed. Two of the main factors are *effectiveness*—how to make the intrusion detection system classify malign and benign activity correctly—and *efficiency*—how to run the intrusion detection system in a cost effective manner. Many different demands can be made on intrusion detection systems. An important requirement is that it be effective i.e. that it should detect a substantial percentage of intrusions into the supervised system, while still keeping the false alarm rate at an acceptable level. We define a false alarm (also called false positive) as flagging by the intrusion detection scheme as an intrusion when in reality it is not an intrusion. We discuss some of the requirements of an Intrusion Detection System (IDS) below [Ax00]:

- **Accuracy:** refers to the correctness of detection of attacks and the absence of false alarms. An example of *inaccuracy* would be an IDS considering a false alarm as a legitimate intrusion.

- **Performance:** refers to the rate at which audit events are processed. Intrusion detection on a real-time basis would not be possible if the performance is low.

- **Completeness:** refers to the ability to detect all attacks. As a direct consequence of this is *incompleteness*, which really means that an attack was not detected.

- **Time of detection:** Two main groups can be identified: those that attempt to detect intrusions in *real-time* or near real-time, and those that process audit data with some delay, postponing detection (*non-real-time*), which in turn delays the time of detection.

3.6 Intrusion Detection in Wireless Ad hoc Networks

Research work done in intrusion detection for traditional, wired networks cannot be easily applied to wireless networks. Even though the core attributes and requirements of an IDS remain the same, designing an IDS for wireless networks has its own set of complexities. In this section, we discuss some of these difficulties.

3.6.1 Difficulties of Wireless Environment

In this subsection, we present the difficulties involved in designing an IDS for wireless environments [Sm01].

3.6.1.1 Lack of Physical Wires

The most obvious difference when building an IDS in a wireless environment is the fact that an attacker no longer has to gain physical access to the system in order to compromise the security of the network. Potentially, it is very simple for someone to eavesdrop on network traffic in a wireless environment owing to the lack of a physical medium to gain access to the traffic.

3.6.1.2 Bandwidth Issues

Wireless networks have more constrained bandwidth as compared to wired networks. This problem can manifest itself in a number of different ways when an IDS is using wireless communication to convey information between parts of IDS on separate nodes. An IDS in a mobile environment must be extremely careful to limit the amount of communication that takes place between nodes. A second problem that may possibly arise because of limited bandwidth is erroneous behavior of the IDS due to communication delay between nodes.

3.6.1.3 Difficulty of Anomaly/Normality Distinction

Distinguishing an anomaly from normalcy has always been somewhat difficult for wired IDSs, and wireless IDSs are no different. If nodes in a network receive false or old routing information from a particular node, then it is difficult to verify if that particular node has been compromised or not. An attacker could have taken the control of the node to send false information to other nodes in the network, or the node could just be temporarily out of sync due to fast movement or other processing requirements.

3.6.1.4 Secure Communication Between IDS Agents

It is likely that, in a wireless network, there will have to be portions of the IDS running on each individual node in the network. Each of these IDS agents will have to communicate with other IDS agents in the network to convey information relating to the status of the system. It is crucial that the information being passed from agent to agent be encrypted as to not allow an attacker to gain access to the communication. An attacker can modify the message content and, not only get away without being detected but also wreak havoc, for instance, by falsely implicating innocent nodes as malicious.

3.6.2 Difficulties of Ad-hoc Environment

In this subsection, we present the difficulties involved in designing an IDS for ad hoc networks [Sm01]. An ad hoc environment is really a subset of the wireless environment and so has all the difficulties of wireless environment plus some more of its own by virtue of being mobile and ad hoc.

3.6.2.1 Lack of Centralized Access/Audit Point

Unlike most static, wired networks which have specific repositories, the lack of centralized audit points (such as switches, gateways, etc.) in ad hoc networks presents difficult problems for intrusion detection. IDSs in ad hoc networks are, thus, limited to use only the current traffic coming in and out of the node as audit data. The algorithms that the IDS uses must be distributed, and take into account the fact that a node can only see a portion of the network traffic. Moreover, distributed algorithms are more difficult to deal with, as is the case with ad hoc networks.

3.6.2.2 Possibility of a Node Being Compromised

Since ad hoc networks are dynamic and nodes can move about freely, there is a possibility that one or more nodes could be captured and compromised, especially if the network is in a hostile environment. If the algorithms of the IDS are cooperative, it becomes important to be skeptical of which nodes one can trust. IDSs on ad hoc networks have to be wary of attacks made from nodes in the network itself, not just attacks from outside the network.

3.6.2.3 Difficulty In Obtaining Enough Audit Data

Mobile networks do not communicate as frequently as their wired counterparts. Bandwidth issues, and other issues such as battery life, contribute to this factor. This lack of communication can become a problem for IDSs attempting to define rules of normalcy for anomaly detection. If only a small amount of data is available to establish normal activity association rules, it is very hard to distinguish an attack from regular network use.

In summary, we must delve into the following issues in developing a viable intrusion detection system for mobile ad hoc networks [ZhLe00]:

- What is a good system architecture for building intrusion detection and response systems that fits the features of mobile ad-hoc networks?

- What are the appropriate audit data sources? How do we detect attacks based on partial, local audit traces – if they are the only reliable audit sources?

A wireless mobile environment throws up a lot of challenges for designing an IDS and our research work aims to address some of these issues. In the next section, we have presented work in the field of intrusion detection in mobile ad hoc networks. This should give the reader an idea of the numerous avenues in this relatively fledgling research field of intrusion detection in ad hoc networks.

3.7 Related Work

In this section, we present a state-of-the-art view of research in intrusion detection systems for wireless ad hoc networks including proposed architectures and ongoing development work.

3.7.1 A pioneering IDS for MANETs

Zhang and Lee proposed an architecture for intrusion detection and response involving all nodes in the wireless ad-hoc network [ZhLe00]. Each node is responsible for detecting signs of intrusion locally and independently, but neighboring nodes can collaboratively investigate in a broader range. In the system, individual IDS agents are placed on each and every node. Each IDS agent runs independently and monitors local activities (including user and systems activities, and communication activities within the radio range). It detects intrusion from local traces and initiates response. If anomaly is detected in the local data, or if the evidence is inconclusive and a broader search is warranted, neighboring IDS agents will cooperatively participate in global intrusion detection actions.

These individual IDS agents collectively form the IDS system to defend the wireless ad-hoc network. Internally, an IDS agent is structured into six pieces as shown in Figure 3.4. The data collection module is responsible for gathering local audit traces and activity logs. Next, the local detection engine uses this data to detect local anomaly. The cooperative detection engine is utilized when detection methods need broader data sets or require collaborations among IDS agents which are places on each node. Both the local response and global response modules provide intrusion response actions. The local response module triggers actions local to this mobile node, for example, an IDS agent alerting the local user, while the global one coordinates actions among neighboring nodes, such as the IDS agents in the network electing a remedy action. Finally, a secure communication module provides a high-confidence communication channel among IDS agents. The proposed approach however, does not clearly mention the details about response through the secure communication channel.

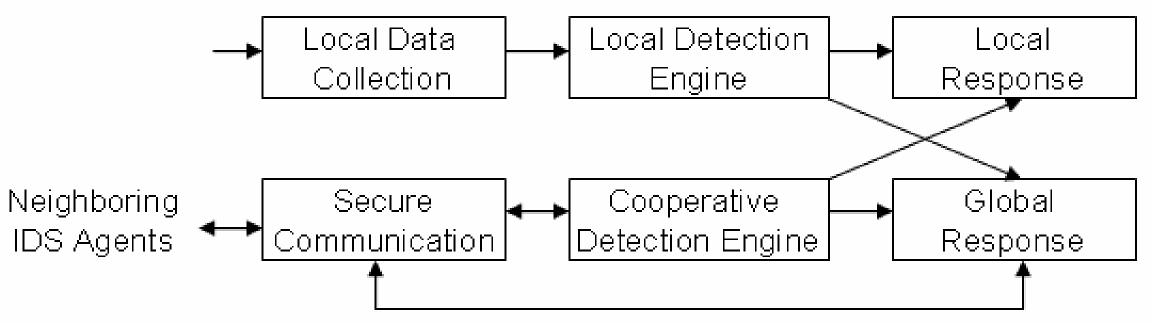


Figure 3.4: Zhang-Lee's IDS for Wireless Ad Hoc Networks.

3.7.2 AODV Protocol-based IDS

An Intrusion Detection Model (IDM) has been proposed to enhance security in Ad hoc On Demand Distance Vector (AODV) protocol [BhAg01] as shown in Figure 3.5. Each node employs IDM that utilizes neighborhood information to detect misbehavior of its neighbors. When the misbehavior count for a node exceeds a predefined threshold, the information is sent out to other nodes as part of global response. The other nodes receive this information and check their local *"Malcount"* for the broadcasted malicious node and add their result to the initiator's response. All this leads to secure communication in AODV protocol.

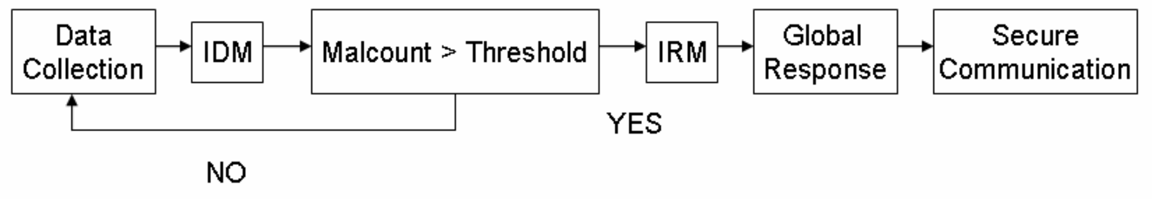


Figure 3.5: Handling of internal attacks in AODV-protocol based IDS

In the Intrusion Response Model (IRM), a node identifies that another node has been compromised when its *Malcount* increases beyond the threshold value for that allegedly compromised node. In such cases, it propagates this information to the entire network by transmitting a special type of packet called "MAL" packet as part of the "Global Response" module. If another node also suspects that the node that has been detected as compromised, it reports its suspicion to the network and retransmits another special type of packet called "REMAL" packet. If two or more nodes report about a particular node, another of the special packets called "PURGE" packet is transmitted to isolate the malicious node from the network. All nodes that have a route through the compromised node look for newer routes. All packets received from a compromised node are dropped. This approach too lacks clear description of how secure communication is handled.

3.7.3 Local Intrusion Detection System based on SNMP Data

Albers et al have proposed a distributed intrusion detection mechanism by implementing a *Local Intrusion Detection System* (LIDS) on each node [Albers02]. In order to make this detection a global concern for the community, the different LIDS co-existing within it collaborate. A LIDS may also take into account the intrusions detected by other members of its community. The different LIDS in a community will thus exchange two types of data:

- Security data: To obtain complementary information from collaborating hosts.
- Intrusion alerts: To inform others of a locally detected intrusion.

The LIDS architecture makes use of SNMP (Simple Network Management Protocol) data located in MIBs (Management Information Base) as the audit source as shown in Figure 3.6. To confirm a detected suspicious action, a LIDS may need complementary external information. Considering the unreliability of the UDP transport protocol underlying SNMP and the dynamic topology of MANETs, the authors have proposed to use mobile autonomous and adaptative agents to transport SNMP requests to remote hosts.

All external or internal communication with an LIDS relies on a common communication framework. If the framework can understand *Intrusion Detection Message Exchange Format* (IDMEF) and *Intrusion Detection Exchange Protocol* (IDXP) messages [idwg] delivered by the network, then, it will facilitate cooperation with other open IDS. Any IDS capable of using the standards implemented by the framework may thus act as a remote data source either providing security data or intrusion alerts.

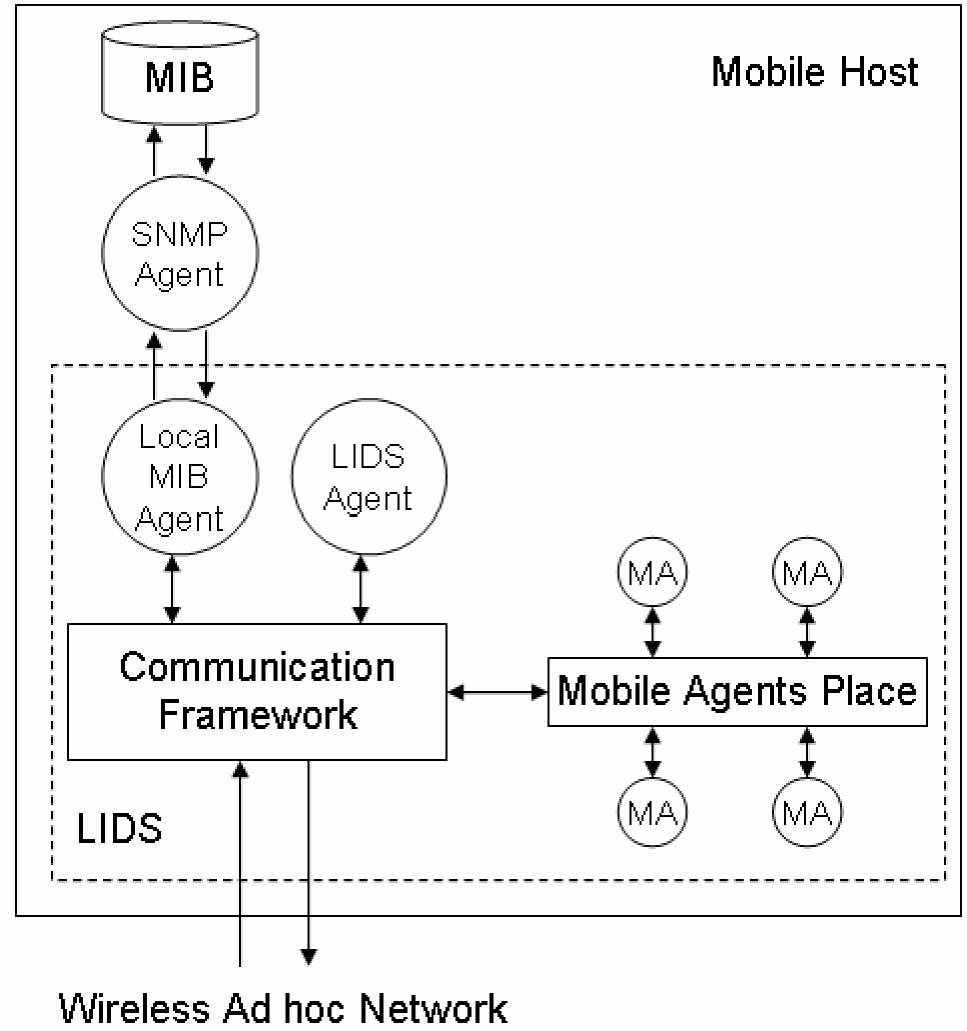


Figure 3.6: LIDS Architecture based on SNMP Data

A local LIDS Agent, is in charge of local intrusion detection and response and reacts to intrusion alerts provided by other nodes in order to protect itself against this intrusion. Mobile Agents collect and process data on remote hosts. They may decide to transfer the results of a computation back to their home LIDS or to migrate to another node for further investigation. The security control of these agents can be taken in charge by the Mobile Agent Place. The local MIB agent provides a means of collecting MIB variables for mobile agents or the Local LIDS Agent. If SNMP runs on the node, the Local MIB Agent simply acts as an interface with the running SNMP agent. Otherwise, an SNMP based agent is required to be developed specifically, to allow optimized updates and retrieval of the MIB variables used by intrusion detection. The Local MIB agent would, in that case, acts as an interface between the LIDS and this tailor-made agent. The Local LIDS Agent could theoretically use either misuse or anomaly detection. But there is no discussion of either approach. Using MIB variables to define signatures of attacks has been considered but not implemented.

As soon as a LIDS detects an intrusion locally, it informs the other nodes in the network. Locally, the user could choose to refuse connections with the suspicious node, to exclude it when performing cooperative actions, or to exclude it from its community until it re-authenticates itself. By being informed of intrusions on remote hosts, a LIDS can act as a preventive security tool and prevent the intruder from attacking it.

3.7.4 Watchdog-Pathrater Approach

Sergio Marti et al. discussed two techniques that improve throughput in ad hoc network in the presence of compromised nodes that agree to forward packets but fail to do so [MaGiLaBa00]. A node may misbehave because it is overloaded, selfish, malicious or broken. An overloaded node lacks the CPU cycles, buffer space or available network bandwidth to forward packets. A selfish node is unwilling to spend battery life, CPU cycles, or available network bandwidth to forward packets not of direct interest to it, even though it expects others to forward packets on its behalf. A malicious node could launch a denial of service attack by dropping packets. A broken node might have a software fault that prevents it from forwarding packets.

To mitigate the decrease in the throughput due to these nodes, the authors use a *watchdog* that identifies misbehaving nodes and a *pathrater* that helps routing protocols avoid these nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is flagged as misbehaving by the "watchdog". The pathrater, run by each node in the network, combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node in the network it knows about. It calculates a path metric by averaging the node ratings in the path.

The watchdog operation is illustrated in Figure 3.7. Suppose there exists a path from node S to D through intermediate nodes A, B, and C. Node A cannot transmit all the way to node C, but it can listen in on node B's traffic. Thus, when A transmits a packet for B to forward to C, A can often tell if B transmits the packet. If encryption is not performed separately for each link, which can be expensive, then A can also tell if B has tampered with the payload or the header.

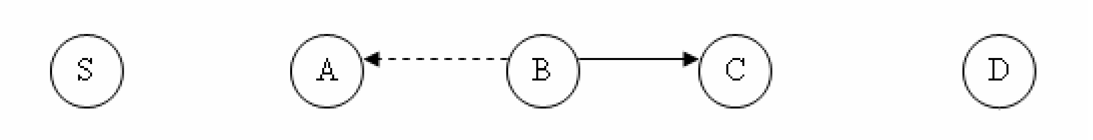


Figure 3.7: Watchdog operation.

Every time a node fails to forward the packet, the watchdog increments the failure tally. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving. The watchdog technique has advantages and weaknesses. DSR with the watchdog has the advantage that it can detect misbehavior at the forwarding level and not just the link level. Watchdog's weaknesses are that it might not detect a misbehaving node in the presence of the following:

1. Ambiguous collisions: Prevents A from overhearing the transmission from B as shown in Figure 3.8.
2. Receiver collisions: Node A can only tell whether B has sent packet, but it cannot tell if C received it or not as shown in Figure 3.9.
3. Limited transmission power: Misbehaving node could limit its transmission power such that the signal is strong enough to be overheard by the previous node but too weak to be received by the true recipient.
4. False misbehavior: This occurs when node falsely reports other nodes as misbehaving.
5. Partial dropping: A node can circumvent the watchdog by dropping packets at a lower rate than the watchdog's configured minimum misbehaving threshold.

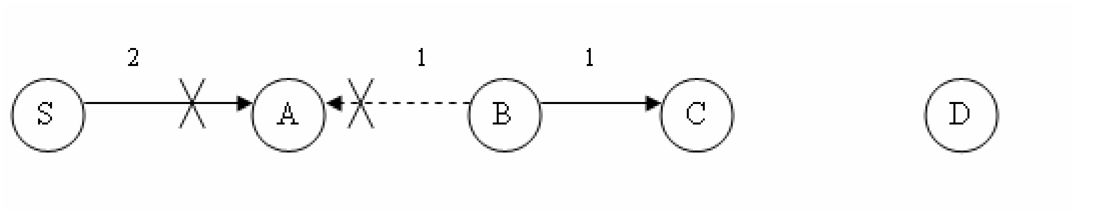


Figure 3.8: Node A does not hear B forward packet 1 to C, because B's transmission collides at A with packet 2 from Source S.

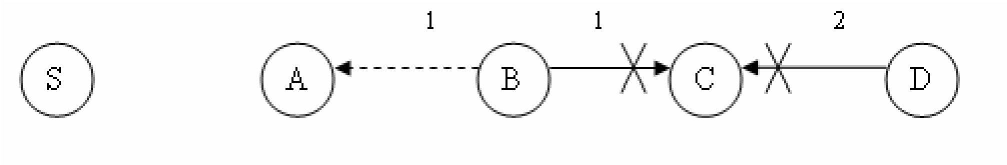


Figure 3.9: Node A believes that B has forwarded packet 1 on to C, though C never received the packet due to a collision with packet 2.

3.7.5 IDS based on a Static Stationary Database

A distributed IDS has been proposed at Mississippi State University wherein each node on the network has an IDS agent running on it [Sm01] as shown in Figure 3.10. The IDS agents on each node in the network work together via a cooperative intrusion detection algorithm to decide when and how the network is being attacked. The architecture is divided into parts: the Mobile IDS Agents, which reside on each node in the network, and the Stationary Secure Database, which contains global signatures of known misuse attacks and stores patterns of each users normal activity in a non-hostile environment. Each node has an IDS agent running on it at all times that is responsible for detecting intrusions based on local audit data and participating in cooperative algorithms with other IDS agents to decide if the network is being attacked.

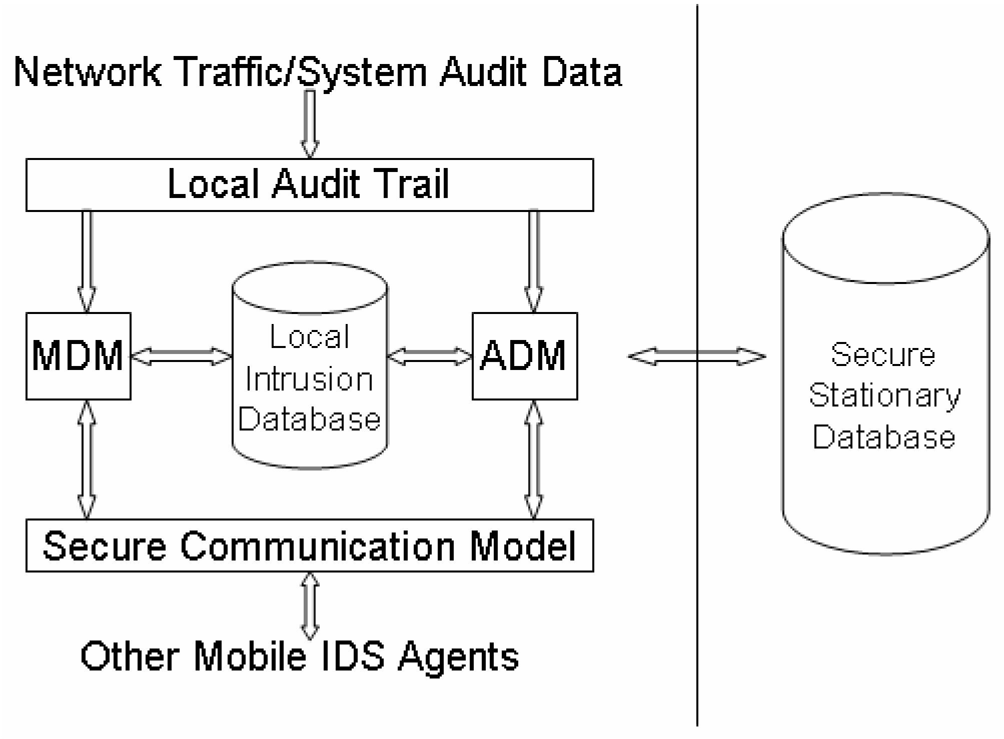

Network Traffic/System Audit Data
Local Audit Trail
MDM
Local Intrusion Database
ADM
Secure Stationary Database
Secure Communication Model
Other Mobile IDS Agents

Figure 3.10: IDS based on a Static Stationary Database.

