UNIVERSITY OF CINCINNATI

Date: February 11,2008

I, Lakshmi Santhanam,

hereby submit this work as part of the requirements for the degree of:

Doctoral of Philosophy

in:

Computer Science and Engineering

It is entitled:

Integrated Security Architecture for Wireless Mesh Networks

This work and its defense approved by:

Chair: Dr. Dharma P. Agrawal

Dr. Raj K. Bhatnagar

Dr. Kenneth Berman

Dr. Wen-Ben Jone

Dr. Ming Ming Lu

Integrated Security Architecture for Wireless Mesh Networks

A Dissertation submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

In partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science and Engineering
of the College of Engineering
February, 2008

By

Lakshmi Santhanam

Bachelor of Engineering (Computer Science and Engineering)
University of Madras,
Chennai, India, 2003

Dissertation Advisor and Committee Chair: **Dr. Dharma P. Agrawal**

Abstract

Wireless Mesh Networks (WMNs) have revolutionized provisioning of economical and broadband wireless internet service to the whole community of users. The self-configurable and self-healing ability of WMNs has encouraged their rapid proliferation, as adding a mesh router (MR) is as simple as plugging and turning on. The plug-and-play architecture of WMN, however paves way to malicious intruders. An attacker can raise several security concerns, like rogue routers, selfishness, and denial-of-service attacks. Unfortunately, current thrust of research in WMNs, is primarily focused on developing multi-path routing protocols; and security is very much in its infancy.

Owing to the hierarchical architecture of WMNs, security issues are multi-dimensional. As mesh routers form the backbone of the network, it is critical to secure them from various attacks. In this dissertation we develop integrated security architecture to protect the mesh backbone. It is important to provide an end-to-end security for mesh clients and hence we design a novel authentication protocol for mutually authenticating mesh clients and mesh routers.

The aim of this dissertation is to explore various issues that affect the performance and security of WMNs. We first examine the threat of an active attack like Denial of service attack on MRs and design a cache based throttle mechanism to control it. Next, we develop a MAC identifier based trace table to determine the precise source of a DoS attacker. We then evaluate the vulnerability of WMNs to passive attacks, like selfishness and propose an adaptive mechanism to penalize selfish MRs that discretely drop other's packets. In order to handle route disruption attacks like malicious route discovery, we design an intelligent Intrusion Detection System. Through extensive simulations, we evaluate effectiveness of our proposed solutions in mitigating these attacks. Finally, we design a light weight authentication protocol for mesh clients using inexpensive hash operations that enables authentication of important control messages and also performs auto-refresh of authentication tokens.

To dear amma & appa,

My dear bro & sis

for their constant support & encouragement

Acknowledgement

I am would like to express my profuse gratitude to my advisor Prof. Dharma P. Agrawal for his visionary guidance and support during the course of my doctoral studies. He actively encouraged me to interact with senior members, visiting faculties, and helped me expand my research horizons. He provided me the necessary latitude to conduct inter-disciplinary research with other faculties, encouraged me to publish conference, journal papers, give tutorials in international conference. I especially thank Dr. Anup Kumar for the numerous hours that he spent with me discussing my research problems. I would also like to thank Dr. Raj Bhatnagar for sharing his expertise in the field of Artificial Intelligence and providing me much needed guidance at a point when I was totally struck with the jargons of AI.

I am also indebted to my beloved father, Prof. R. Santhanam, who urged and motivated me to do PhD from a young age. I would like to thank my beloved mother Mrs. T. Marakadam for her constant love and support. I am also thankful to my dearest brother Dilip and sister Rama who have always been with me in all my endeavors. I am deeply thankful to my colleagues Nagesh and Dr. Bin Xie for their continuous academic support and guidance. I also owe my friends, Nisha, Sadhana, Prithvi, Manjari, Gayathri, Nithya, Karen, Deepti, Radha, Aravind, Jagdish, Arun, Abhinay, Akhil, Yuva, Madhuri, Neena, Karen, Veni, Anusha, Sravanti, Priyanka, Latha and my east campus friends Charity, Soni, Marybeth, Ingrid, Harini, a huge bunch of thanks whose constant support has helped me face challenges & disappointments boldly.

I also express my thanks to Dr. Ken Berman, Dr. Wen-Ben Jone, and Dr. Mingming Lu for taking their time to be on my dissertation committee and offering valuable suggestions to enhance the quality of this dissertation.