Each agent has five parts: the Local Audit Trial, the Local Intrusion Database (LID), the Secure Communication Module, the Anomaly Detection Modules (ADMs), and the Misuse Detection Modules (MDMs). The Stationary Secure Database (SSD) acts as a secure, trusted repository for mobile nodes to obtain information about the latest misuse signatures and to find the latest patterns of normal user activity. It is assumed that the attacker will not compromise the SSD, as it is stored in an area of high physical security. The mobile IDS agents collect and store audit data (such as user commands, network traffic, etc.) while in the field, and transfer this information when it is attached to the SSD. The SSD then uses this information for data mining of new anomaly association rules (newest misuse signatures are specified in the SSD). When the IDS agents are connected to SSD, they gain access to the latest attack signatures automatically. Using the SSD to communicate the new attack signatures and establish new patterns of normalcy limits the amount of communication that must take place between IDS agents in the mobile ad hoc network.

3.7.6 Intrusion Detection Using Mobile Agents

Kachirski and Guha have proposed a distributed intrusion detection system for ad hoc wireless networks based on mobile agent technology [KaGu02]. By efficiently merging audit data from multiple network sensors, their bandwidth-conscious scheme proposes to analyze the entire ad hoc wireless network for intrusions at multiple levels, as shown in Figure 3.11, and tries to inhibit intrusion attempts. This scheme attempts to provide a lightweight, low-overhead mechanism based on mobile security agent concept. Agents are dynamically updateable, lightweight, have limited functionality and can be viewed as components of flexible, dynamically-configurable IDS. This scheme restricts computation-intensive analysis of overall network security state to a few key nodes. These nodes are dynamically elected, and overall network security is not entirely dependent on any particular node. The modular approach taken has advantages such as increased fault tolerance, communications cost reduction, improved performance of the entire network, and scalability.

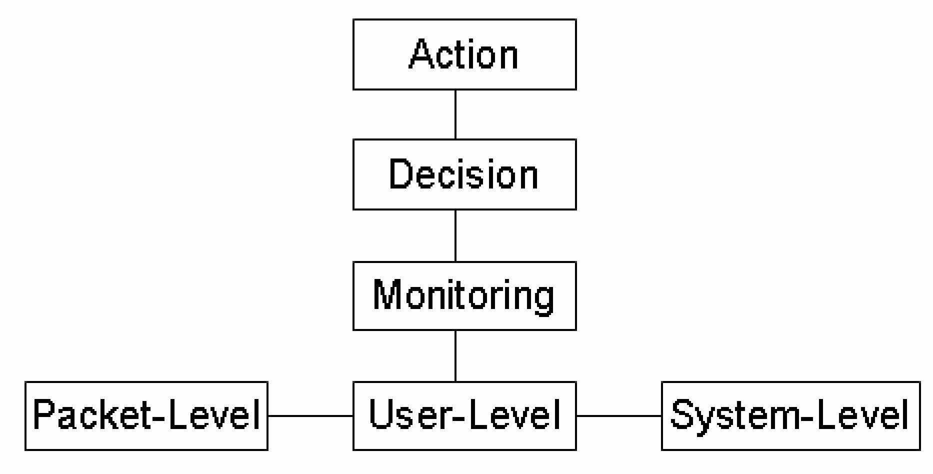


Figure 3.11: Intrusion Detection using Mobile Agents.

It is a non-monolithic system and employs several sensor types that perform specific functions, such as:

• *Network monitoring:* Only certain nodes have sensor agents for network packet monitoring to preserve total computational power and battery power of mobile hosts.
• *Host monitoring:* Every node on the mobile ad hoc network is monitored internally by a host-monitoring agent. This includes monitoring system-level and application-level activities.
• *Decision-making:* Every node decides on its intrusion threat level on a host-level basis. Certain nodes collect intrusion information and make collective decisions about network-level intrusions.
• *Action:* Every node has an action module responsible for resolving intrusion situation on a host.

There are three major agent categories – monitoring, decision-making and action agents. Some are present on all mobile hosts, while others are distributed to only a selected group of nodes. While all the nodes accommodate host-based monitoring sensors of IDS, a distributed algorithm has been utilized to assign a few nodes to host sensors that monitor network packets, and agents that make decisions. The mobile network is logically divided into clusters with a single cluster head for each cluster that monitors packets within the cluster. The selected nodes host network-monitoring sensors that collect all packets within communication range, and analyze them for known patterns of attacks. Monitoring agents are categorized into packet monitoring sensors, user activity sensors and system-level sensors. While packet monitoring is activated only when a node participates in the network (is a member of a cluster), local activity sensors are present on each node and are active all the time. Each sensor performs certain level of monitoring activity and reports anomalies to the decision agents. Packet-monitoring agents reside on each selected node.

Local detection agents are located on each node of an ad-hoc network, and act as user-level and system-level anomaly-based monitoring sensors. These agents look for suspicious activities on the host node, such as unusual process memory allocations, CPU activity, I/O activity, user operations (invalid login attempts with a certain pattern, super-user actions, etc). If an anomaly is detected with strong evidence, a local detection agent terminates the suspicious process or locks out a user and initiates re-issue of security keys for the entire network. If some inconclusive anomalous activity is detected on a host node by a monitoring agent, the node is reported to the decision agent of the same cluster that the suspicious node is a member of. If more-conclusive evidence is gathered about this node from any source (including packet monitoring results from a network-monitoring agent), the action is undertaken by the agent on that node.

3.8 Chapter Summary

In this chapter, we presented a background on intrusion detection including a few examples of IDSs for wired networks. The attributes and requirements of an IDS were discussed along with the difficulties involved in designing an IDS for wireless mobile ad hoc networks. Research done for wired networks cannot be easily applied to ad hoc networks. The architecture for an intrusion detection system for ad hoc networks needs to be distributed. This is because of the fact that the underlying network structure itself is not centralized and there is no network management machinery that can analyze audit data for the entire network as such. The next chapter provides details of our intrusion detection scheme for wireless mobile ad hoc networks and discusses the design rationale.

4. Proposed Scheme

In this chapter, we discuss our intrusion detection scheme in detail. We have proposed an intrusion detection architecture for MANETs based on combining the principles of misuse and anomaly detection. Our protocol-independent design makes use of a self-adjusting threshold scheme that accurately detects *a priori* known attack patterns. Since bandwidth constraints of MANETs necessitate the efficiency of any security scheme, we have focused on security but not at the expense of overloading the already resource-constrained ad hoc network.

4.1 Novelty of Proposed Approach

Even though wireless networks have been under the research microscope, the realm of ad hoc networks (or MANETs) is a relatively new one. The research community is still working towards developing a viable intrusion detection model for security in ad hoc networks. Zhang and Lee proposed a distributed anomaly detection model based on classification algorithms [ZhLe00]. This approach tries to flag abnormal behavior by looking at the routing tables. Anomaly detection, by itself, flags behavior deviating from "normal" as an intrusion, leading to a high degree of false alarms as compared to misuse detection.

Albers et al have proposed the use of misuse as well as anomaly detection for their intrusion detection system though no specific details of detection have been discussed [Albers02]. The audit data source used SNMP (Simple Network Management Protocol) data located in MIBs (Management Information Base), as an audit source for Local Intrusion Detection System (LIDS). Any a priori trust assumptions (LIDS is based on trust model) can be especially damaging in military scenarios. This is because, in sensitive applications like a battlefield, nodes can be captured and one cannot afford to assume trustworthiness of any node. Moreover, the use of SNMP data located in MIBs might take its toll on the network resources, which are already scarce in ad hoc networks.

Another proposed approach for MANETs is a watchdog approach focused on improving throughput [MaGiLaBa00]. Although this scheme is not an intrusion detection system, *per se*, it can be thought of as one, owing to its attempt to flag misbehaving nodes by means of misuse detection. The watchdog tries to identify misbehaving nodes and needs to know where a packet should be in two hops. A malicious node can circumvent the threshold level used to identify a misbehaving node by dropping packets at a lower rate. This approach is limited to source routing protocols and does not take into consideration the adverse effect a security scheme can have on an ad hoc network especially latency and packet delivery.

Bhargava and Agrawal have proposed security enhancements for the AODV protocol [BhAg01]. Their approach is based on only misuse detection with fixed thresholds. The concept of fixed threshold values can be exploited by attackers. It is possible to avoid being detected by working under the fixed levels knowing fully well that the IDS utilizes a single constant threshold level to flag intrusive behavior.

Thus, the requirements made on intrusion detection schemes by the ad hoc environment are not fully satisfied by any one scheme. Each of the proposed approaches has its share of advantages and weaknesses. In this subsection, we present the salient features of our own proposed intrusion detection scheme.

4.1.1 Compound detection

Anomaly detection is based on statistical analysis of audit data and there being a fuzzy line between normalcy and anomaly, intrusion detection systems based purely on anomaly detection lead to a high degree of false positives as compared to intrusion detection systems based on misuse detection. This can have a negative impact on the network throughput. Our scheme is inherently based on misuse detection that accurately matches signatures of known attacks. However, anomaly detection is also utilized to lay the rules for a normal profile of a network, which improves the overall robustness of an intrusion detection scheme. Ideally, a compound intrusion detection scheme can provide the best of both worlds. The novelty of the approach lies in the self-adjusting threshold approach taken to integrate the best of misuse and anomaly detection [NaMi03].

4.1.2 Adaptive

The novelty of our intrusion detection scheme lies in the fact that our mechanism aims to integrate misuse detection with anomaly detection leading to a viable compound intrusion detection scheme for MANETs. More specifically, a predefined threshold for a misuse detection scheme, *per se*, cannot be considered as a fool-proof mechanism. Attackers, if they target the loopholes in the IDS itself, can operate below threshold levels and get away without ever being detected. Our proposed mechanism not only makes use of the accuracy of misuse detection but also adapts itself to the network by gathering intrusion data and combining it with audit data in order to adjust the threshold. This has the effect of making the underlying architecture versatile and protecting the IDS itself from the dangers of an inflexible approach.

4.1.3 Effective

The prototype implementation has shown significant results about the accuracy of our intrusion detection scheme. The success of intrusion detection systems is measured by the accuracy of detection that is reflected by the correctness with which intrusions are flagged. Simulation results have provided conclusive data about the effectiveness of our scheme under varying (even extreme at times) network conditions and multiple instances of each type of attack. On an average, our scheme detects intrusions with an average accuracy of over 90%. Moreover, the low degree of false alarms adds significant value to the effectiveness of our scheme.

4.1.4 High performance

The battery powered operation of wireless mobile ad hoc networks gives attackers ample opportunity to launch denial-of-service attacks by creating additional transmissions or expensive computations carried out by a mobile node in an attempt to exhaust its batteries. Ideally, one would want the ad hoc network to be secured but an inefficient security scheme basically defeats the purpose behind securing the network. Thus, bandwidth and battery constraints in MANETs make it paramount for an IDS to be efficient and energy conserving. Unlike the watchdog approach, our scheme focuses not only on accuracy of detection but also non-degradability of network performance including latency and packet delivery. Our proposed approach for a viable compound intrusion detection scheme for MANETs focuses on the underlying constraints of the network. We have focused on correctly detecting malicious nodes with a low (ideally zero) level of false positives but not at the expense of hogging the network resources. The efficiency of our security scheme is an important aspect in the overall scheme of things.

4.1.5 Protocol-independent

A wireless ad hoc network wholly depends on the underlying routing protocol for its functional working. We have successfully implemented our intrusion detection scheme for Destination Sequenced Distance Vector (DSDV) routing protocol [PeBh94]. The design rationale for our IDS takes into consideration the wide range of routing protocols being proposed for ad hoc networks. Protocol-independence feature of our scheme allows the concept to be applied to any ad hoc scenario irrespective of the underlying routing protocol.

4.2 DSDV Protocol Details

Since the implementation of our IDS has been achieved for the DSDV routing protocol, we briefly present an overview of the protocol. DSDV routing protocol [PeBh94] is derived from a classical distance vector algorithm, Distributed Bellman-Ford (DBF) algorithm [JuTo87]. Enhancements are made in order to avoid the looping problem present in the basic DBF. Formation of loops is avoided by tagging each route table entry with a sequence number to order the routing information.

In DSDV, each node maintains a routing table, which has an entry for each destination in the network. The attributes for each destination are the next hop, metric (hop counts) and a sequence number, which is originated by the destination node. To maintain the consistency of the routing tables, DSDV uses both periodic and triggered routing updates; triggered routing updates are used in addition to the periodic updates in order to propagate the routing information as quickly as possible when there is any topological change. The update packets include the destinations accessible from each node and the number of hops required to reach each destination along with the sequence number associated with each route. Upon receiving a route update packet, each node compares it to the existing information regarding the route. Routes with old sequence numbers are simply discarded.

In case of a route with equal sequence number, the advertised route replaces the old one if it has a better metric. The metric is then incremented by one hop since incoming packet will require one more hop to reach the destination. Newly recorded routes are immediately advertised to its neighbors. When a link to the next hop is broken, any route through that next hop is immediately assigned an infinity metric and an updated sequence number. This is the only case where sequence numbers are not assigned by the destination. When a node receives an infinity metric, and it has an equal or newer sequence number with a finite metric, a route update broadcast is triggered. Therefore, routes with infinity metric will be quickly replaced by real routes propagated from the newly located destination. DSDV also employs a mechanism to damp out fluctuations in route table updates. In an environment where many independent nodes transmit routing information asynchronously, some fluctuations could develop. For example, a node could receive two routes to the same destination with the same sequence number; however, the one with the worst metric may arrive first. This could lead to continuous outbursts of route updates and fluctuations of the routing tables. DSDV solves this problem by using "settling time" data. Specifically, time duration until the route becomes stable (termed settling time) is predicted, and the settling time is allowed before advertising any new route information to the network. In other words, the settling time is used to decide how long to wait before advertising new routes. By delaying the advertisement of unstable routes, fluctuations of the routing tables are prevented, and consequently, the number of route updates is reduced.

Security mechanisms for distance-vector protocols for wired networks have been discussed by Smith et al [SmMuGa97]. However, their techniques of using digital signatures and sequence numbers for providing authentication and integrity services do not apply well in an ad hoc environment since they require knowledge of possible links between nodes. In an ad hoc network, any pair of nodes could be within radio range, thus, forming a link and the highly dynamic topology keeps these links changing over time.

4.3 Assumptions

Our research on intrusion detection involved the following assumptions:

1. The wireless links between nodes are bi-directional. This is necessary because the underlying routing protocol requires updates to be sent between mobile nodes.
2. Nodes are resource-constrained which underscores the need for efficiency.
3. Nodes are in a promiscuous mode such that they are on the lookout for potential intruders.
4. Byzantine failures (multiple attackers coordinating an attack) are disregarded.

4.4 Detection

Our design is based on a self-adjusting threshold scheme which sets a predefined threshold for each type of attack and then adjusts the threshold based on the network data gathered about the "normal" frequency of events to account for the highly dynamic network topology. The idea behind the adaptive approach towards thresholds is based on closed-loop control systems. These systems are known to have advantages such as reduction of the effects of the variation of the parameters, improvement in the system performance (transient and steady-state) and elimination of the effects of disturbances [Yang03]. These systems use the output of the process to modify the process to produce the desired result and thus continually adjust the process. By self correcting, such an approach minimizes effects of system changes. This network adaptive approach, as shown in Figure 4.1, combines anomaly detection (setting normalcy rules) with misuse detection (pattern-matching).

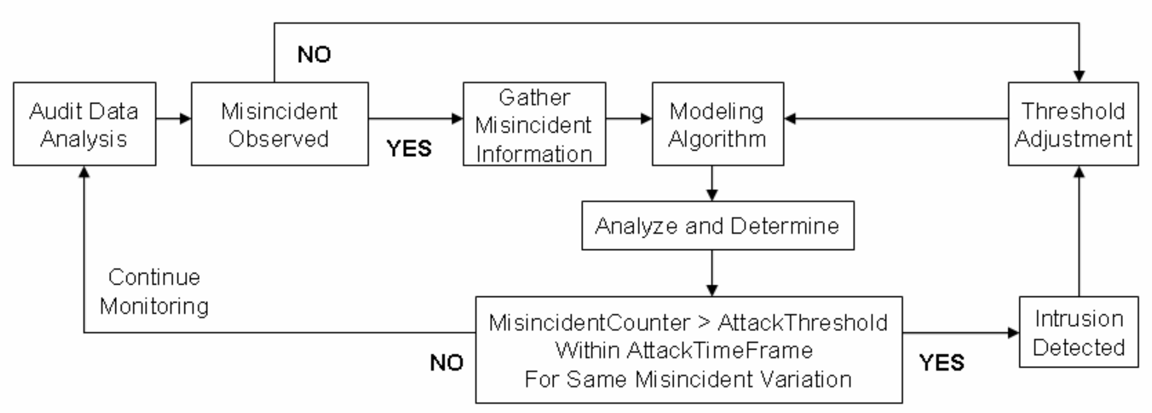


Figure 4.1: Proposed Intrusion Detection Scheme Illustration

Audit data is the basis of intrusion detection and it includes network information such as routing information, packet headers, etc. This information is utilized by each node for audit data analysis which reveals symptoms of any kind of attack. We refer to symptoms of an attack as misincidents, examples of which include an outdated routing update, dropped packet, a broadcast packet, etc. When a misincident is observed, whatever information is available about that misincident is gathered. This continual process builds up a repository of misincident information over time. The modeling algorithm utilizes the misincident information to analyze and determine if the mis-incidents of an attack are taking place at a rate, not generally seen as part of normal node behavior. This is done by checking if the threshold value for that attack is exceeded and within a time interval for all variations of misincidents. If the threshold is not exceeded then the node continues its monitoring. However, if the threshold is exceeded, then, the intrusion detection scheme reaches a reasonable conclusion that it is a possible malicious node launching an attack. This intrusion information is then passed onto the threshold adjustment module. The threshold adjustment module (which incidentally also utilizes audit data to adjust the thresholds) adjusts the threshold for that attack based on the intrusion data. We elaborate on the process in each of the following subsections:

4.4.1 Initialization

The scheme does not come into effect until after a certain settling time. This is because of the initialization process involved in setting up the thresholds and time frames. Flexibility in detecting a wide range of attacks necessitates a reasonably accurate setup of predefined initial thresholds and time intervals. The intrusion detection scheme might backfire if the predefined set of threshold values are not set appropriately. For instance, a threshold value for an attack might be too low. It is entirely possible that nodes might be flagged as malicious when in fact they are not malicious. Alternatively, if a threshold value for an attack is set too high, some malicious nodes launching that attack might get away with it and it may be too late by the time they are detected. Our approach is to analyze the network traffic and gather information about the normal behavior of the network. This is essentially an application of anomaly detection principle. Based on the normal frequency of events, we set up the predefined initial threshold values for each kind of attack. These values reflect the normal behavior of nodes in the network.

4.4.2 Audit Data Analysis

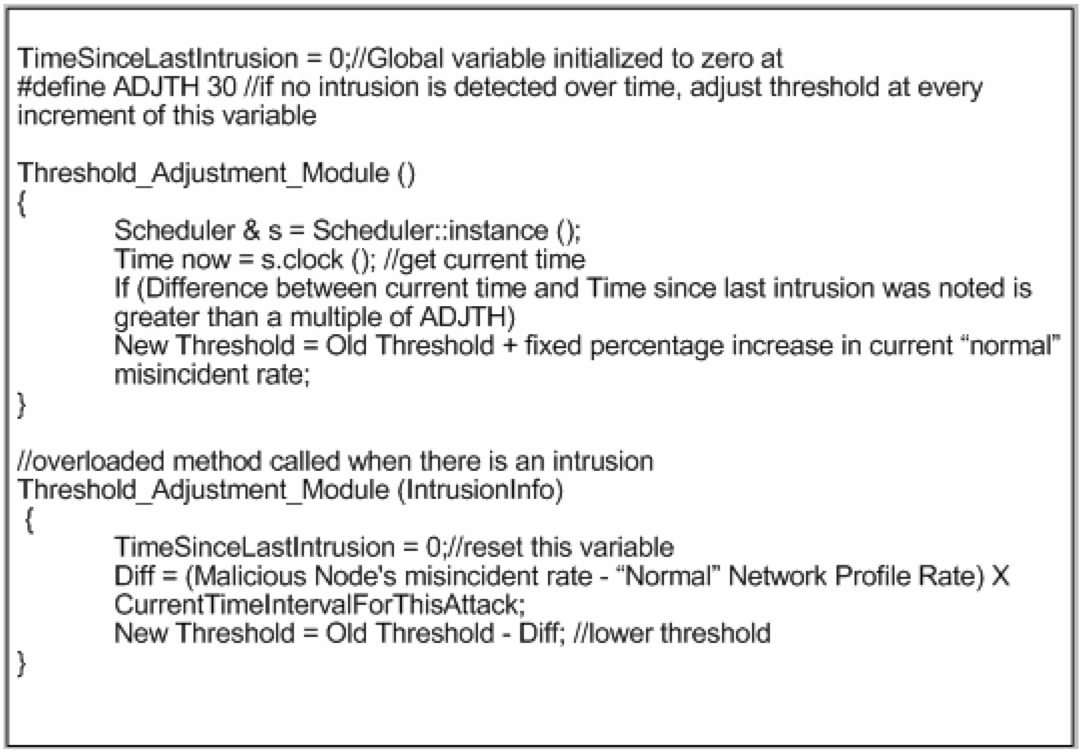
As mentioned before, the audit data consists of network information such as routing updates and packet headers. This audit data is utilized for analysis and identifying any misincident. One misincident does not imply an attack. Nor for that matter does a number of mis-incidents. However, a series of mis-incidents, all associated with the same variation of attack symptoms, occurring continually at a frequency higher than normal would certainly trigger alarm bells. A counter for each type of attack is incremented the moment a *mis-incident* (a symptom of a known attack pattern) is observed. For instance, Denial-of-Service attack could be launched by flooding the network with broadcast packets, in which case, a mis-incident would be an incoming broadcast packet. Also, a certain attack might be launched in more than one ways resulting in variations of misincidents for that attack. The proposed approach takes this into consideration by analyzing audit data for each variation of misincident.

Determining that an intrusion is taking place on the basis of one mis-incident would really be inconclusive and might even turn out to be a one-off anomalous but perfectly innocent instance of node behavior. The scheme updates the status of that node to *'suspicious'* and continues to monitor the node for any possible intrusion while maintaining mis-incident details such as sending node id, time of occurrence, etc. An attack would logically involve a recurrence of suspicious activity and a corresponding increase in mis-incidents. It is only when a predefined threshold is crossed, and more importantly, *within* a time frame (since multiple one-off instances of attack symptoms interspersed over a long time might not necessarily constitute an attack) and for the same variation of attack that the mobile node is determined as an intruder.

4.4.3 Threshold Adjustment

The underlying assumption for threshold adjustment is that if there is no intrusion detected as time passes by, there could be malicious nodes getting away without being detected by operating under threshold levels. The threshold adjustment module utilizes information passed to it about intrusions detected by the detection algorithm as well as information about time elapsed since last intrusion was detected. More specifically, each attack (note that an attack might have variations in the ways it is launched) has a threshold value and a time interval associated with it. The threshold adjustment module continually checks the current time against a variable holding the time elapsed since the last intrusion was detected. At regular intervals (driven by a predefined value which may be set depending on the network) during the time no intrusions are detected, the time interval for an attack is increased by a fixed value. This value is a fixed percentage increase in current “normal” misincident rate. This has the effect of trapping those possibly malicious nodes working under threshold levels. However, in this thesis, we have not focused on determining an optimal value for this fixed percentage increase. Future work could involve further experimentation to even dynamically adjust this fixed percentage increase.

When the intrusion detection scheme detects an intrusion or an attack, it gathers vital information about attackers. The variable holding time elapsed since last intrusion is reset to zero. Information associated with an intrusion has the frequency of misincidents generated by that malicious node, which was far above the normal profile of the network. By “normal” profile, we mean the average rate of misincidents observed in the network during the initial settling period. Based on this frequency of misincidents, we lower the threshold value by a fixed value. The new threshold value is obtained by multiplying the difference in the detected malicious node’s misincident rate and the current “normal” misincident rate by the time interval for the attack. Thus, an intrusion lowers the threshold and time interval values whereas no intrusion detected as time increases leads to an increase in the threshold and time interval values. This process is shown as an algorithm in Figure 4.2.

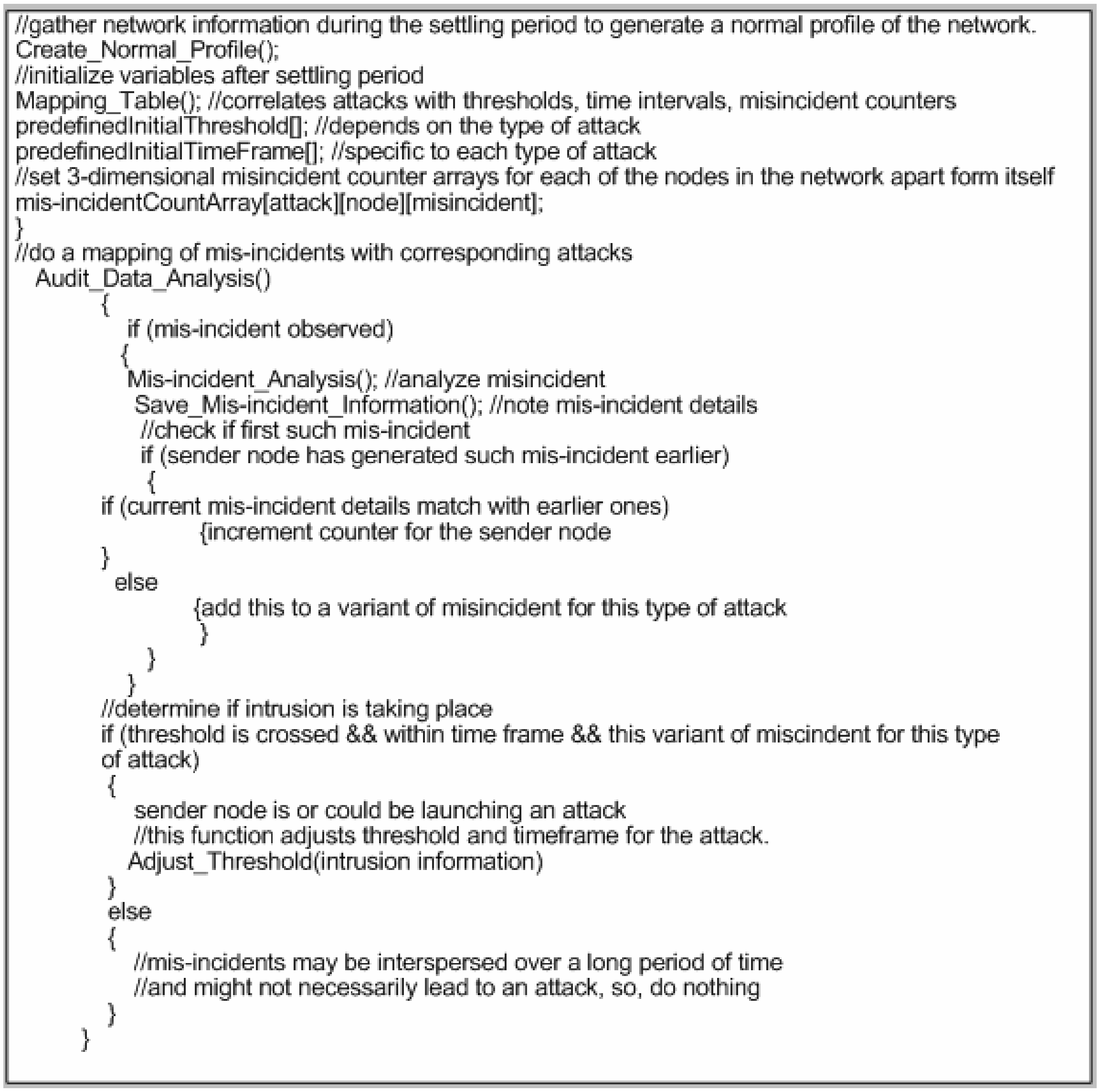


```
TimeSinceLastIntrusion = 0;//Global variable initialized to zero at
#define ADJTH 30 //if no intrusion is detected over time, adjust threshold at every
increment of this variable

Threshold_Adjustment_Module ()
{
        Scheduler & s = Scheduler::instance ();
        Time now = s.clock (); //get current time
        If (Difference between current time and Time since last intrusion was noted is
        greater than a multiple of ADJTH)
        New Threshold = Old Threshold + fixed percentage increase in current “normal”
        misincident rate;
}

//overloaded method called when there is an intrusion
Threshold_Adjustment_Module (IntrusionInfo)
 {
        TimeSinceLastIntrusion = 0;//reset this variable
        Diff = (Malicious Node's misincident rate - “Normal” Network Profile Rate) X
        CurrentTimeIntervalForThisAttack;
        New Threshold = Old Threshold - Diff; //lower threshold
}
```

Figure 4.2: Algorithm for Threshold Adjustment

4.4.4 Intrusion Detection Algorithm Description

The intrusion detection algorithm is shown in Figure 4.3. After the initial settling time utilized to gather network information and generate a “normal” network profile, we set up an array of predefined threshold values and another array of time intervals for each kind of attack. These arrays are initialized with values based upon the “normal” profile of the network. Each of the nodes holds an array of attack-specific misincident counters for all the other nodes in the network which records misincidents initiated by the sender. This misincident counter array is initialized to zero. We have a mapping table which correlates the thresholds, time intervals and misincident counters.



```
//gather network information during the settling period to generate a normal profile of the network.
Create_Normal_Profile();
//initialize variables after settling period
Mapping_Table(); //correlates attacks with thresholds, time intervals, misincident counters
predefinedInitialThreshold[]; //depends on the type of attack
predefinedInitialTimeFrame[]; //specific to each type of attack
//set 3-dimensional misincident counter arrays for each of the nodes in the network apart form itself
mis-incidentCountArray[attack][node][misincident];
}
//do a mapping of mis-incidents with corresponding attacks
  Audit_Data_Analysis()
         {
            if (mis-incident observed)
           {
            Mis-incident_Analysis(); //analyze misincident
             Save_Mis-incident_Information(); //note mis-incident details
              //check if first such mis-incident
              if (sender node has generated such mis-incident earlier)
               {
        if (current mis-incident details match with earlier ones)
                  {increment counter for the sender node
        }
          else
                  {add this to a variant of misincident for this type of attack
                   }
               }
            }
        //determine if intrusion is taking place
        if (threshold is crossed && within time frame && this variant of miscindent for this type
        of attack)
         {
             sender node is or could be launching an attack
             //this function adjusts threshold and timeframe for the attack.
            Adjust_Threshold(intrusion information)
         }
         else
         {
             //mis-incidents may be interspersed over a long period of time
             //and might not necessarily lead to an attack, so, do nothing
         }
      }
```

Figure 4.3: Intrusion Detection Algorithm

Based on the audit data analysis, nodes flag behavior as a misincident if it resembles a symptom of any attack. To expand the scope of detection, we do not restrict ourselves to one type of misincident for an attack. For instance, if we monitored only a specific type of misincident for an attack, it is entirely possible that the same attack is launched with a variation. We determine if such a misincident has occurred earlier. If yes, then we add it to the array holding misincident information about such types of misincidents. If not, then we create a variant of the existing misincident array for the same attack. For each misincident that is observed, the misincident counter is incremented. Essentially, each node has a 3-dimensional array available for analyzing misincident information related to each attack for each of the other nodes in the network and for each variation in the type of misincident for the attack. Thus, a repository of misincident information is built over time and the detection scheme utilizes this to determine if a possible intrusion is taking place. This is done by checking if the threshold value is exceeded for any of the variations of misincidents of the attack and within the specified time interval for that attack. If yes, then a possible intrusion has been detected. If not, then no action is taken. For example, an attack can be launched in multiple ways. Even though the basic type of the attack is the same, a rigid intrusion detection scheme might not detect a slight variation in the way that attack is launched. Each such variation of the attack would have recurring misincidents from the same source (a possible malicious node) but not a pattern of the attack variation. Also, a series of misincidents may be interspersed over a long period of time and might not necessarily constitute an attack. If, however, an intrusion is detected, the information about the intrusion is passed on to the threshold adjustment module. This facilitates the process of adapting to the ever-changing network profile, which is especially true in the case of ad hoc networks.

4.5 Attacks

Attacks can be classified as external and internal where *external* refers to a malicious node that does not belong to the network and mounts an attack, and *internal* refers to a node, which is a part of the network but is now compromised. Thus, internal attacks originate from within the network whereas external attacks originate from entities outside the network. Deterministic attacks really mean that the pattern followed by attackers is known for pattern-matching inherent in any misuse detection based IDS. Non-deterministic attacks, on the other hand, require a traffic pattern analysis and complex classification methods to correctly detect an anomaly from a normal behavior of a node. There is a high probability of confusing perfectly innocent but deviant behavior of a mobile node with a possible intrusion taking place. More specifically, a mobile node, say node X, may behave in a way such that other mobile nodes (or even the intrusion detection scheme) determine node X to be a possible intruder but, in reality, node X is an innocent node doing something not generally observed as part of its normal profile.

We have focused on a few attacks that are highly probable and have a high severity of negative impact leading to the downfall of the network. A compromised node can wreak havoc in a seemingly secure network by advertising non-existent links and flooding the network with routing packets. Compromised nodes could render an ad hoc network non-functional. A simplistic instantiation of a compromised node would be a node not cooperating with others, which can have ramifications on the correct functioning of wireless ad hoc networks relying on cooperation among nodes. Without any underlying infrastructure or overall network management, the crux of MANET functionality solely relies on the correct working of the routing protocol, which, in our case, is the DSDV protocol. Also, with the network resources already scarce, malicious attackers can make use of the bandwidth-constrained network characteristic to launch a Denial-of-Service which can lead to non-availability of network resources. We discuss some of the deterministic attacks that can be accurately detected by our IDS:

4.5.1 Compromised Node

Detecting compromised nodes is based, in theory, on the way 'ping' works [RFC 792]. No response from a node indicates a misincident but the possibility of lost connectivity is too overwhelming to ignore. Determining that the node in question is compromised just because there was no response from it when one was expected would be fallacious. When no response is received, we treat this as a misincident and another HELLO packet is sent to the node after some time. This is done a few times to make sure that the node is not responding due to lost connectivity. No response at all even after multiple attempts leads to identifying a non-cooperating compromised node.

4.5.2 Replay Attack

A malicious node can generate false outdated routing updates in an attempt to have nodes update their routing tables with stale information, which can have a propagating effect throughout the network. A mis-incident in this attack would be an outdated routing update received from a particular node and when the threshold is exceeded within a time frame, we can accurately identify the node as one attempting to unnecessarily send outdated routing updates.

4.5.3 Denial-of-Service (DoS)

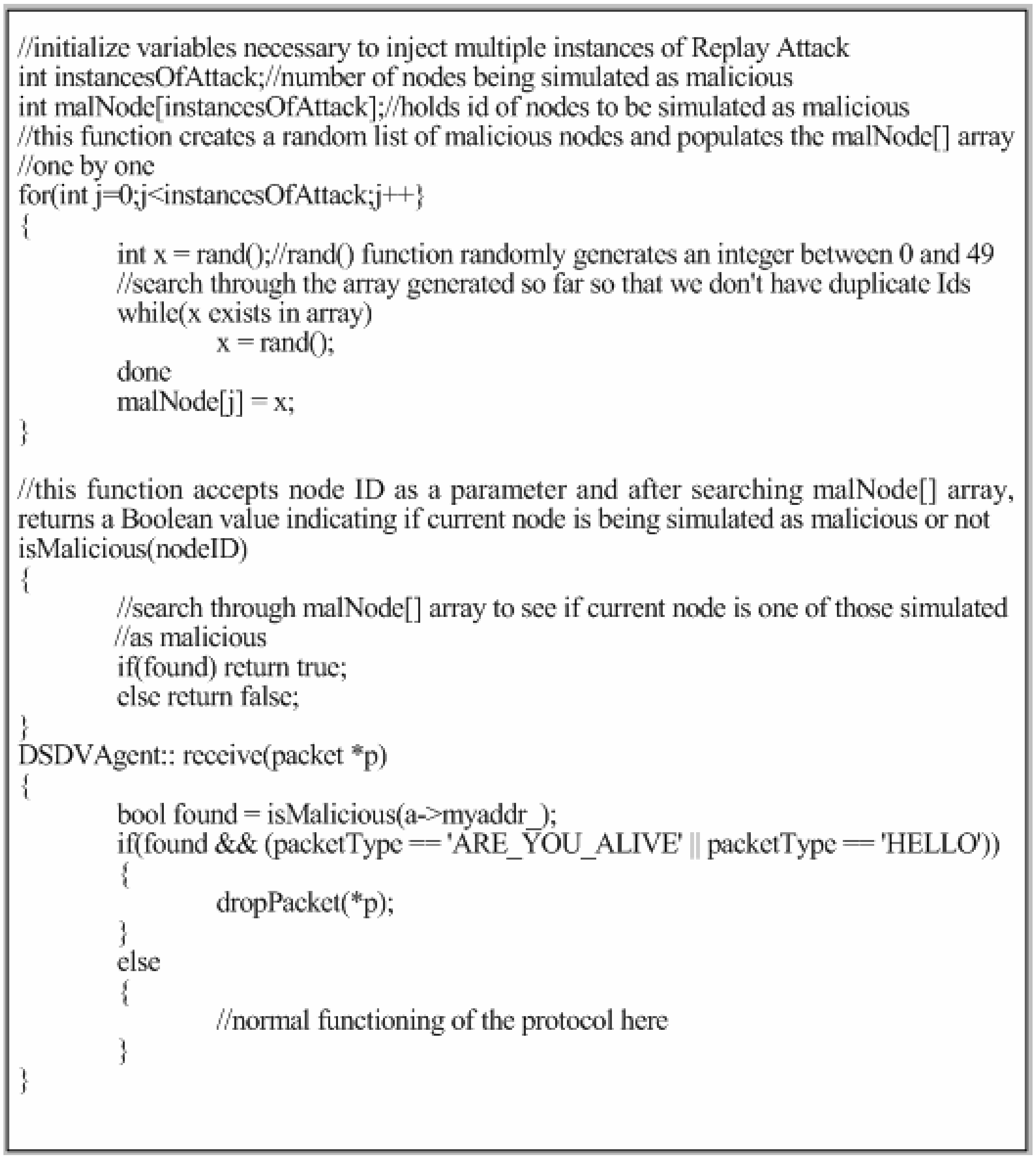
Intentionally and unnecessarily sent broadcast messages would result in a DoS experienced by the end nodes leading to a non-availability of network resources. When the number of broadcast packets originating from a particular node exceeds a threshold within a time frame, we can accurately identify the node as one launching a DoS attack.

4.6 Injection of Attacks into the network

In order to detect any kind of attack in a simulated environment, we need to inject those attacks. In this section, we discuss how each of the attacks were injected into the network along with a brief description of the algorithm used for the same.

4.6.1 Compromised node

The simplest way in which a compromised node can have an adverse effect on the MANET is by not cooperating with other nodes. This can have a damaging effect since the whole idea behind the MANET approach is a fully-distributed architecture involving complete cooperation among mobile nodes. The DSDV code is modified such that certain nodes do not respond to HELLO/ARE_YOU_ALIVE kind of packets from other nodes in the network, which want to make sure that the destination node is up and running. Under normal circumstances, a fully cooperative node would send in a response back to the sender on the lines of ICMP packets sent in ‘ping’. We can inject a compromised node by having it to drop any such HELLO/ARE_YOU_ALIVE packet and not send in an expected response. The pseudo code for injecting such compromised nodes in shown in Figure 4.4.



```
//initialize variables necessary to inject multiple instances of Replay Attack
int instancesOfAttack;//number of nodes being simulated as malicious
int malNode[instancesOfAttack];//holds id of nodes to be simulated as malicious
//this function creates a random list of malicious nodes and populates the malNode[] array
//one by one
for(int j=0;j<instancesOfAttack;j++}
{
        int x = rand();//rand() function randomly generates an integer between 0 and 49
        //search through the array generated so far so that we don't have duplicate Ids
        while(x exists in array)
                x = rand();
        done
        malNode[j] = x;
}

//this function accepts node ID as a parameter and after searching malNode[] array,
returns a Boolean value indicating if current node is being simulated as malicious or not
isMalicious(nodeID)
{
        //search through malNode[] array to see if current node is one of those simulated
        //as malicious
        if(found) return true;
        else return false;
}
DSDVAgent:: receive(packet *p)
{
        bool found = isMalicious(a->myaddr_);
        if(found && (packetType == 'ARE_YOU_ALIVE' || packetType == 'HELLO'))
        {
                dropPacket(*p);
        }
        else
        {
                //normal functioning of the protocol here
        }
}
```

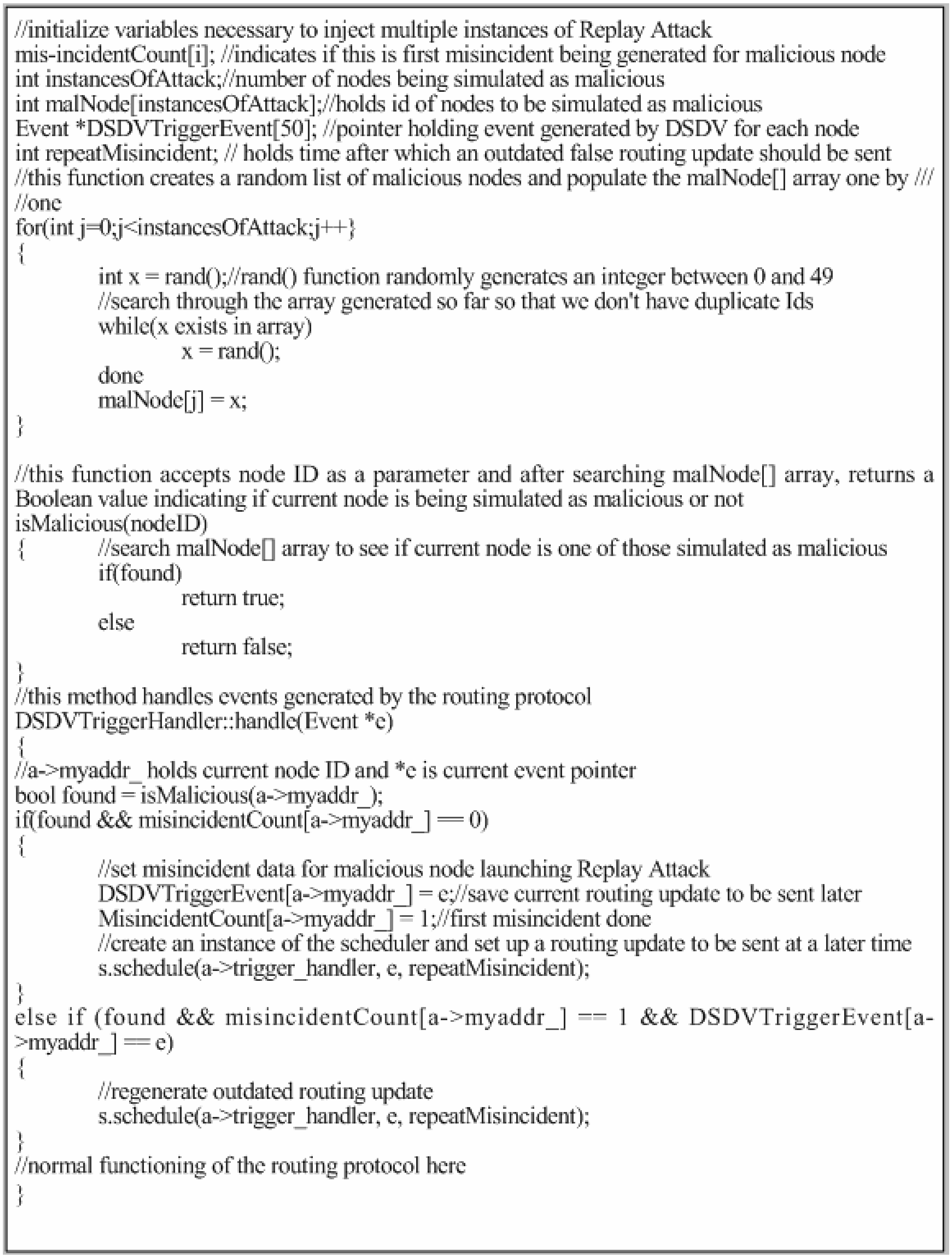
Figure 4.4: Injecting Compromised nodes into the network

4.6.1.1 Description of the algorithm

Variables utilized for the injection of compromised nodes are initialized. These include the node ids being simulated as malicious. A random number generator function is utilized to generate a random list of nodes to be simulated as malicious and they are stored in an array. A check is done by searching the array list to find out if the current node is a malicious node. If not, then the node sends in a response indicating that it is a fully functional cooperating node. A compromised node, on the other hand, drops all packets that evoke a response to be sent in to the sender. This is done for all such compromised nodes every time a ‘ping’ like packet is received.