Lastly, I am thankful to all my fellow CDMC lab mates with whom I have spent memorable amount of time discussing my research & assignments, and enjoyed friendly chit chats in between work.

Contents

List of Figures

List of Tables

Chapter

1. Introduction

Wireless Mesh Networks (WMNs) have revolutionized the provisioning of economical and broadband wireless internet service to a community of users. The self-configurable and self-healing ability of WMNs has encouraged their rapid proliferation, as adding a mesh router (MR) is as simple as plugging it and turning it on. A group of static mesh router automatically interconnect themselves to form a web of connection and employ multi-hop forwarding to connect to the Internet Gateway (IGW). The underlying paradigm of WMNs is similar to multi-hop MANETs [1] [2].

The plug-and-play architecture of WMN, however paves way to malicious intruders [3]. It is a very common misconception that MRs are always managed by a single trusted entity. This is not always true [4] and it is critical to protect the open architecture of WMN from various attacks conducted by a malicious intruder. Security is a fundamental characteristic that should be embedded at an early stage of network functionality of WMNs [1]. Thus, the focus of this dissertation is to systematically explore the vulnerabilities that can be exploited by attackers to

conduct various attacks, like denial of service attack, selfish node attack, route flooding attacks, etc.. We then provide a detailed description of our integrated security framework that employs intelligent detection agents to thwart the infiltration of such attacks.

In this chapter, we present a detailed overview of WMN and emphasize the necessity for developing new security solutions for WMNs. We first understand various operational details of WMNs. Then, we outline why schemes for ad hoc or cellular network cannot be directly applied to a WMN and present the need for developing new security approaches. We then describe motivation behind our work, outlining some key security challenges that are hindering the wide scale deployment of WMNs. Finally, we provide an outline of our dissertation and summarize the key contributions of our research.

1.1 Wireless Mesh Networks

Wireless Mesh Networks (WMNs) are evolving as a new paradigm of ubiquitous broadband Internet access that promotes a tight integration of the concept in MANETs (Mobile Ad hoc Networks) with infrastructure technology. In spite of decade's research effort on MANETs; their applicability domain seems to be primarily limited to the military arena. On the other hand, within a short span of time, WMNs have stirred considerable interest in the commercial and academic spheres [5].

With the growing popularity of wireless networks, service providers are facing a tedious task of laying wired infrastructure to the Access Points (APs). The WMNs have popularized the age old concept of MANET by applying its underlying characteristics to mesh APs, commonly referred to as Mesh Routers (MRs) in WMNs. Thus, WMNs obviate the need for extensive wired infrastructure at every MR (or APs) by connecting only a subset of MRs known as Internet Gateways (IGWs) to the wired Internet. A WMN excels in the performance by providing

seamless broadband connectivity [6], as compared with other peer technologies such as cellular networks and WLAN. A cellular network offers wide area coverage, but provides low channel capacity (at best 3Mbps in 3-G and at best 100 Mbps in 4-G); while the WLANs 802.11 network offers an attractive high bandwidth connectivity (802.11g currently in user at 54 Mbps and 802.11n with a theoretical throughput of 540Mbps) at a very limited range.

A typical WMN consists of a hierarchical architecture consisting of three layers as shown in Figure 1.1. IGWs, located at the top of the hierarchy, form the backbone infrastructure for providing Internet service to the second level. The second level of the hierarchy consists of MRs.