4.6.2 Replay Attack

The working of a mobile ad hoc network depends entirely on the routing protocol and, in turn, relies on the routing updates. Incorrect routing information, for instance, in the form of outdated routing updates can lead to incorrect functioning of the MANET. In order to inject such an attack, we have worked around the DSDV routing protocol code and created malicious nodes, which generate false outdated routing updates in an attempt to have nodes updates their routing tables with stale information. This can have a drastic propagating effect throughout the network. . The rate at which misincidents are generated by each of the malicious nodes is picked up from a set of increasing values such that each node generates misincidents at different rate. This has the effect of providing a tougher challenge for the IDS to detect malicious nodes each launching Replay Attack at different flooding rates as compared to, say, malicious nodes each launching Replay Attack at the rate flooding rate. The pseudo code for the process of injecting multiple instances of replay attack is shown in Figure 4.5.



```
//initialize variables necessary to inject multiple instances of Replay Attack
mis-incidentCount[i]; //indicates if this is first misincident being generated for malicious node
int instancesOfAttack;//number of nodes being simulated as malicious
int malNode[instancesOfAttack];//holds id of nodes to be simulated as malicious
Event *DSDVTriggerEvent[50]; //pointer holding event generated by DSDV for each node
int repeatMisincident; // holds time after which an outdated false routing update should be sent
//this function creates a random list of malicious nodes and populate the malNode[] array one by ///
//one
for(int j=0;j<instancesOfAttack;j++}
{
            int x = rand();//rand() function randomly generates an integer between 0 and 49
            //search through the array generated so far so that we don't have duplicate Ids
            while(x exists in array)
                        x = rand();
            done
            malNode[j] = x;
}

//this function accepts node ID as a parameter and after searching malNode[] array, returns a
Boolean value indicating if current node is being simulated as malicious or not
isMalicious(nodeID)
{           //search malNode[] array to see if current node is one of those simulated as malicious
            if(found)
                        return true;
            else
                        return false;
}
//this method handles events generated by the routing protocol
DSDVTriggerHandler::handle(Event *e)
{
//a->myaddr_ holds current node ID and *e is current event pointer
bool found = isMalicious(a->myaddr_);
if(found && misincidentCount[a->myaddr_] == 0)
{
            //set misincident data for malicious node launching Replay Attack
            DSDVTriggerEvent[a->myaddr_] = e;//save current routing update to be sent later
            MisincidentCount[a->myaddr_] = 1;//first misincident done
            //create an instance of the scheduler and set up a routing update to be sent at a later time
            s.schedule(a->trigger_handler, e, repeatMisincident);
}
else if (found && misincidentCount[a->myaddr_] == 1 && DSDVTriggerEvent[a-
>myaddr_] == e)
{
            //regenerate outdated routing update
            s.schedule(a->trigger_handler, e, repeatMisincident);
}
//normal functioning of the routing protocol here
}
```

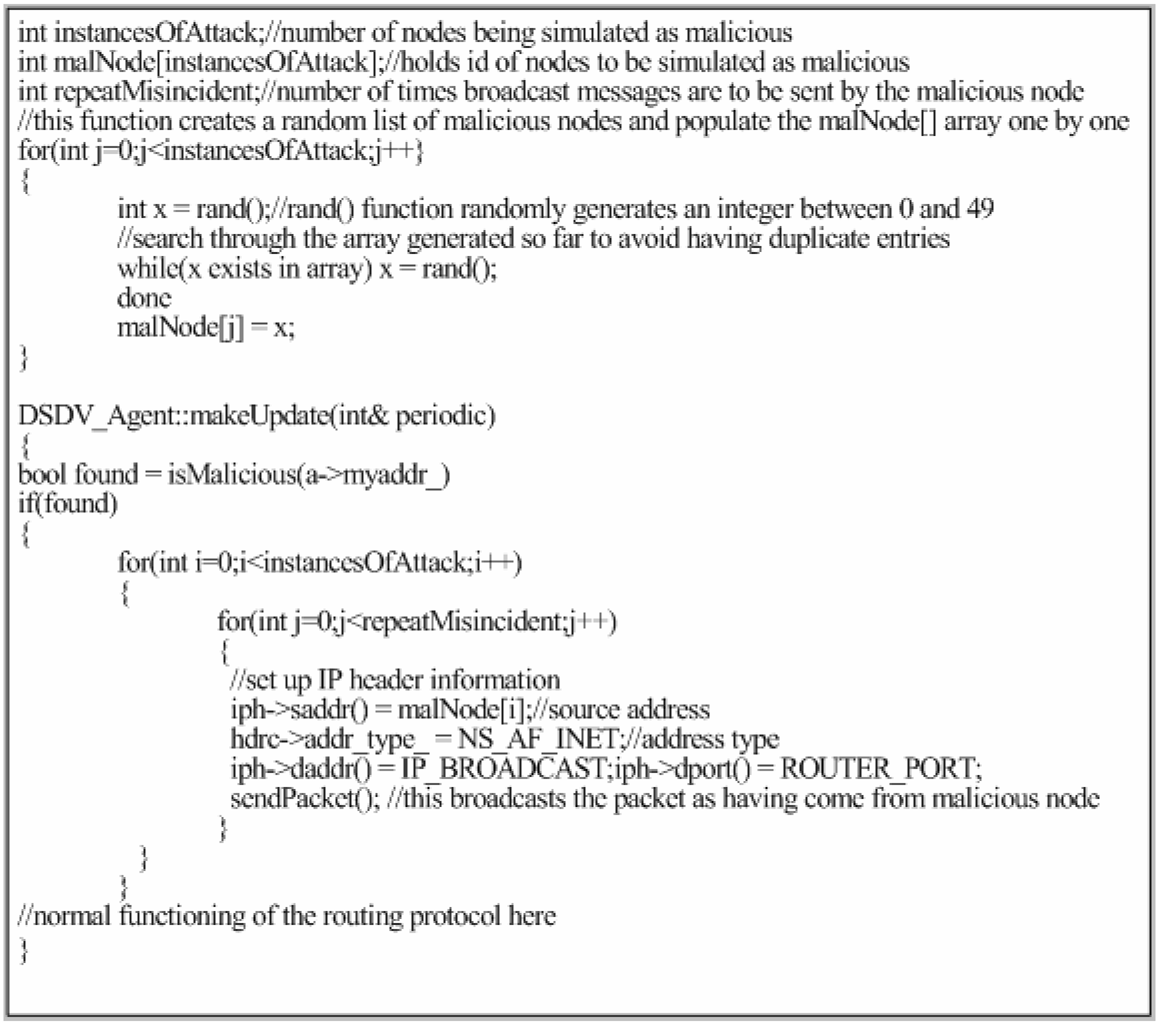
Figure 4.5: Injecting Replay Attack into the Network

4.6.2.1 Description of the algorithm

Variables utilized for the injection of Replay attack are initialized. These include a flag for the first misincident being generated for a malicious node, the nodes being simulated as malicious, pointers holding routing update events and a variable for the time after which an outdated false routing update should be sent. A random number generator function is utilized to generate a random list of nodes to be simulated as malicious and they are stored in an array. A check is done by searching the array list to find out if the current node is a malicious node. If yes, then the current routing update for each of the malicious nodes is stored and replayed back at a later time by creating an instance of the scheduler.

4.6.3 Denial-of-Service (DoS)

Intentionally and unnecessarily sent broadcast messages would result in a DoS experienced by the end nodes leading to a non-availability of network resources. This attack has been injected by having a node to send broadcast packets to its neighbors. The rate at which misincidents are generated by each of the malicious nodes is picked up from a set of increasing values such that each node generates misincidents at different rates. This has the effect of providing a tougher challenge for the IDS to detect malicious nodes, each launching DoS at different flooding rates as compared to, say, malicious nodes each launching DoS at the same flooding rate. The pseudo code for the process of injecting a DoS attack is shown below in Figure 4.6.



```
int instancesOfAttack;//number of nodes being simulated as malicious
int malNode[instancesOfAttack];//holds id of nodes to be simulated as malicious
int repeatMisincident;//number of times broadcast messages are to be sent by the malicious node
//this function creates a random list of malicious nodes and populate the malNode[] array one by one
for(int j=0;j<instancesOfAttack;j++}
{
        int x = rand();//rand() function randomly generates an integer between 0 and 49
        //search through the array generated so far to avoid having duplicate entries
        while(x exists in array) x = rand();
        done
        malNode[j] = x;
}

DSDV_Agent::makeUpdate(int& periodic)
{
bool found = isMalicious(a->myaddr_)
if(found)
{
        for(int i=0;i<instancesOfAttack;i++)
        {
                for(int j=0;j<repeatMisincident;j++)
                {
                 //set up IP header information
                 iph->saddr() = malNode[i];//source address
                 hdrc->addr_type_ = NS_AF_INET;//address type
                 iph->daddr() = IP_BROADCAST;iph->dport() = ROUTER_PORT;
                 sendPacket(); //this broadcasts the packet as having come from malicious node
                }
           }
        }
//normal functioning of the routing protocol here
}
```

Figure 4.6: Injecting Denial-of-Service Attack into the Network

4.6.3.1 Description of the algorithm

Variables utilized for the injection of DoS attack are initialized. These include the nodes in the network to be simulated as malicious, the number of misincidents to be generated, which in the case of DoS attack, are broadcast packets. A random number generator function is utilized to generate a random list of nodes to be simulated as malicious and they are stored in an array. A check is done by searching the array list to find out if the current node is a malicious node. If yes, then a broadcast packet is set up by setting the IP and common packet header information. The broadcast packets are then sent out continuously for each of the malicious nodes.

4.7 Chapter Summary

We discussed our proposed intrusion detection scheme for wireless mobile ad hoc networks based on the DSDV protocol. The proposed intrusion scheme aims to combine the best of misuse and anomaly detection and is based on a self-adjusting threshold scheme. We have focused on attacks such as Replay Attack, Denial-of-Service and Compromised Node, which have the potential to paralyze an ad hoc network. The generic conceptual approach behind the proposed scheme promises easy incorporation into other ad hoc routing protocols for MANETs.

5. Experimental Model

We have chosen the ns-2 simulator [ns] for this research because it realistically models arbitrary node mobility as well as physical radio propagation effects such as signal strength, interference, capture effect, and wireless propagation delay. The simulator also includes an accurate model of the IEEE 802.11 Distributed Coordination Function (DCF) wireless MAC protocol. The DSDV routing protocol for wireless ad hoc networks is available within ns-2.

5.1 Rationale behind choosing ns-2

The Network Simulator (NS2) is a discrete event simulator developed by the University of California at Berkeley and the VINT project [FaVa97]. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The Monarch research group at Carnegie-Mellon University [CMU] developed support for simulation of multihop wireless networks complete with physical, data link, and medium access control (MAC) layer models on NS2 [BrMaJoHuJe98]. It is open source, and can be downloaded from the Internet. It provides tools for generating data traffic and mobile node mobility scenario patterns for the simulation.

NS2 provides a split-programming model. The simulation kernel is implemented with a systems language (C++), Tcl scripting language is used to express the definition, configuration and the control of the simulation. This split-programming approach can benefit the research productivity. Also, NS2 can produce a detailed trace file and an animation file for each ad hoc network simulation that is very convenient for analyzing the routing behavior. The disadvantage of NS2 is that it is a large system with a relatively steep initial learning curve.

In NS2, the Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer protocol. The radio model uses characteristics similar to a commercial radio interface, Lucent's WaveLAN [Wavelan]. WaveLAN is modeled as a shared-media radio with nominal bit rate of 2Mb/s and a nominal radio range of 250 meters. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model. When a transmitter is within the reference distance of the receiver (denoted by r), the free space model (where the signal attenuates as $1/r^2$) is used. Outside of this distance, the ground reflection model (where the signal falls off as $1/r^4$) is used. Here is a summary for the implementation of wireless networks in NS2:

- Mac Layer: IEEE 802.11
- Address Resolution Protocol (ARP)
- Ad hoc routing protocols: DSDV, DSR, TORA, AODV
- Radio Propagation Model
  - Friss-space attenuation at near distances
  - Two-ray ground at far distance
- Antenna: an omni-directional antenna having unity gain

5.2 Simulation Setup

The overall goal of the simulation experiments is to measure the accuracy and robustness of our intrusion detection scheme for wireless mobile ad hoc networks while continuing to successfully deliver data packets to their destinations. To measure this ability, a variety of workloads were applied to the simulated network, including node movement, data traffic patterns, node density and varying percentages of malicious nodes.

5.2.1 Movement Space

In order to accurately assess the effect of node density, we have two different movement space area scenarios. More specifically, we have an ad hoc network in our simulation with a 670m x 670m flat space with 50 nodes in it. We then increase the flat space area by 25% while keeping the number of mobile nodes the same such that it is now a 837x837 meter flat space with 50 mobile nodes in it.

The Medium Access Control (MAC) layer used for this research is IEEE 802.11 and the transport layer used is User Datagram Protocol (UDP), both of which are available within ns-2. Above the physical layer, the 802.11 MAC layer is implemented with the carrier sense, RTS/CTS, and back-off mechanisms. The wireless channel has a radio propagation range of 250 meters and capacity of approximately 2 Mb/s.

5.2.2 Movement Model

In our simulations, nodes move according to the random waypoint mobility model [JoMa96]. In the random waypoint model, each node remains stationary for a predefined pause time. Then a destination in flat area is randomly generated and the node moves to the generated destination at a speed uniformly distributed between 0 and the predefined maximum speed (20m/s in our simulation). Once the node reaches the destination, the node pauses again for the predefined pause time, chooses another destination, and moves there. When the nodes reach the edge of the flat area, they bounce back and move along the opposite directions at the same speed. Each node repeats this behavior during the simulation. We ran each of the simulations for 1000 seconds. The movement patterns are generated for 11 different pause times: 0, 100, …, 1000 seconds. Assuming that the mobility of the ad-hoc networks is proportional to the pause time, we simulated the mobility by use of pause time. The longer the pause time, the lesser the mobility. In case of 1000 seconds pause time, the ad-hoc network is essentially static. If the pause time is 0 seconds, every node moves around without any pause and the resulting mobile network is highly dynamic.

5.2.3 Communication Model

The traffic patterns consist of 20 constant bit rate (CBR) flows that send packets at a rate of 4 packets per second with packet size of 512 bytes. Since TCP sources can dynamically adjust the sending rates based on the network's traffic condition by using flow control and congestion control mechanisms, we do not use TCP sources. CBR sources can provide a uniform traffic load, which is based on the beginning time and the number of packets, so that we can directly compare the network characteristics under different routing protocols and the same traffic load. The reason why we do not choose TCP sources is that TCP adapts to the load of the network. For the same data traffic and node movement scenario, the time when a node sends a packet will be different if TCP is used, for simulations using different protocols or different versions of the same protocol. Then, it is difficult to compare the performance between different protocols and between the original and modified protocols. Table 5.1 details the parameters to be used in our simulations.

Number of nodes	50
Maximum velocity v_{max}	20 m/s
Dimensions of test bed	670m x 670m, 837m x 837m
CBR Flows	20
Individual Source Data Rate	4 packets/second
Application Data Payload Size	512 bytes/packet
Channel Capacity	2 Mbps
Nominal Radio Range	250m

Table 5.1: Parameter specifications used for simulations.

5.2.4 Simulation Cases

For each of the 3 types of attacks, namely, Denial-of-Service, Replay Attack and Compromised Nodes, we had two movement spaces - one with flat meter space 670mx670m and the other with 837mx837m (i.e. 25% more) with the same number of mobile nodes in it (i.e. 50 nodes). For each of these two movement spaces for each of the three attacks, we had 4 different percentages of malicious nodes in the networks (i.e. 5%, 10%, 15% and 20% of the total number of mobile nodes in the network). For each for these scenarios, we ran eleven 1000-second simulations starting with 0 seconds as the pause time through to 1000 seconds as the pause time in 100-second increments. For each pause time, we calculated the average of values over 75 simulation runs. Figure 5.1 illustrates the simulation scenarios.

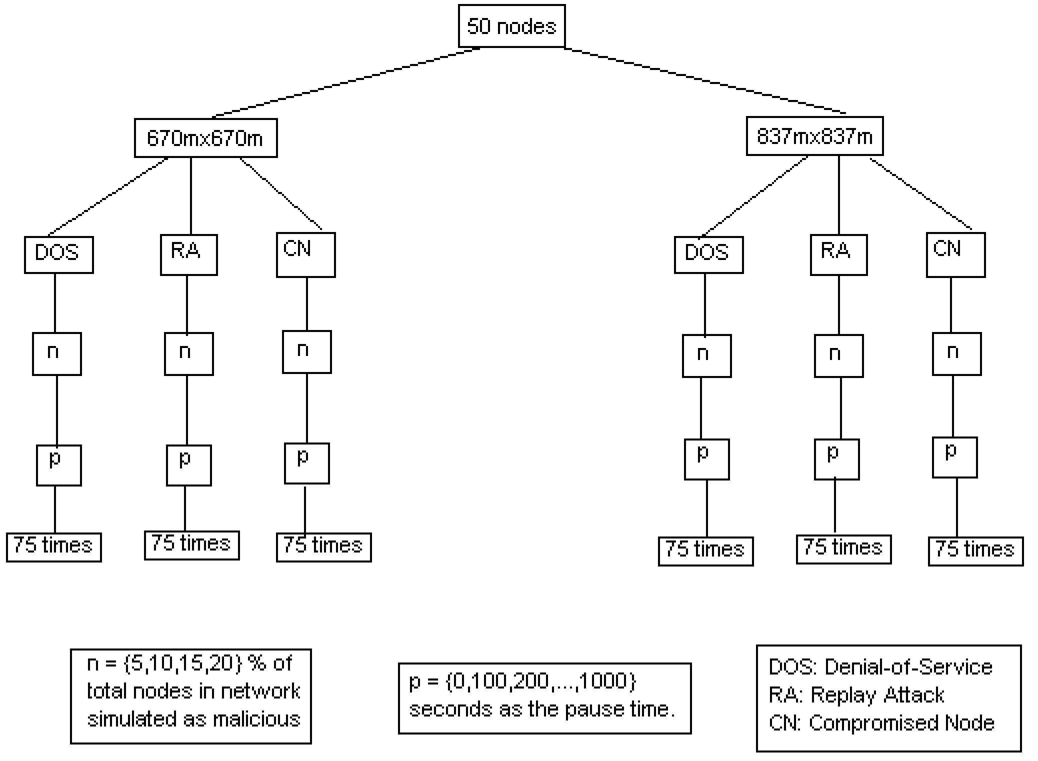

50 nodes
670mx670m
837mx837m
DOS
RA
CN
n
p
75 times
n = {5,10,15,20} % of total nodes in network simulated as malicious
p = {0,100,200,...,1000} seconds as the pause time.
DOS: Denial-of-Service
RA: Replay Attack
CN: Compromised Node

Figure 5.1: Simulation Scenarios.

5.3 Performance Metrics

To evaluate the performance of the routing protocols, the following three metrics are used:

5.3.1 Average End-to-End Delay

This is defined as the average of all end-to-end delays for all packets, which are successfully delivered. This includes all possible delays caused by buffering during routing update delay processing, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times.

5.3.2 Packet Delivery Ratio

Packet delivery ratio (PDR) is defined as the percentage of total not dropped CBR packets.

$$PDR = \frac{TotalGeneratedCBRPackets - TotalDroppedDataPackets}{TotalGeneratedCBRPackets} x100$$

It is also defined as the ratio between the numbers of packets originated by CBR sources and the number of packets received by the CBR sink at the final destination. The difference between the two definitions is whether we count the data packets in the queue at the end of the simulation.

5.3.3 Normalized Routing Load

This is defined as the number of control packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission. It is a better metric than absolute number of routing packets, because it shows the relationship between routing packets and CBR packets.

5.4 Chapter Summary

This chapter focused on the simulation methodology and details about our experimental model. We have used ns-2 as the simulation tool for validating our proposed intrusion detection scheme because it is a favored tool for wireless network simulations in the research community. In the next chapter, we provide details about our findings and discuss in depth as to how accurate, effective and robust our scheme is and what effect, factors such as node density, node movement, etc. have on our scheme.

6. Discussion of Results

In this chapter, we analyze our intrusion detection scheme. Primarily, we have focused on the accuracy of our scheme in the face of various attacks i.e., whether malicious nodes are being detected and what is the accuracy of detection. One primary concern would be false positives or false alarms. Ideally, the percentage of such false alarms should be zero. A MANET being resource and bandwidth constrained, a security scheme for ad hoc networks cannot afford to hog bandwidth and other network resources. In effect, a security scheme needs to be efficient as far as performance is concerned. We have evaluated our intrusion detection scheme by taking into consideration the effect of three metrics: Average End-to-End Delay, Packet Delivery Ratio and Normalized Routing Load.

6.1 Accuracy of Detection

Of the 50 nodes in our simulated network, a certain percentage of nodes misbehave i.e. certain nodes in the network have been simulated as malicious nodes. For instance, a malicious node might be a node that launches a Replay Attack on its own. We vary the percentage of these malicious nodes from 5% to 20% in 5% increments. While a network with 20% of the nodes being malicious nodes might seem a bit unrealistic, we have simulated it for the simple reason that it gives us an opportunity to observe the robustness of our intrusion detection scheme. We have studied the effect of various factors over an average of 75 simulations for each of our 1000-second simulations. The pause time has been varied in 100-second increments between the two ends of the node mobility spectrum i.e. a 1000-second pause time essentially simulated a static wireless network and a 0-second pause time essentially simulated a constantly mobile wireless network. This was done to study the behavior of our intrusion detection under challenging conditions even though some of the scenarios might not be entirely practical.

In our simulations, we have increased the flat node movement space by 25% in order to analyze the effect of node density on our intrusion detection scheme. If more nodes are packed together, more nodes are in contention if they want to unicast. As such, we have two graphs for each kind of attack simulated; one with area 670mx670m and the other with area 837mx837m. In each graph, the suffix to each "MalNodes" legend indicates the percentage of malicious nodes which instantiated that particular attack in the simulation runs. For instance, MalNodes15 for an accuracy of detection graph for Replay Attack indicates that 15% of the nodes in the network were simulated as malicious with each of these malicious nodes launching Replay Attack on its own. Moreover, for each of our performance metrics, we have indicated the percentage of malicious nodes in the network while measuring the metric. For example, a legend of PDR5 indicates that we have measured Packet Delivery Ratio with the percentage of malicious nodes set to 5% of the total number of nodes in the network.

When doing simulations, often, we either want to get absolutely repeatable (deterministic) results or different results each time. Each approach requires a different seeding method in ns-2. We have chosen to utilize *$defaultRNG* seed heuristic *n* wherein the seed is generated based on current time, *"n"* has no effect on the random sequence. This generates a different random number sequence each time the program is run. The different set of random numbers generated in each of the simulations increases the level of confidence and reliability of our results and give us sufficient reason to reach a statistical significant conclusion.

6.1.1 Compromised Node

Figures 6.1-6.2 plots the accuracy of detection taken as an average over 75 simulation runs for each pause time for compromised nodes. The percentage of malicious nodes in the network was increased from 5% to 20% of the total number of nodes in the network, in increments of 5%. We find that the average accuracy of detection is over 85% for each of the four cases. The case wherein the number of malicious nodes in the network is 20% of the total number of nodes seems to have lowest accuracy compared to the other cases. At pause times 300 and 800, we notice a sharp fall in the percentage of malicious nodes detected. This could possibly be because with so many nodes in the network simulated as compromised, a HELLO/ARE_YOU_ALIVE packet intended for one such compromised node is dropped by another compromised node acting as an intermediate hop instead of forwarding it onto the actual destination node. The detection scheme flags the intermediate node as compromised and does not detect the actual end node as compromised. We looked at the ± percentage change in accuracy of detection and based on the results, there is a 3.24% ± percentage change at 0-second pause time and 6.16% ± percentage change at 1000-second pause time, which is reasonable. With respect to mobility, there is a ± percentage change of around 6.64% in the percentage of malicious nodes detected as the node mobility varies from highly dynamic to static. When the node density is reduced by increasing the movement space by 25%, we notice a 4.95% ± percentage change in the percentage of malicious nodes detected.

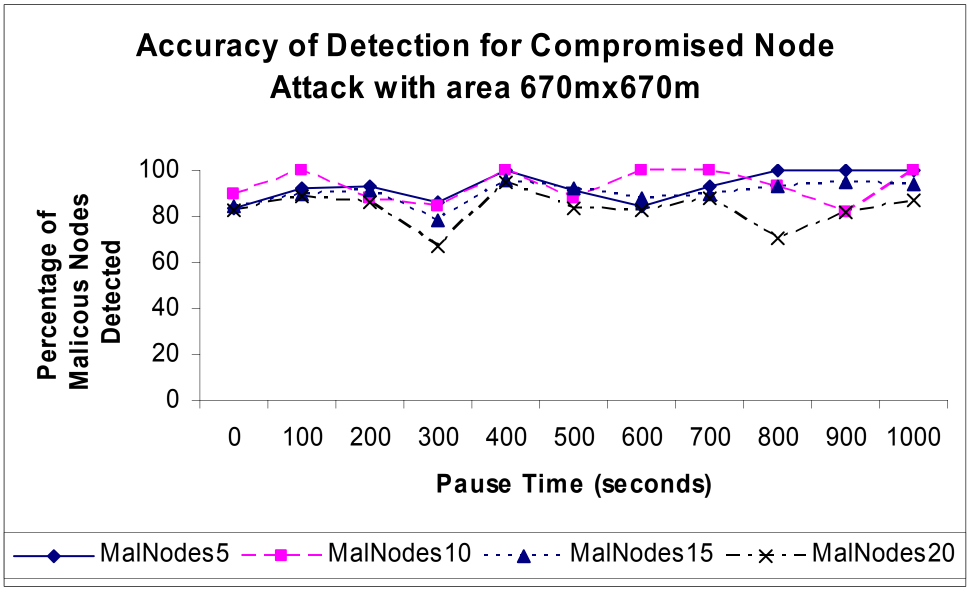


Figure 6.1: Accuracy of Detection for Compromised Node Attack with area 670mx670m

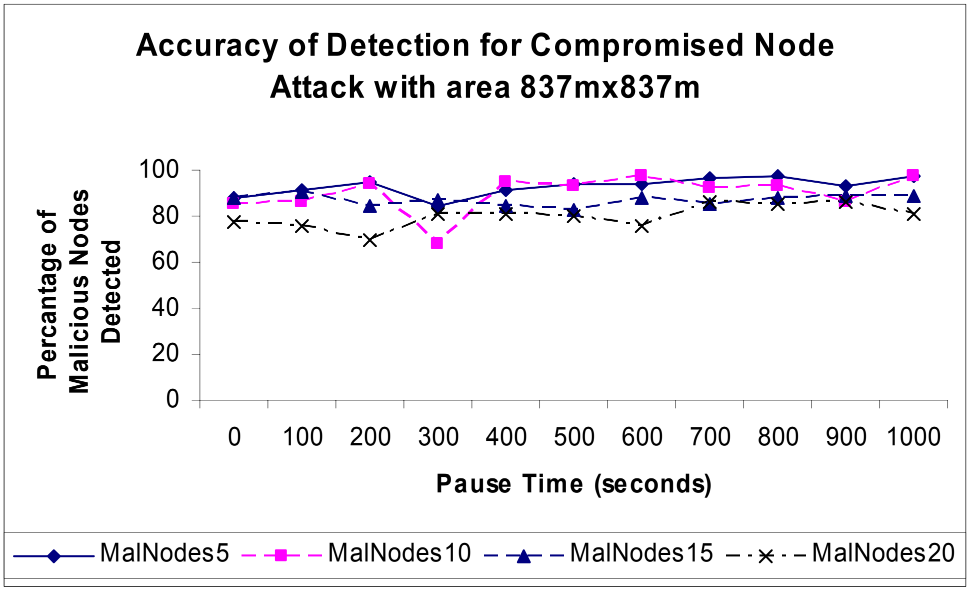


Figure 6.2: Accuracy of Detection for Compromised Node Attack with area 837mx837m

6.1.2 Replay Attack

Figures 6.3-6.4 plots the accuracy of detection taken as an average for each pause time for Replay Attack simulated while varying the percentage of malicious nodes in the network from 5% to 20% in increments of 5%. In contrast to Compromised Node Attack, Replay Attack has been detected with higher degree of accuracy and consistency. The node density does seem to affect marginally the accuracy of detection of compromised nodes. We find a 3% decrease in the overall average accuracy of detection across all pause times for the case with area 837mx837m. This could be because of the fact that with more flat meter movement space, there are more routing changes and hence more routing updates are sent among nodes. Routing buffer overflows could mean that some of these are dropped and affect the accuracy ever so slightly. The ± percentage change in the accuracy of detection hovers around the 1.15% mark as the node mobility decreases with movement space being 670mx670m. When the node density is reduced, the ± percentage change observed is 4.42% at 0 pause time and decreases gradually to 1.41% at 1000 pause time. The ± percentage change is around 1.94% as the percentage of malicious nodes in the network is changed from 5% to 20% for are 670mx670m and is around 2.76% when the area is increased by 25%.

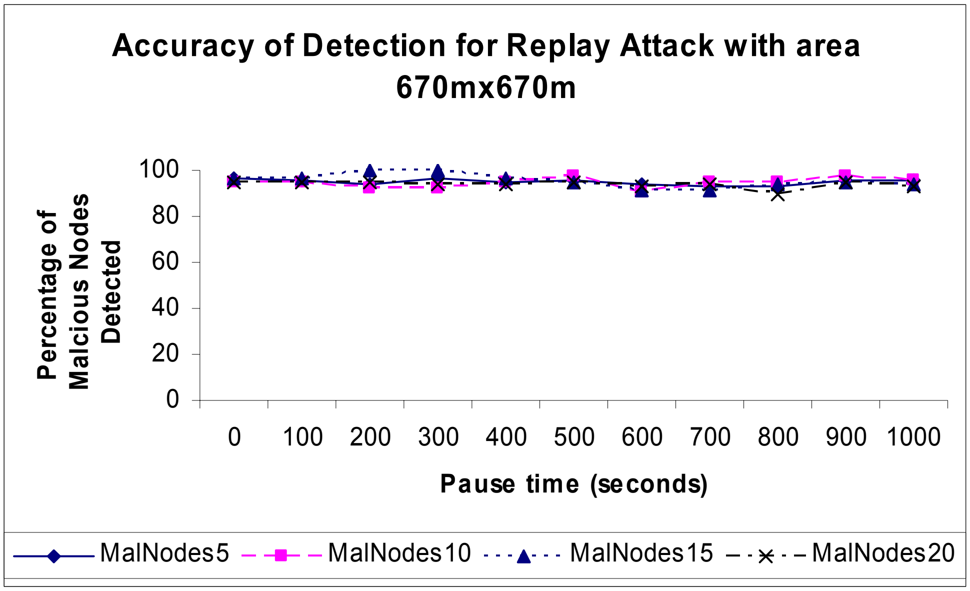


Figure 6.3: Accuracy of Detection for Replay Attack with area 670mx670m

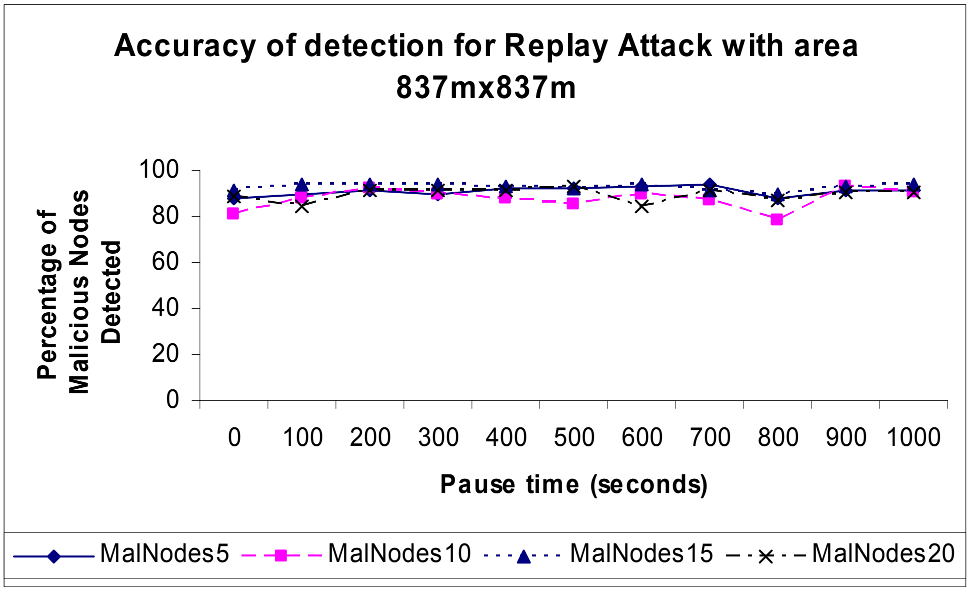


Figure 6.4: Accuracy of Detection for Replay Attack with area 837mx837m

6.1.3 Denial-of-Service Attack

Figures 6.5-6.6 plots the accuracy of detection taken as an average for each pause time for Denial-of-Service attack simulated while varying the percentage of malicious nodes in the network from 5% to 20% in increments of 5%. Like Replay Attack, Denial-of-Service Attacks have been detected with higher degree of accuracy and consistency notwithstanding the effect of node mobility, node density (seen by the effect of increasing the flat meter space by 25% while keeping the number of mobile nodes the same) and varying percentage of malicious nodes in the network. The average accuracy of detection over all pause times for a given percentage of malicious nodes in the network is close to 95% for DoS Attacks irrespective of the node density. When we look at the variation in the detection w.r.t node mobility, we observe a ± percentage change of around 3.78% at 0 pause time and decreases gradually to about 1.39% at 1000 pause time for 670mx670m as the node mobility decreases. As we decrease the node density, we observe an even lower ± percentage change in the accuracy. As the node mobility decreases with lower node density, the ± percentage change decreases from 2.2% to 0.6%. The variation in detection accuracy as the percentage of malicious nodes is increased from 5% to 20% is reflected in the ± percentage change which is around 1.98% with area 670mx670m and is about 2.3% as the node density is decreased.

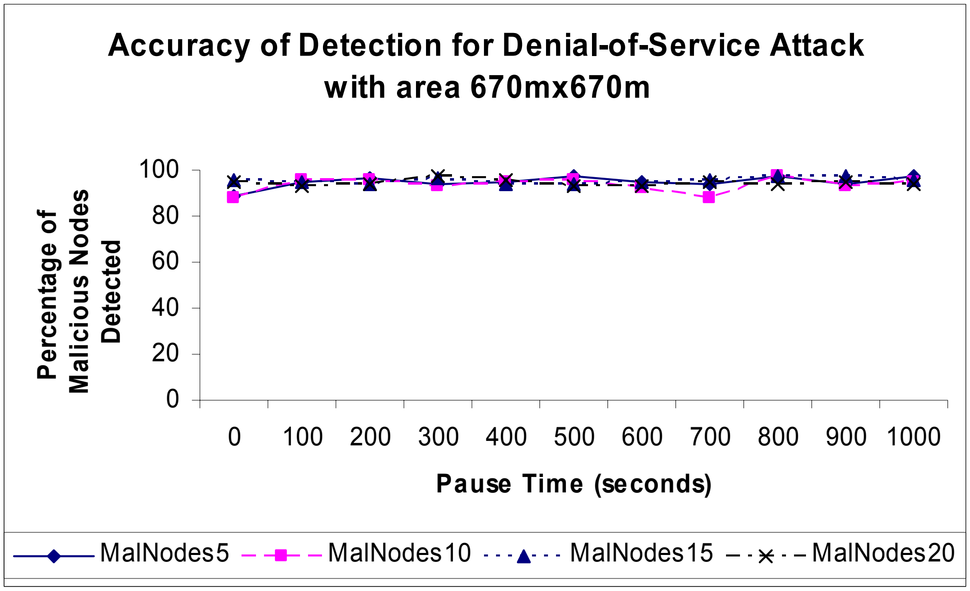


Figure 6.5: Accuracy of Detection for Denial-of-Service Attack with area 670mx670m

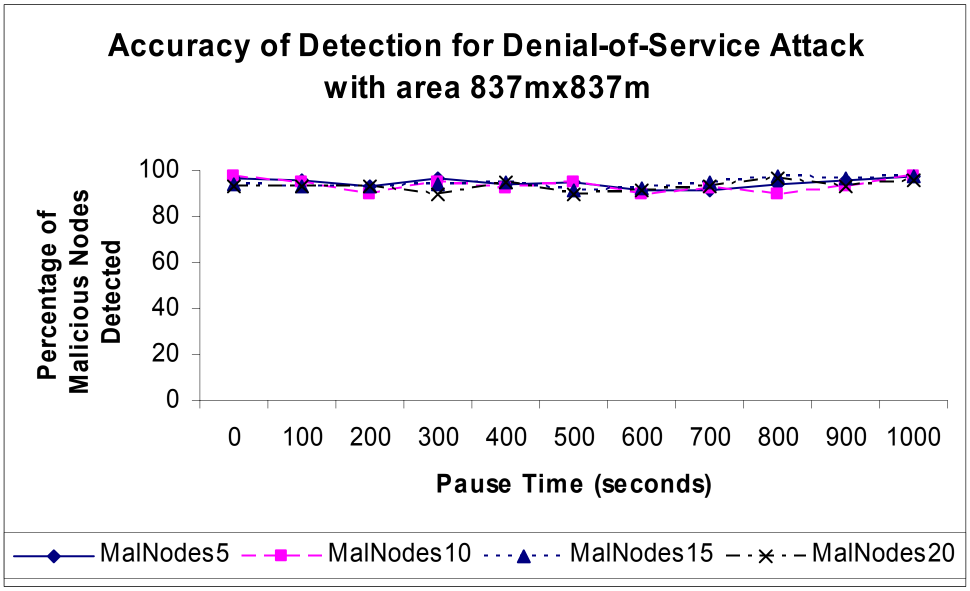


Figure 6.6: Accuracy of Detection for Denial-of-Service Attack with area 837mx837m

6.1.4 Average Accuracy of Detection

Figure 6.7 plots the average accuracy of detection across all pause times for a specific percentage of malicious nodes in the network for each of the attacks simulated. The average accuracy of detection over all pause times for a given percentage of malicious nodes in the network is close to 95% for Replay and DoS Attacks irrespective of the node density. However, the average accuracy of detection for Compromised Node Attack is around 89% with area 670mx670m and around 87% with area 837mx837m. The increased movement space means that mobile nodes have more potential to go out of listening range of other mobile nodes leading to inconclusive evidence of compromised nodes. Moreover, we find that as the number of malicious nodes in the network increases from 5% to 20%, the accuracy of detection decreases marginally (about 3% decrease). This is possibly because the length of simulation period is not long enough to detect all of the malicious nodes simulated as compromised nodes in the network.

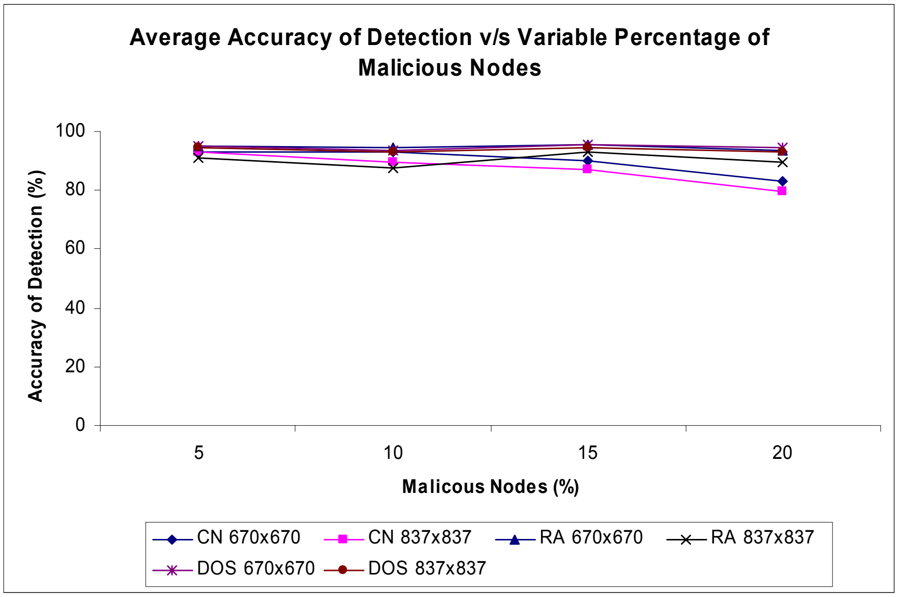


Figure 6.7: Accuracy of Detection v/s Variable Percentage of Malicious Nodes

6.2 Effect of False Positives

In this section, we discuss the effect of false positives on the network. False positives occur when our intrusion detection scheme reports that a node is misbehaving (i.e., it is a malicious node) when in fact it is not. Our simulations of Compromised Node and Denial-of-Service Attacks did not result in any false positives. Even though Replay Attack did result in false positives as shown in Figures 6.8 and 6.9, the percentage of nodes incorrectly detected as malicious nodes was negligibly small (the average over all pause times is about 2% for a given percentage of malicious nodes in the network). The extremely high mobility (as seen by pause times closer to 0 seconds) does play its part in flagging incorrectly false positives. At high mobility, there is constant movement of the mobile nodes leading to constant routing updates. The humungous amount of routing updates can have negative effect on our intrusion detection scheme as we can see in the case of FalsePos5 through FalsePos20. The behavior shown by FalsePos20 at pause time of 900 seconds in the 837mx837m area case is odd because other simulation cases have shown negligible percentage of false positives while the network is more on the static side rather than the dynamic side. The routing updates of some innocent nodes could have been flagged as a possible replay attack by mistake. The variation in the false positives flagged is reflected in the ± percentage change which decreases from about 2.6% to 0% as the node mobility decreases with area 670mx670m. With area increased by 25% (i.e. with decrease the node density), the ± percentage change decreases from 1.29% to 0 with decrease in node mobility. With respect to the varying percentage of malicious nodes in the network, the ± percentage change in the false positives flagged by the IDs is around 1.12% and increases slightly to 1.53% with decrease in node density.