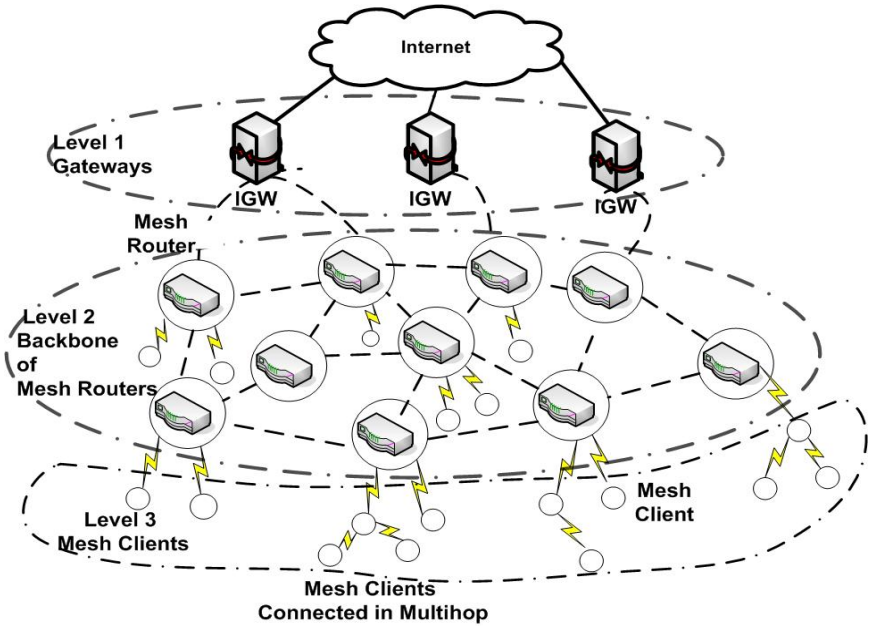


Figure 1.1: Hierarchical architecture of WMN

The wireless MRs can be operated on multiple orthogonal channels (e.g., 12 for IEEE 802.11a and 3 for IEEE 802.11b/g) to facilitate multi-channel communication [7]. The interconnected mesh of MRs (core routers) collaboratively forwards the network traffic in a multi-hop fashion towards the IGW. On the lowest level of the hierarchy are the mesh clients that connect to the nearest MR in a single hop or multi-hop fashion to get Internet services like multi-media applications, messaging, e-mail, etc.

1.2 WMN Operational Structure

The underlying paradigm of WMNs is similar to multi-hop MANETs [1]. All existing routing protocols in WMNs assume the intermediate MRs to function honestly. However, such an assumption is valid only in a WMN operated by a single trusted authority. *A-priori trust relationship* exists only in some WMN application scenarios like enterprise network, military, or university. A community based WMN, for example, is formed by a group of MRs that are managed independently by different operators or networks [4]. It is a tough challenge to establish a-priori trust in such a user-deployed WMN due to their divergent interest.

It is a common misconception that a WMN is managed by a single trusted entity. In fact, WMNs can be classified into three categories based on their operational structure: fully managed network, semi-managed network and unmanaged network.

- In a fully managed WMN, the entire network is administered by a single ISP (Internet Service Provider).

  The Cisco's project covering various U.S cities is one such example. As all the MRs are under the control of an ISP, the MRs can assume each other to be trustworthy.

- In a semi-managed WMN, only few MRs are managed by an ISP.

  While the core of the network is managed by a single ISP, some of the MRs are left unmonitored as they lie outside the jurisdiction of an ISP.

- In an unmanaged WMN, the MRs are managed by independent entities.

  They are formed on ad hoc basis and do not possess any permanent infrastructure. Hence, there exists no authority to manage them. For example, each user in a community can independently install a MR on the roof of their house and share the internet connection to a near-by ISP.

The existence of different kinds of operational structures of WMNs has always been debated by the researchers. Bruno et al. [5] define WMN as an “opportunistic ad hoc networking” that facilitates a low-cost extension to the wired infrastructure. But, researchers like Kyasanur et al. [8] are strong proponents of infrastructure based WMNs. Salem et al. [3] acknowledge the existence of two kinds of networks: single operator WMNs (owned by a single ISP) and multi-operator WMNs (MRs owned by independent operators/networks). A fully managed network is secure from internal attackers due to the trustworthiness of the MRs. But, an unmanaged/semi-managed networks raises several security issues from an inside as well as outside attacker. The underlying heterogeneous communication via independent MRs and multi-hop forwarding raises several security challenges like traffic interception, modification, and dropping [9].