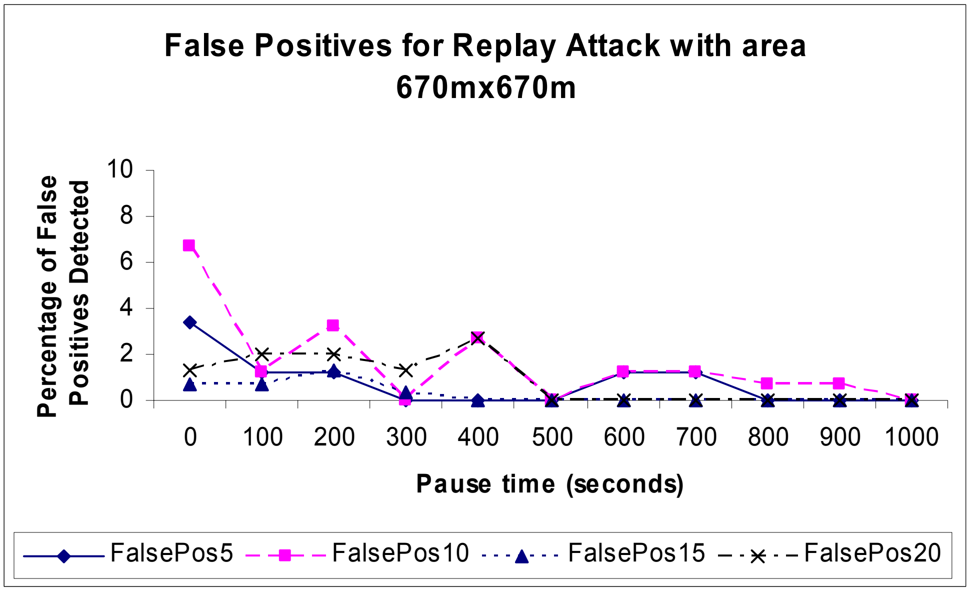


Figure 6.8: False Positives for Replay Attack with area 670mx670m

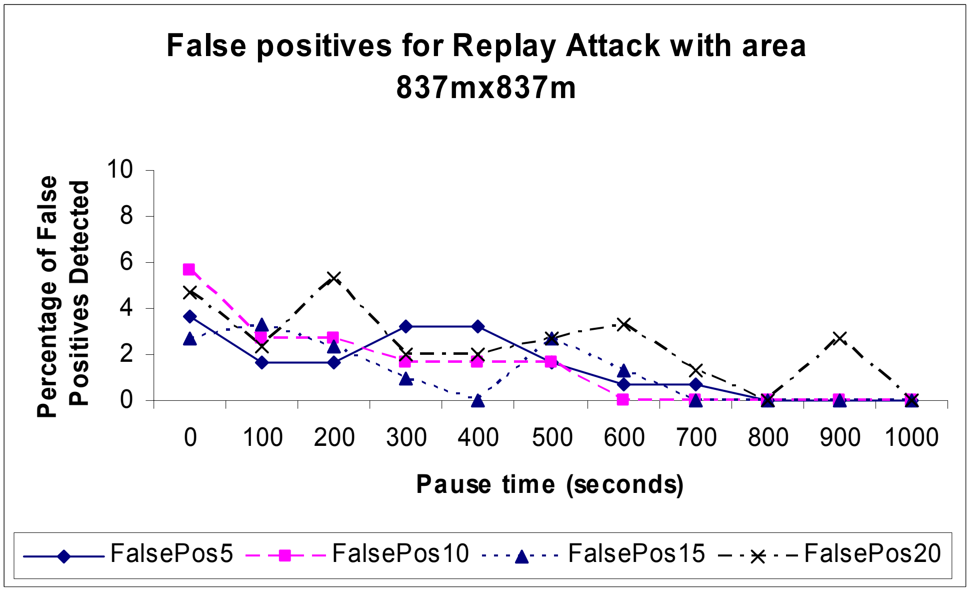


Figure 6.9: False Positives for Replay Attack with area 837mx837m

6.3 Performance

We have measured the performance of our intrusion detection scheme based on the evaluation of three critical metrics: Average End-to-End Delay, Packet Delivery Ratio and Normalized Routing Load. We have analyzed the effect of introducing our intrusion detection scheme into the MANET and whether it negatively affects the network performance as compared to the scenario with plain DSDV protocol.

6.3.1 Average End-to-End Delay

In this subsection, we present our simulation results of the average end-to-end delay. This performance metric, if affected greatly, can reflect negatively on the efficiency of a security scheme. Our simulation results are very promising.

6.3.1.1 Compromised Node

The average end-to-end delay measured for Compromised node scenarios remains fairly the same as that of the scenario without the introduction of our intrusion detection scheme as shown in Figures 6.10-6.11. However, we do see a lot of fluctuation in the measured values with flat meter space 837mx837m. With longer distances available for movement for each of the nodes, the time taken to transmit packets hop by hop tends to increase. This could be a possible explanation. Secondly, the higher mobility levels do play their part due to the constant movement of nodes and numerous intermediate hops which take more time to send a packet from source to destination.

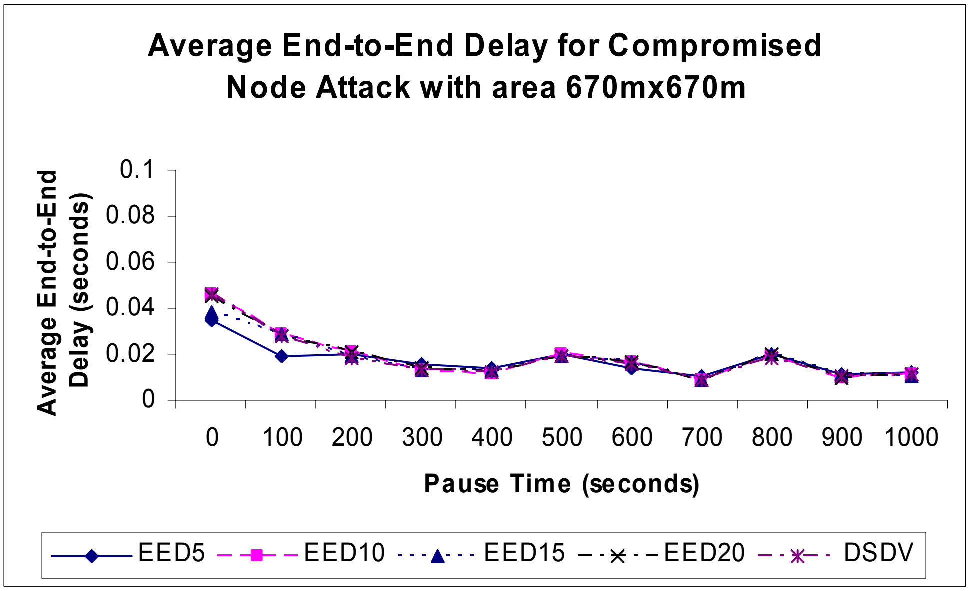


Figure 6.10: Average End-to-End Delay for Compromised Node Attack with area 670mx670m

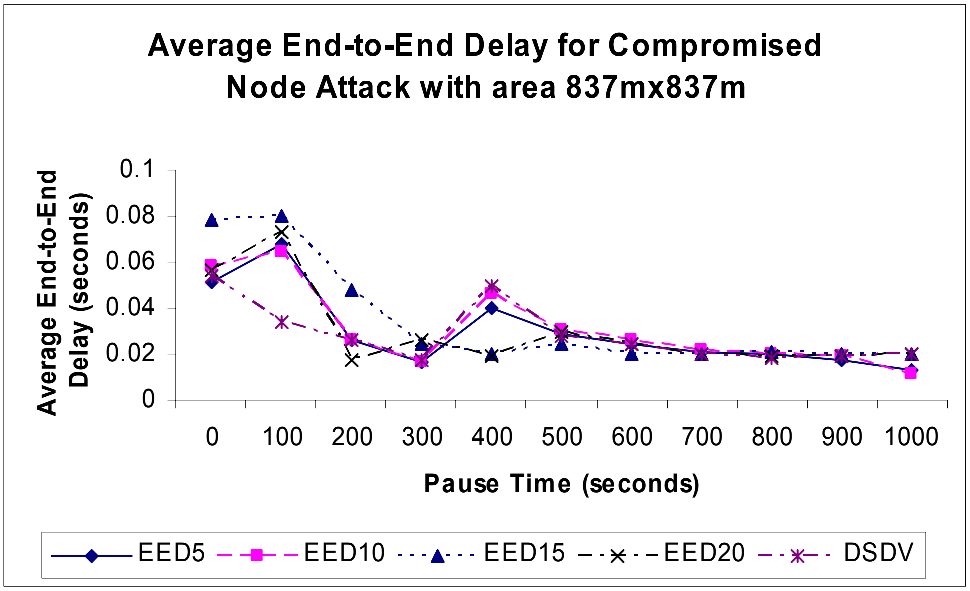


Figure 6.11: Average End-to-End Delay for Compromised Node Attack with area 837mx837m

6.3.1.2 Replay Attack

Figures 6.12-6.13 plot the average end-to-end delay versus the pause times as the network changes from static to highly dynamic state for Replay Attack. The simulation results closely match that of plain DSDV for each of the four malicious node percentage cases with the area 670mx670m. However, with the area increased by 25%, the measured values of average end-to-end delay are a lot lesser than that of the plain DSDV protocol in the mid stages of network mobility (see pause times 400-700). Interestingly, with reduced node density, the average end-to-end delay values are higher for plain DSDV than any of the scenarios with our IDS introduced into the network. This is strange because one would expect the delay to increase. It could be due to the fact that the plain DSDV behavior was observed without any instances of replay attack. It would be interesting to note the behavior of plain DSDV with increasing percentage of malicious nodes in the network with each node initiating replay attack.

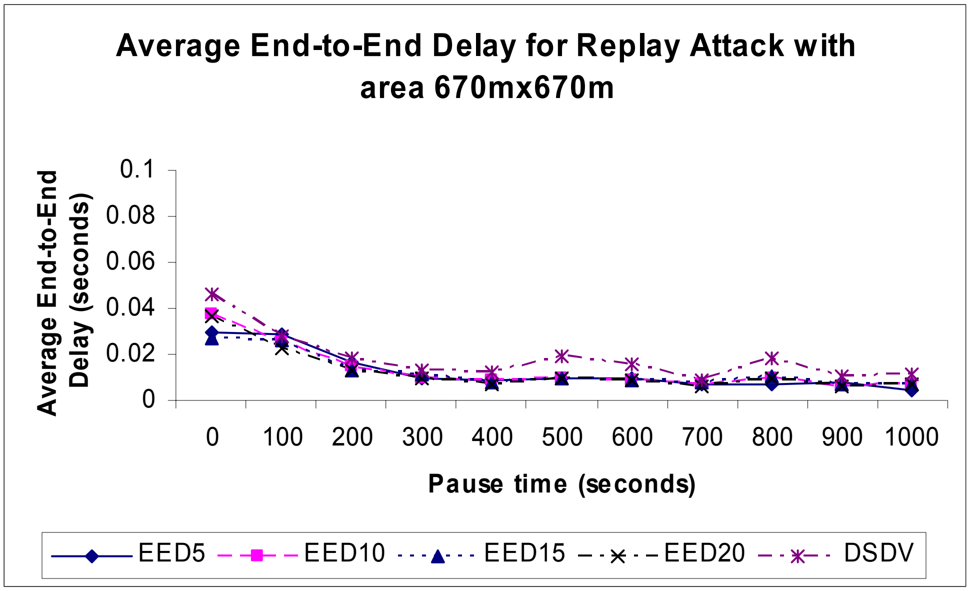


Figure 6.12: Average End-to-End Delay for Replay Attack with area 670mx670m

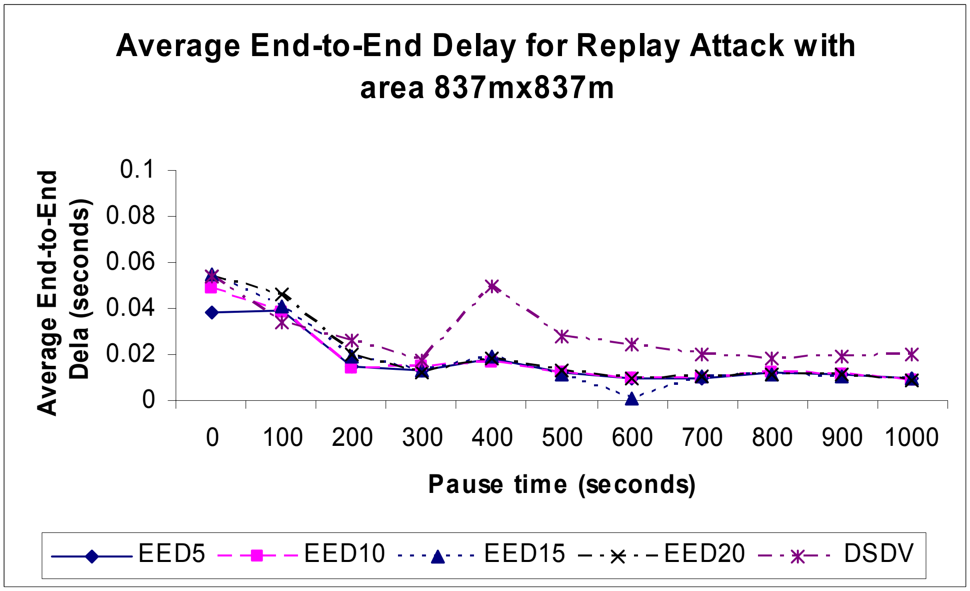


Figure 6.13: Average End-to-End Delay for Replay Attack with area 837mx837m

6.3.1.3 Denial-of-Service Attack

Figures 6.14-6.15 plot the average end-to-end delay versus the pause times as the network changes from static to highly dynamic state for Denial-of-Service Attack. The measured values closely match that of plain DSDV for each of the four malicious node percentage cases with the area 670mx670m as well as with area increased by 25%. These promising results indicate that the node density and node mobility do not negatively affect the average end-to-end delay for DoS Attack for any of the four malicious node percentage cases. At pause times 100 and 400 seconds, we observe an average difference of 0.023 seconds and 0.07 seconds (in both cases, it is an increase in average end-to-end delay with the introduction of our IDS) respectively compared to the average end-to-end delay for plain DSDV protocol.

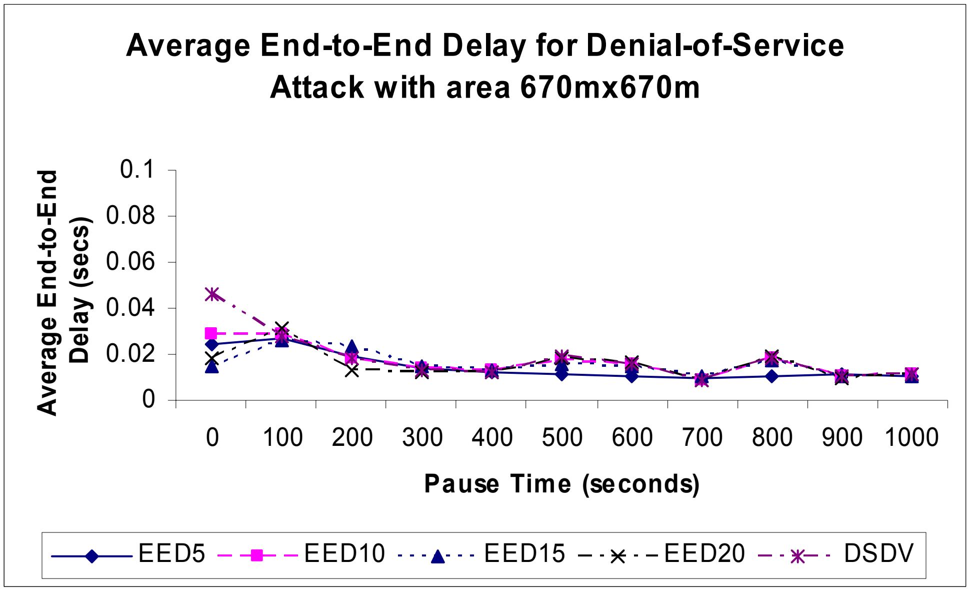


Figure 6.14: Average End-to-End Delay for Denial-of-Service Attack with area 670mx670m

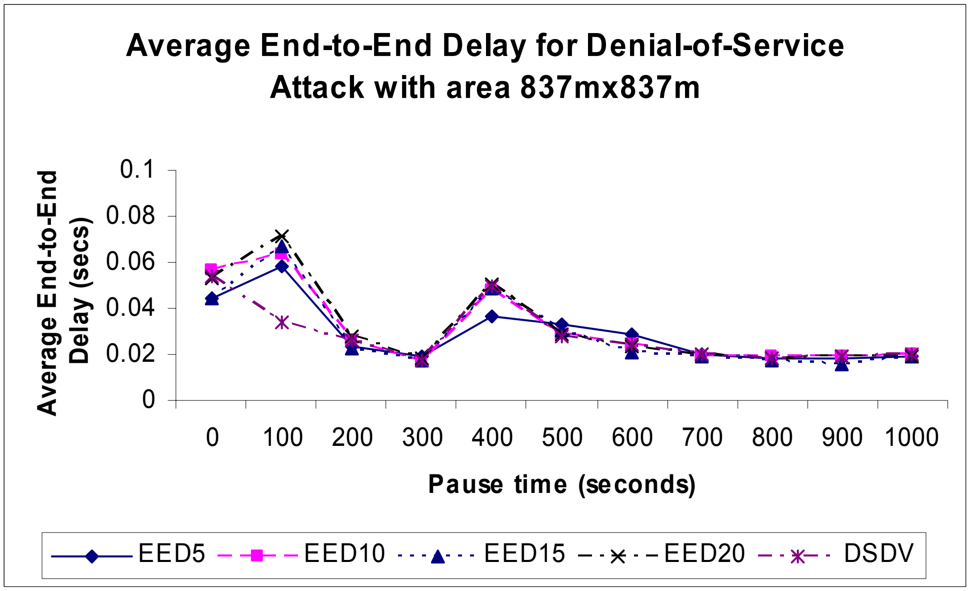


Figure 6.15: Average End-to-End Delay for Denial-of-Service Attack with area 837mx837m

6.3.1.4 Overall Average End-to-End Delay

Figure 6.16 plots the overall average end-to-end delay across all pause times for each of the attacks for each node density scenario. We have measured the overall average end-to-end delay for DSDV protocol with area 670mx670m and then with area increased by 25%. There is negligible difference in the overall average end-to-end delay measured for each of the attack scenarios with our intrusion detection scheme introduced.

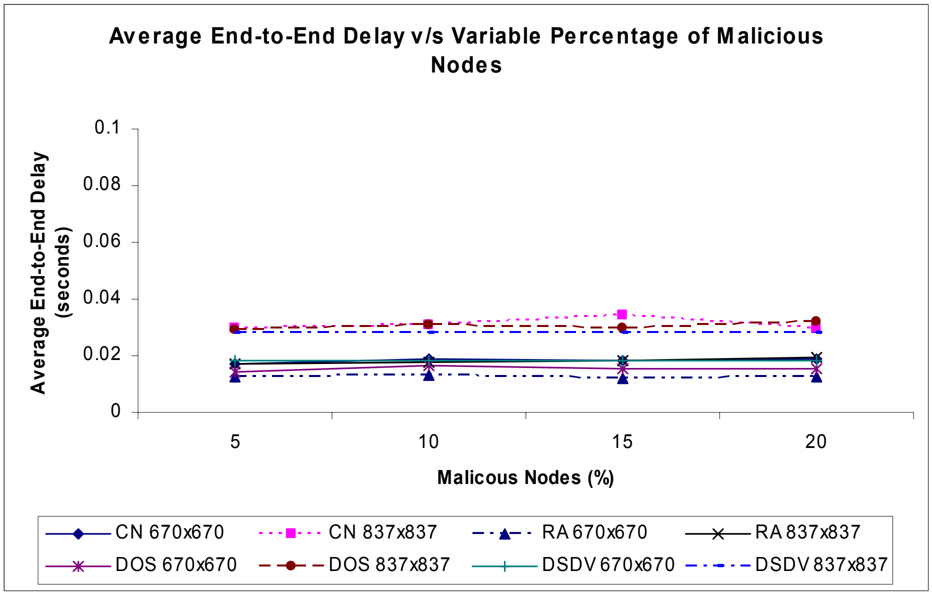


Figure 6.16: Average End-to-End Delay versus varying percentage of malicious nodes

6.3.2 Packet Delivery Ratio

In this subsection, we present the simulation results of Packet Delivery Ratio. This performance metric measures the network throughput by way of looking at the number of packets originated versus the number of packets received.

6.3.2.1 Compromised Node

Figures 6.17-6.18 plot the packet delivery ratio for compromised node scenarios with the pause time varied from 1000 seconds to 0 seconds in 100-second increments for each of the four malicious node percentage cases and with the node density varied as well. This metric's performance closely matches that of plain DSDV protocol with the number of malicious nodes in the network varied from 5% to 20%. With an increase in node mobility, the packet delivery ratio decreases by about 12%. With an increase in the movement space, i.e., with a corresponding decrease in node density, the packet delivery ratio behavior remains essentially the same. We observe a decrease in values by about 10% with increased node mobility. Network congestion could lead to a decrease in packet delivery ratio values with increase in node mobility and corresponding routing changes. If there is a network congestion because of high load, interface queues may get filled up. In this case, packets received, and packets which cannot be delivered for a long time, may get dropped.

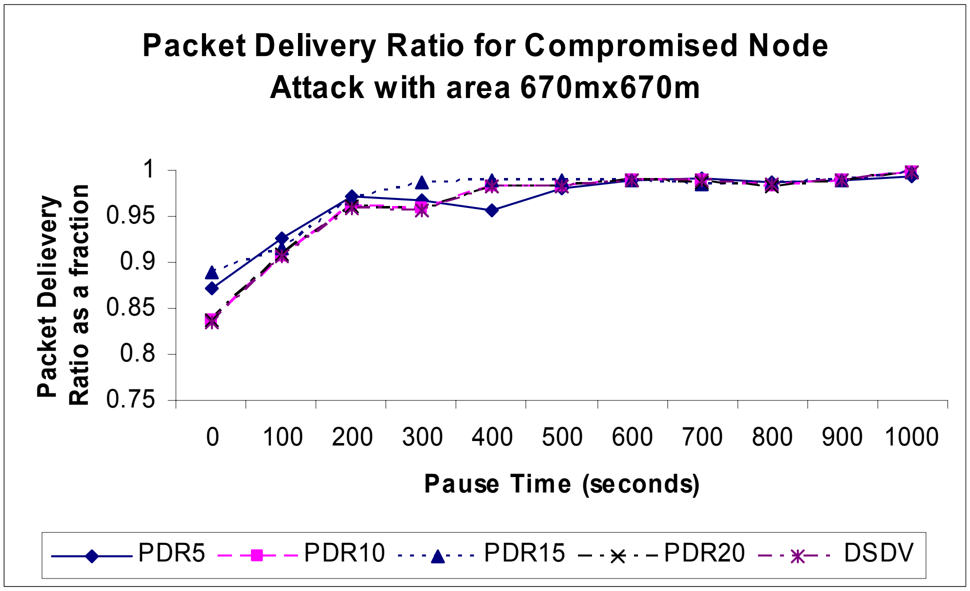


Figure 6.17: Packet Delivery Ratio for Compromised Node Attack with area 670mx670m

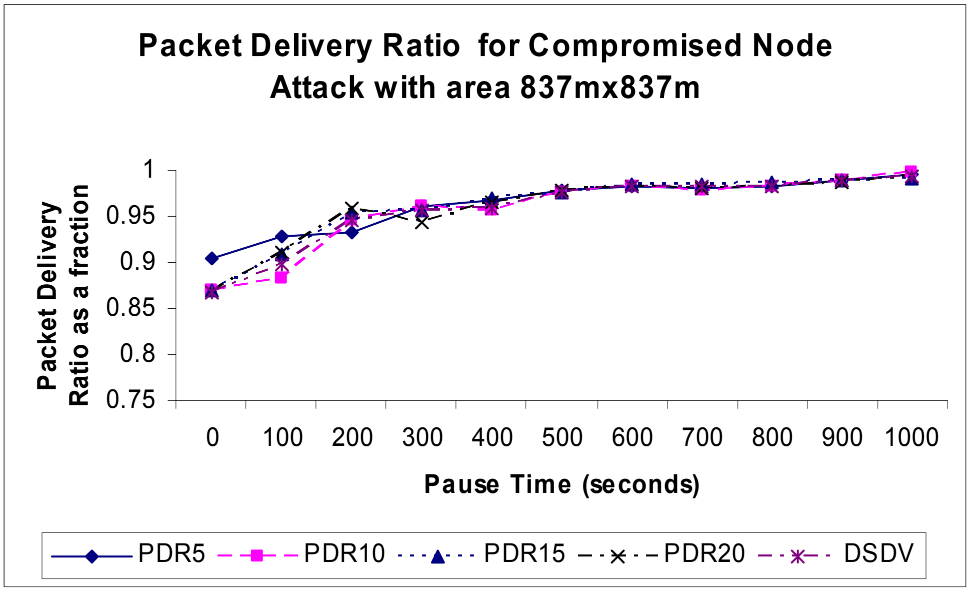


Figure 6.18: Packet Delivery Ratio for Compromised Node Attack with area 837mx837m

6.3.2.2 Replay Attack

Figures 6.19-6.20 plot the packet delivery ratio for Replay Attack scenarios with the pause time varied from 1000 seconds to 0 seconds in 100-second increments for each of the four malicious node percentage cases and with the node density varied as well. This metric’s performance closely matches that of plain DSDV protocol with the number of malicious nodes in the network varied from 5% to 20%. With an increase in node mobility, the packet delivery ratio decreases. As with the packet delivery behavior observed for compromised ndoes, we find that the packet delivery ratio decreases by about 10% as we increase the node mobility from a static to a continuously dynamic network. Decreased node density did seem to have a better packet delivery behavior with about 7% decrease. Also, an increase in the malicious nodes seemed to have a negative effect on the packet delivery as we can see in the graphs 6.19-6.20. Higher number of malicious nodes in the network had lower values of packet delivery ratio. This is seen in the value of packet delivery ratio at 5% which is higher than that at 20%. One possible explanation is that due to higher node mobility and increasing number of routing changes, routing agents might drop packets. A routing agent may drop the packet if it knows that it will not be able to route the received packet, because there is no route.

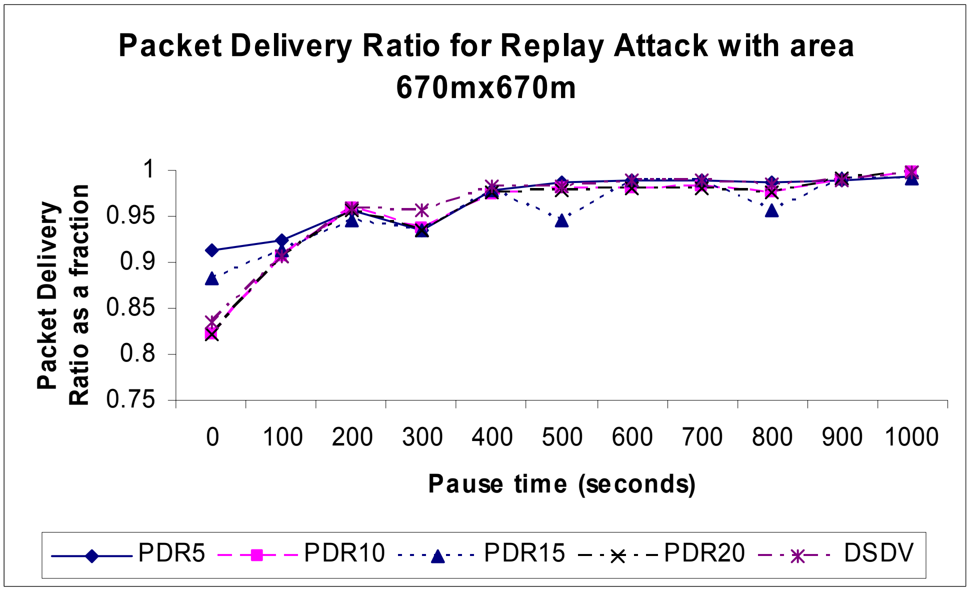


Figure 6.19: Packet Delivery Ratio for Replay Attack with area 670mx670m

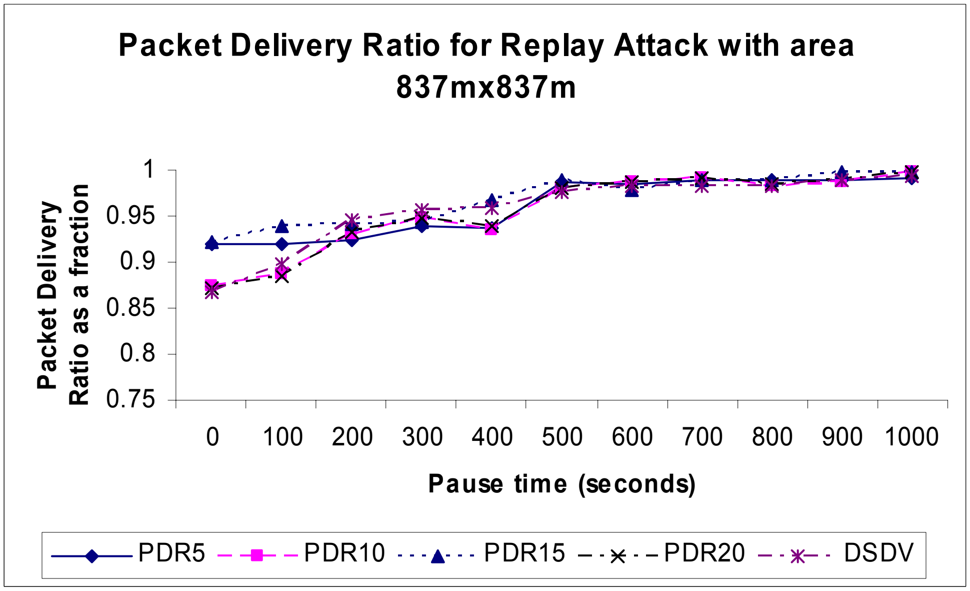


Figure 6.20: Packet Delivery Ratio for Replay Attack with area 837mx837m

6.3.2.3 Denial-of-Service Attack

Figures 6.21-6.22 plot the packet delivery ratio for Denial-of-Service Attack scenarios with the pause time varied from 1000 seconds to 0 seconds in 100-second increments for each of the four malicious node percentage cases and with the node density varied as well. This metric's performance closely matches that of plain DSDV protocol with the number of malicious nodes in the network varied from 5% to 20%. The only exception is in the packet delivery behavior at 0 second pause time. The extremely high levels of node mobility lead to abnormal behavior for high percentage of malicious nodes. The packet delivery ratio decreases from about 0.95 to 0.9 or thereabouts at high levels of node mobility unlike the general trend. There is a 5.5% difference in values at 0 second pause time. We find that an increase in node mobility leads to a decrease in packet delivery ratio. Drops could occur in different network layers such as routing agents, interface queues, buffer overflows, etc.

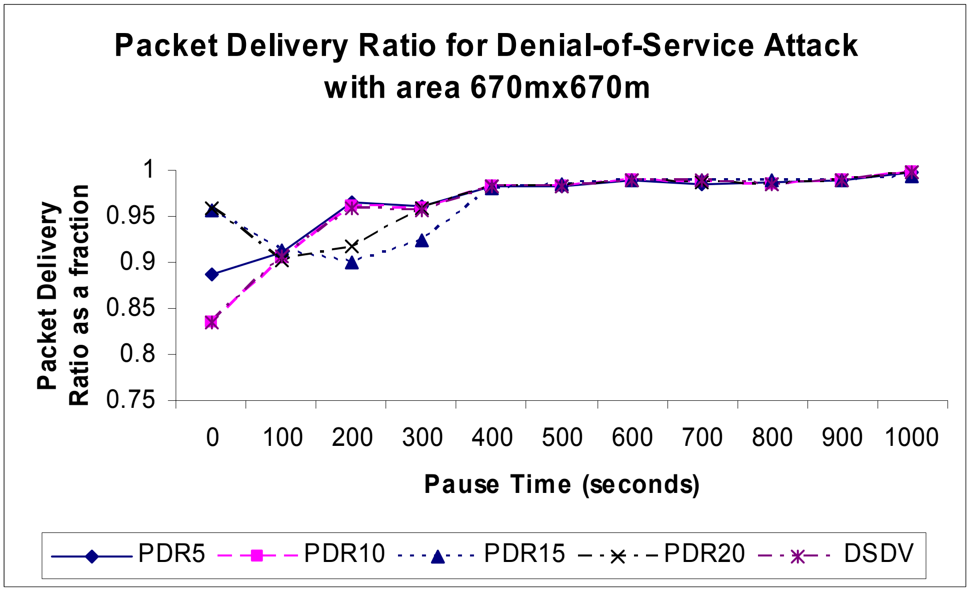


Figure 6.21: Packet Delivery Ratio for Denial-of-Service Attack with area 670mx670m

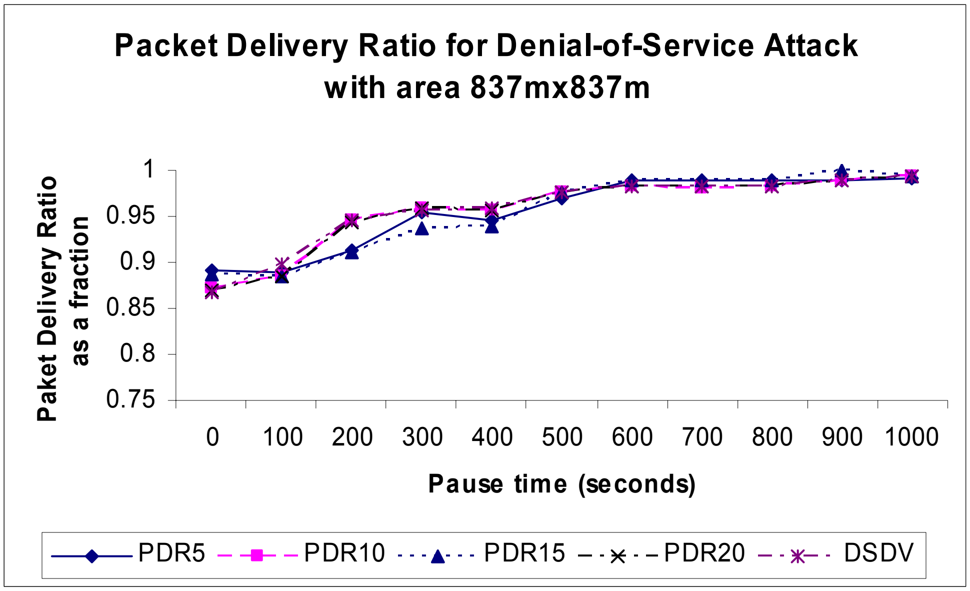


Figure 6.22: Packet Delivery Ratio for Denial-of-Service Attack with area 837mx837m

6.3.2.4 Overall Packet Delivery Ratio

An increase in the number of routing packets means a decrease in the number of data packets reaching the destination. This in turn leads to a decrease in the network throughput. However, the decrease in throughput is relatively small (about 2%) as shown in Figure 6.23. The presence of high mobility implies frequent link failures and DSDV fails to converge below lower pause times. At high mobility (lower pause times), DSDV does poorly, dropping to around 75% packet delivery ratio. Nearly all of the dropped packets are lost because a stale routing table entry directed them to be forwarded over a broken link. DSDV maintains only one route per destination and consequently each packet that the MAC layer is unable to deliver is dropped since there are no alternate routes.

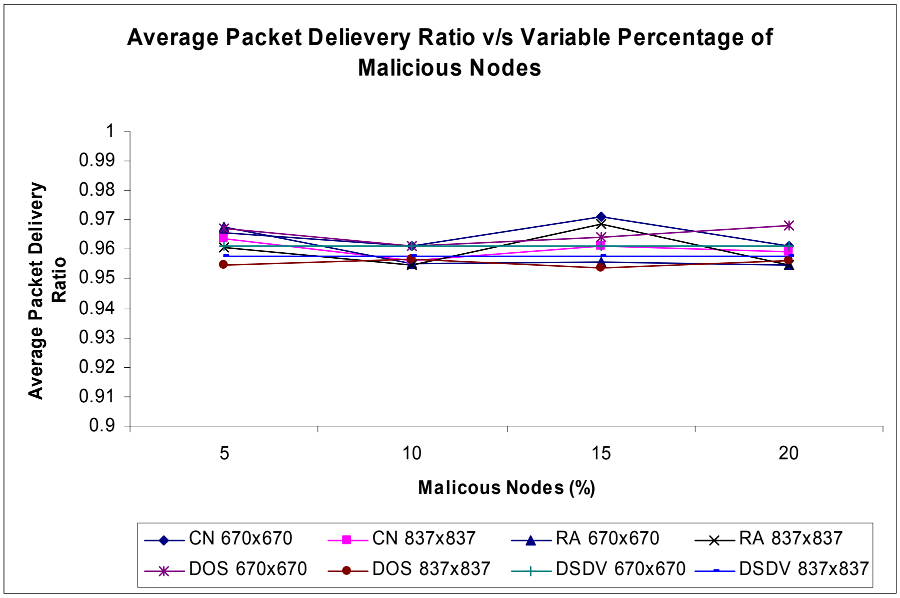


Figure 6.23: Average Packet Delivery Ratio versus varying percentage of malicious nodes

6.3.3 Normalized Routing Load

In this subsection, we analyze the simulation results of another performance metric, namely, normalized routing load. This metric is known to reflect the routing protocol efficiency.

6.3.3.1 Compromised Node

In Figures 6.24-6.25, we have plotted the normalized routing load values versus pause time for compromised node scenarios. This has been done with varying percentage of malicious nodes in the network as well as by increasing the flat meter movement space by 25%. The measured values through simulation of compromised node scenarios with varying percentages of malicious nodes in the network are almost exactly the same as those of plain DSDV protocol and we find that the results are not affected by node mobility.

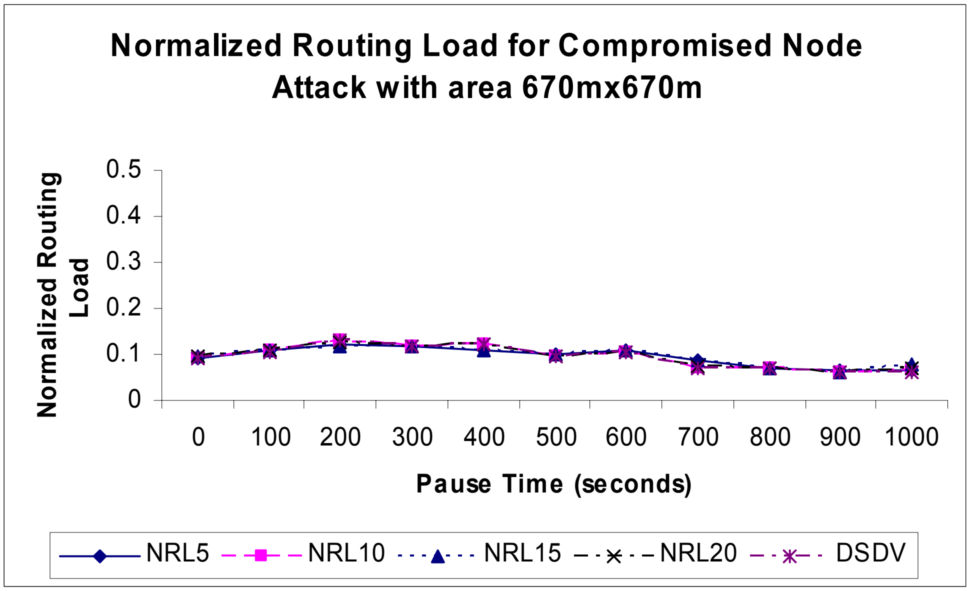


Figure 6.24: Normalized Routing Load for Compromised Node Attack with area 670mx670m

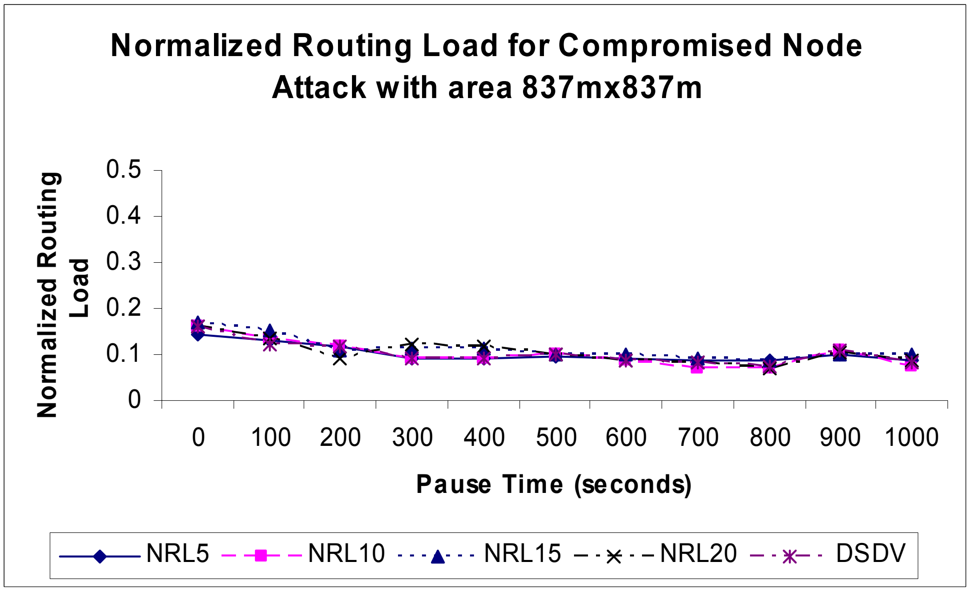


Figure 6.25: Normalized Routing Load for Compromised Node Attack with area 837mx837m

6.3.3.2 Replay Attack

In Figures 6.26-6.27, we have plotted the normalized routing load values versus pause time for Replay Attack scenarios. This has been done with varying percentage of malicious nodes in the network as well as by increasing the flat meter movement space by 25%. Unlike compromised node scenarios, we find that simulation of replay attack led to higher values of normalized routing load. This in turn reflects negatively on the routing protocol efficiency. Fortunately, the percentage increase in values with our intrusion detection scheme introduced into the network is only about 2% as compared to plain DSDV.

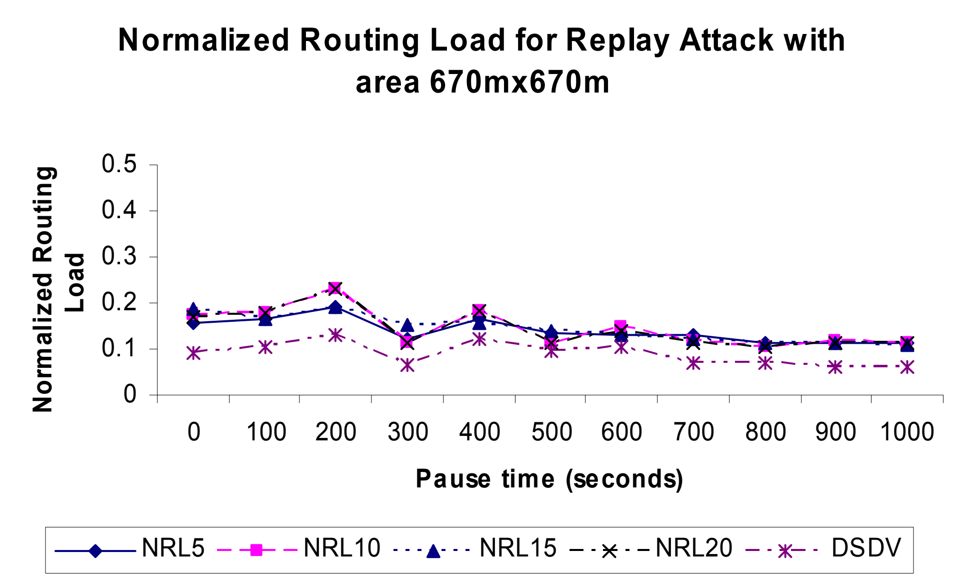


Figure 6.26: Normalized Routing Load for Replay Attack with area 670mx670m

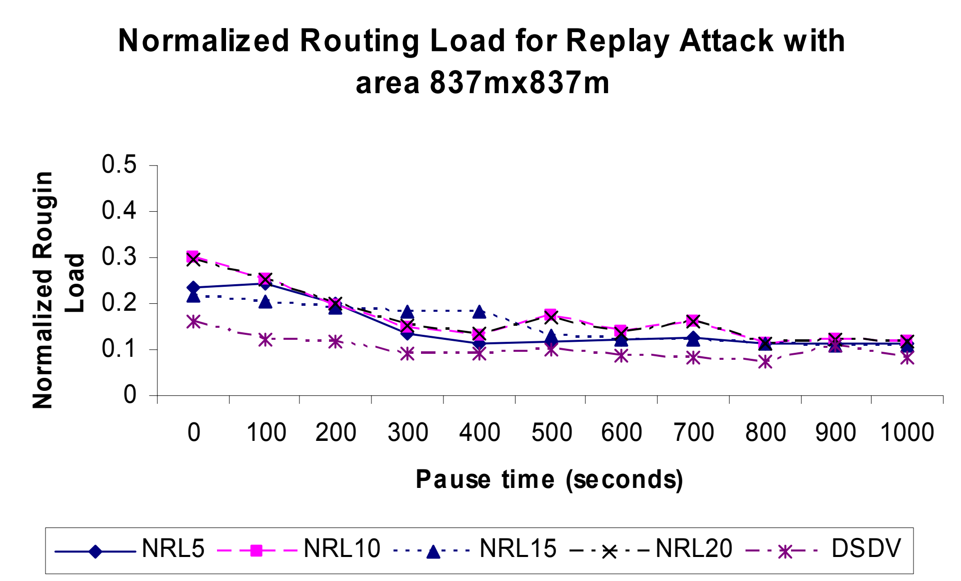


Figure 6.27: Normalized Routing Load for Replay Attack with area 837mx837m

6.3.3.3 Denial-of-Service Attack

In Figures 6.28-6.29, we have plotted the normalized routing load values versus pause time for Denial-of-Service scenarios. This has been done with varying percentage of malicious nodes in the network as well as by increasing the flat meter movement space by 25%. Simulation of Denial-of-Service attacks provided excellent results. There was a 0.1% ± change in the performance of this metric in the presence of our IDS as compared to plain DSDV. Node mobility and also node density (as seen by the behavior of normalized routing load with an increase of 25% in the movement space), did not have an adverse effect on the performance.

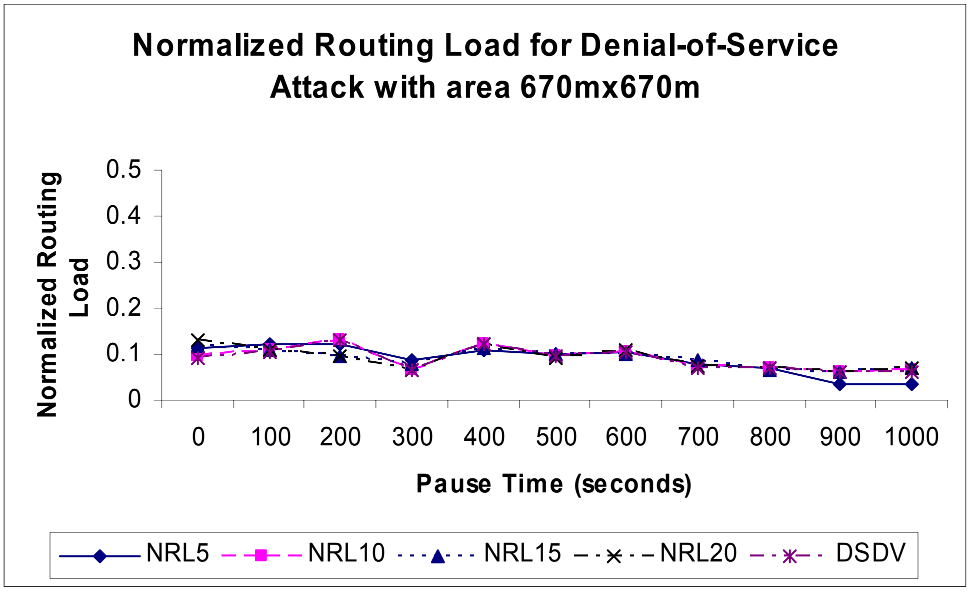


Figure 6.28: Normalized Routing Load for Denial-of-Service Attack with area 670mx670m

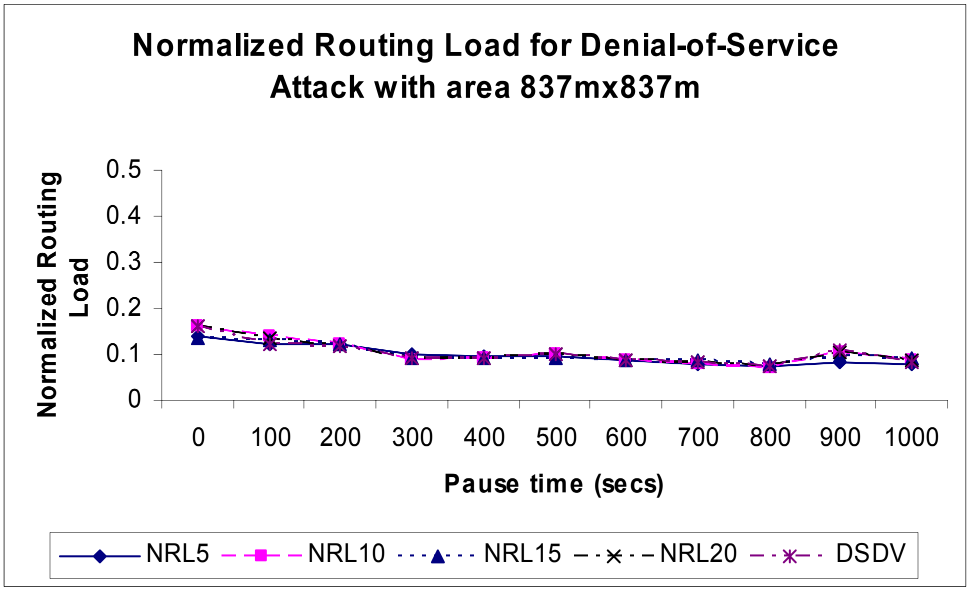


Figure 6.29: Normalized Routing Load for Denial-of-Service Attack with area 837mx837m

6.3.3.4 Overall Normalized Routing Load

The performance of the network under various levels of misbehaving nodes is not negatively affected by the introduction of our IDS. As we can see from the graph plotted in Figure 6.30, the overall NRL remains relatively the same as that under network conditions with the original protocol. In certain scenarios though, we find that routing load increases slightly. This could be attributed to the additional processing being done by our IDS to detect malicious nodes in the network. A high-level view of the routing load plots indicate that at higher levels of mobility in the network, there is more routing load compared to lower levels of mobility. This is because with the routes and hops changing more frequently, routing updates are sent at a faster rate and this leads to an increase in routing load.

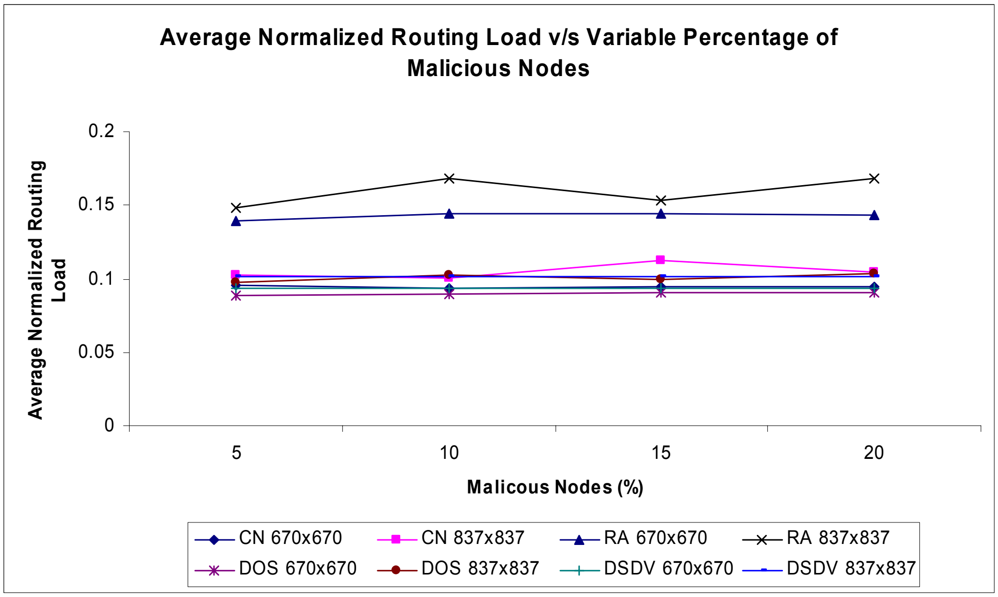


Figure 6.30: Average Normalized Routing Load versus varying percentage of malicious nodes

6.4 Chapter Summary

In this chapter, we analyzed the performance of our intrusion detection scheme. This involved analyzing the robustness of our IDS in the presence of malicious nodes. The primary research goal was to study the accuracy of our detection scheme (i.e. whether our IDS correctly detects simulated malicious nodes and also whether it flags false positives). Moreover, the performance of our IDS was studied as regards the efficiency of the security scheme which included analyzing the effect of performance metrics such as Average End-to-End Delay, Packet Delivery Ratio and Normalized Routing Load. The robustness of the scheme was analyzed by taking into consideration the effect of varying node mobility, node density and percentage of malicious nodes in the network.

7. Conclusion and Future Work

Wireless mobile ad hoc networks (MANETs) have come under the research microscope in recent times because of the tremendous potential for rapid deployment in harsh geographic terrain, on-the-fly setup in conference-room like scenarios and in military operations. However, the very set of characteristics which make these networks advantageous, lead to inherently vulnerable structure and make these networks highly susceptible to attackers with malicious intent. Thus, security assumes prime importance. With the network topology changing dynamically, attacks can come from within the network in addition to the classical attacks from outside entities. A high-survivability network necessitates the need for intrusion detection in addition to intrusion prevention measures such as authentication and encryption.

The absence of an underlying infrastructure in MANETs means that security schemes already developed for traditional wireless networks cannot be easily implemented for ad hoc networks. There is a need to look at developing security solutions in a wholly different manner, ideally involving a fully distributed and cooperative approach. Moreover, there is a need to take a layered approach while designing security solutions.

7.1 Conclusion

Intrusion detection is conceptually based in principle either on anomaly detection or misuse detection. Our research involved intrusion detection in wireless mobile ad hoc networks based on integrating misuse detection with anomaly detection. There is a very fuzzy line between normalcy and anomaly, in general, and more so, in the case of ad hoc networks, which leads to a high degree of false alarms if only anomaly detection is used. Compound detection brings together the accuracy of misuse detection and the normal profiling of anomaly detection, leading to a powerful audit data analysis for intrusion detection.

We focused on some significant deterministic attacks that can have a damaging effect on the working of a MANET especially in military scenarios. A compromised node, for instance, can, not only lead to passive eavesdropping but also to active interfering. The routing protocol's correct functioning breaks down if a malicious node launches a replay attack on the system. The bandwidth constraints of a MANET can be exploited by a malicious attacker by launching a Denial-of-Service attack.

Experimental validation of our model provided significant results about the accuracy and robustness of our intrusion detection scheme:

- The accuracy of the intrusion detection scheme is, on an average, over 90% and the degree of false alarms is negligibly low (about 2% and that too, only for Replay Attack).
- Node density and increasing levels of malicious nodes in the network have a marginally adverse effect on the accuracy of the scheme (around 5% decrease in

accuracy with increased flat meter space for Compromised Node attack). This indicates the robustness of the scheme.

- The network performance does not degrade upon introduction of our IDS into the MANET. The performance metrics utilized for the simulation analysis (i.e. average end-to-end delay, packet delivery ratio and normalized routing load) are marginally affected by our intrusion detection scheme (about ± 2% variation from plain DSDV).

Based on combining misuse with anomaly detection, our IDS for MANETs accurately and efficiently detects attacks such as DoS, replay attack and compromised nodes. Our results have shown a great promise for the future, which would focus on making the scheme more robust by taking a broader range of attacks into consideration and making use of audit data to accurately adjust the threshold.

7.2 Limitations

In this section, we identify some of the shortcomings of our intrusion detection scheme which can provide ideas for future work in this area. Some of the limitations of our intrusion detection scheme include:

- Our scheme has been targeted at distance vector protocols. It has been implemented for DSDV protocol and could just as easily be implemented for other table-driven distance vector protocols. However, source initiated on-demand routing protocols would involve a different approach for intrusion detection.

7.3 Future Work

Our research on intrusion detection has thrown up some interesting issues, which can form the basis of future work:

- Integration of anomaly-based detection into the overall intrusion detection architecture.

- Suitability of our intrusion detection approach for source-initiated on-demand routing protocols.

- Increasing the number and types of attacks and optimizing threshold values.

- Analyze the impact of non-deterministic attacks, which require statistical analysis.

- Collection of mobile nodes working in tandem to launch an attack, which essentially involves study of Byzantine failure.

Bibliography

[Aid] http://www-rnks.informatik.tu-cottbus.de/~sobirey/aid.e.html

[Albers02] Albers et al., "Security in Ad hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches", http://www.supelecrennes.fr/ren/perso/bjouga/documents/wis-short2002.pdf

[Ax00] Stefan Axelsson, "Intrusion Detection Systems: A Taxomomy and Survey", Technical Report No 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000

[BhAg01] Sonali Bhargava and Dharma P. Agrawal, "Security Enhancements in AODV protocol for Wireless Ad Hoc Networks", Vehicular Technology Conference, Fall 2001, Atlantic City, October 7-11, 2001

[BrMaJoHuJe98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 85-97, Dallas, TX, October 1998.

[ChGe98] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks", Proc. IEEE ICC'98, 5 pages.

[CMU] CMU Monarch Group. CMU Monarch Extensions to ns. http://www.monarch.cs.cmu.edu

[CoEp95] M.S. Corson and A. Ephremides, "Lightweight Mobile Routing protocol (LMR), A distributed routing algorithm for mobile wireless networks", Wireless Networks 1 (1995).

[DuRaWaTr97] R. Dube, C. D. Rais, K. Wang, And S. K. Tripathi, "Signal Stability based adaptive routing (SSR alt SSA) for ad hoc mobile networks", IEEE Personal Communication, Feb. 1997.

[Emerald] http://www.sdl.sri.com/projects/emerald/index.html

[FaVa97] Kevin Fall and Kannan Varahan, editors. NS Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997.

[HuJoPe02] Y. Hu, D. Johnson, A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), IEEE, Calicoon, NY, June 2002.

[idwg] http://www.ietf.org/html.charters/idwg-charter.html

[ietf] http://www.ietf.org/html.charters/manet-charter.html

[Jo94] D.B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," Proc. IEEE Workshop Mobile Computing Systems and Applications, Dec. 1994.

[JoMa96] David B. Johnson and David A. Maltz, "Dynamic source routing in ad hoc wireless networks", in Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181.Kluwer Academic Publishers, 1996.

[JoMaHuJe01] D. Johnson, D. Maltz, Y-C. Hu And J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", Internet Draft, draft-ietf-manet-dsr-05.txt, work in progress, June 2001.

[JuTo87] John Jubin and Janet D. Tornow, "The DARPA Packet Radio Network Protocols", Proceedings of IEEE, 75(1), pages 21-32, January 1987.

[KaGu02] Oleg Kachirski, Ratan Guha, "Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks", IEEE Workshop on Knowledge Media Networking (KMN'02)

[KrTo02] Krugel, C., Toth, T. "Flexible, Mobile Agent Based Intrusion Detection for Dynamic Networks" http://www.ing.unipi.it/ew2002/proceedings/081.pdf

[MaGiLaBa00] Sergio Marti, TJ Giuli, Kevin Lai, and Mary Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", in Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, August 2000

[MiNa02] Mishra, A., and Nadkarni, K., "Security in Ad Hoc Wireless Networks", book chapter in the "Handbook of Ad Hoc Wireless Networks", Mohammad Illyas (Editor), CRC Press, December 2002

[MuGa96] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, Oct. 1996, pp. 183-97.

[NaMi03] Nadkarni, K. and Mishra A., "Intrusion Detection in MANETs – The Second Wall of Defense" to be published in Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society, November, 2003

[Nid] http://ciac.llnl.gov/cstc/nid/intro.html

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

[PaCo] V. Park, S. Corson, “Temporally-Ordered Routing Algorithm (Tora)” Version 1 Internet Draft, draft-ietf-manet-tora-spec-03.txt, work in progress, June 2001.

[PeBh94] Charles E. Perkins and Pravin Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers”, in Proceedings of the SIGCOMM ’94 Conference on Communications Architectures, Protocols and Applications, pages 234–244, August 1994.

[PeRoDa02] C. Perkins, E.Royer And S. Das, “Ad hoc On-demand Distance Vector (AODV) Routing”, Internet Draft, draft-ietf-manet-aodv-11.txt, work in progress, Aug 2002.

[RFC 792] http://www.faqs.org/rfcs/rfc792.html

[RFC 2501] http://www.faqs.org/rfcs/rfc2501.html

[RoTo99] Elizabeth Royer and C-K Toh, “A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks”, IEEE Personal Communications Magazine, April 1999, pp. 46-55.

[Toh97] C.K. Toh, “Associativity-Based Routing for Ad-Hoc Mobile Networks”, Wireless Personal Communications, Vol. 4, No. 2, pp. 1-36, Mar. 1997.

[Sm01] Smith, Andrew B., “An Examination of an Intrusion Detection Architecture for Wireless Ad Hoc Networks”, Presented to the 5th National Colloquium for Information System Security Education. (May 2001).

[SmMuGa97] B. R. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves, “Securing distance vector routing protocols”, in Proceedings of Internet Society Symposium on Network and Distributed System Security, pages 85–92, San Diego, California, February 1997.

[Wavelan] Bruce Tuch, Development of WaveLAN, an ISM Band Wireless LAN. AT&T Technical Journal, 72(4), pages 27-33, July/August 1993.

[Yang03] http://ame-www.usc.edu/crs/summer03/me/451/Lecture%204.pdf

[ZhHa99] L. Zhou and Z. J. Haas, “Securing ad hoc networks”, IEEE Network, 13(6):24–30, Nov/Dec 1999.

[ZhLe00] Y. Zhang and W. Lee, “Intrusion detection in wireless ad-hoc networks”, in Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000), pages 275–283, Boston, Massachusetts, August 2000.

APPENDIX A – Simulator Information

Network Components in a Mobile Node in NS2
The network stack for a mobile node consists of a link layer (LL), an ARP module connected to LL, an interface priority queue (Ifq), a mac layer (MAC), a network interface (netIF), all connected to the channel. These network components are created and combined in Otcl.

Link Layer: The link layer for a mobile node has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally, all outgoing (into the channel) packets are handed down to the LL by the routing agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel) the mac layer hands up packets to the LL, which are then handed off to node_entry_point. The class LL is implemented in: ns-allinone-2.1b6/ns-2.1b6/ll.{cc,h}

ARP: The Address Resolution Protocol module receives queries from Link layer. If ARP has the hardware address for the destination, it writes it into the mac header of the packet. Otherwise it broadcasts an ARP query and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. In case additional packets to the same destination are sent to ARP, the earlier buffered packet is dropped. Once the hardware address of the packet's next hop is known, the packet is inserted into the interface queue. The class ARPTable is implemented in ns-allinone-2.1b6/ns-2.1b6//arp.{cc,h}

Interface Queue: The class PriQueue is implemented as a priority queue that gives priority to routing protocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address. The class is implemented in ns-allinone-2.1b6/ns-2.1b6/priqueue.{cc.h}

Mac layer: The IEEE 802.11 distributed coordination function (DCF) MAC protocol has been implemented by CMU. It uses the RTS/CTS/DATA/ACK pattern for all unicast packets and simply sends out DATA for all broadcast packets. The implementation uses both physical and virtual carrier sense. The class Mac802_11 is implemented in nsallinone-2.1b6/ns-2.1b6/mac-802_11.{cc,h}

Network Interfaces: The Network Interface layer serves as a hardware interface that is used by mobile node interfaces to the channel. The wireless shared media interface is implemented as class Phy/WirelessPhy. This interface is subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength. This meta-data in the packet header is used by the propagation model on the receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (Lucent WaveLen direct-sequence spread-spectrum). The class is implemented in ns-allinone-2.1b6/ns-2.1b6/wireless-phy.{cc,h}

Radio Propagation Model: The Radio Propagation Model uses Friss-space attenuation ($1/r^2$) at near distances and an approximation to Two Ray Ground ($1/r^4$) at far distance. The approximation assumes specula reflection off a flat ground plane. The class is implemented in ns-allinone-2.1b6/ns-2.1b6/tworayground.{cc,h}

Antenna: An omni-directional antenna with unity gain is used by mobile nodes. The class in implemented in ns-allinone-2.1b6/ns-2.1b6/antenna.{cc,h}

Wireless Channel: The wireless channel duplicates packets to all mobile nodes attached to the channel except the source itself. It is the receiver's responsibility to detect if it can receive the packet.

Creating Mobile Node Movement Scenario Files
The node-movement generator is available under:
ns-allinone-2.1b6/ns-2.1b6/indep-utils/cmu-scen-gen/setdest directory.

Run setdest with arguments as shown below:

```
./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] [-x maxx] [-y
maxy] > [outdir/movement-file]
```

Creating Random Traffic Pattern for Wireless Scenarios
Random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic-scenario generator script. This traffic generator script is available under:
ns-allinone-2.1b6/ns-2.1b6/indep-utils/cmu-scen-gen

It is called cbrgen.tcl. The command line looks as the follows:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate
rate]>[outdir/movement-file]
```

The rate parameter is only used for CBR packets. Its inverse value is used to compute the time interval between the CBR packets. To set the data packet size, modify the parameter item in cbrgen.tcl: "set opt(pktsize) 64"

Vita

Ketan Nadkarni, the son of Milind and Sunita Nadkarni, was born on October 26th, 1977, in Bombay (now known as Mumbai), India. He graduated from Goa University, India in July 1999 with a B.E. in Electronics and Telecommunication Engineering. He was awarded "Best Outgoing Student of the Year" Award by Goa Engineering College, India in 1999. He was also awarded the "Late Smt. S.V. Khandeparker Prize" at Goa University's Annual Convocation Ceremony in April 2000 for topping overall B.E. (Electronics and Telecommunication).

Ketan was recruited on campus by Tata Infotech Ltd., one of India's top I.T. companies. After working for two years as a Systems Engineer, he came to the U.S. in the fall of 2001 to pursue his Master's degree in Computer Science at Virginia Tech. This thesis completes his Master’s degree in Computer Science and Applications. He is currently working as a Software Developer for Meridium, Inc. - one of the world leaders in enterprise asset performance management for the process industry.