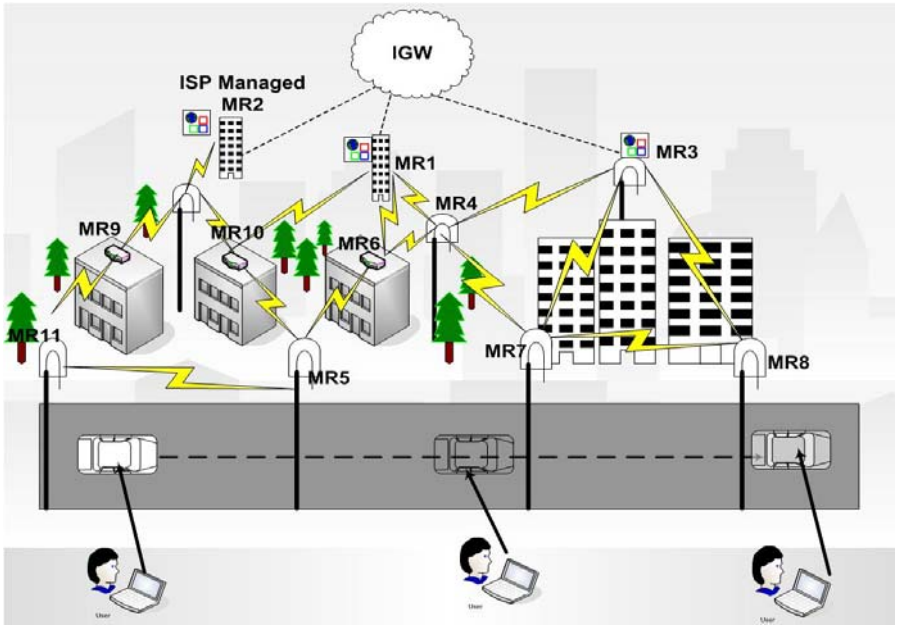


Figure 1.2: An Application Scenario of a Semi-managed WMN

Consider a semi-managed mesh architecture in which few MRs (e.g., MRs 1, 2, and 3 in Figure 1.2) are under the jurisdiction of an ISP. But, some MRs (e.g., MRs 6, 9, and 11 in Figure 1.2) are left unmonitored (deployed by independent enterprises). Figure 1.2 shows an application scenario of a semi-managed WMN. Such an architecture would provide broadband Internet

access to its mesh clients at all places and at all time; even on their way from office to home or as they stroll through a park or a mall. Hence, it is critical to ensure a secure environment to reap the benefits of a ubiquitous wireless Internet service.

1.3 Applicability of Existing Security Schemes to WMNs

Though, the underlying paradigm of WMNs is somewhat similar to multi-hop ad hoc networks, they cannot be treated in the same way due to several discrepancies. Due to the unique properties of WMNs, the security threats are yet to be understood fully.

- In WMNs, the bulk of the traffic is forwarded by the MRs near the IGW and an attack on these MRs affects the descendant MRs drastically. Also, WMNs are envisioned to serve a large community of users. A mere 130 outdoor mesh routers can provide broadband service to a city area as large as 8 square miles (operated by Airmesh Communication Inc [5]). Thus, the consequence of an attack on the backbone MRs is widespread.
- Unlike peer-to-peer traffic in an ad hoc network, the traffic in a WMN is predominantly oriented towards/from the IGW. The routers near the IGW are potential hotspots for congestion. Hence, they can fall easy prey to denial-of-service (DoS) attack by malicious attackers or can selfishly avoid forwarding packets in order to avoid congestion.
- Unlike ad hoc networks, WMNs are static and are not power constrained. The fixed WMN network architecture inherits dependency on the routing paths, as a set of MRs is connected to certain MRs and solely depend on them for forwarding the traffic [9]. Any malicious behavior would result in drastic performance degradation.
- In a MANET, it is extremely difficult to establish a-priori trsut relationship between two nodes in a distributed manner. On the contrary, in a WMN, authentication authority can be

conveniently deployed on IGW (e.g., AAA server). But, as the network size increases the authentication overhead should not become excessively high.

Thus, while designing any security protocol for WMN, the above mentioned characteristics of WMN should be kept in mind.

1.4 Motivation

The healthy functioning of WMN is contingent to the presence of benign routers. Unlike cellular network or WLANs where the backbone infrastructure is managed by a single trusted entity, community based WMNs are formed by group of independent MRs that could be managed by different entities [4]. Therefore, establishing *a-priori* of trust is a tough challenge. As MRs are typically deployed on rooftops, buildings, poles, towers, traffic signals, even an authenticated node can be easily tampered by an adversary, paving way for foul play [3]. It is economically infeasible to incorporate tamper-proof hardware for MRs [3] [5]. Hence, an attacker, if unnoticed, can cause jeopardize and cause total havoc to the operation of WMN. Unfortunately, the current thrust of research in WMNs, is primarily focused on developing multi-path routing protocols; and security could be said to be very much in its infancy.

Owing to its hierarchical architecture, security issues in WMNs are multi-dimensional. As the MRs form the backbone infrastructure, highest level of security is required here to provide a reliable and guaranteed service to the attached mesh clients. A malicious attacker can disrupt the operation of WMN in several ways. These attacks can be typically classified as: active attacks and passive attacks. An active attack is conducted to intentionally disrupt the network activity (Flooding: Denial of Service (DoS) attack, Tunneling: Wormhole attack, Blackhole attack: Suction of packet towards attacker with false route advertisement, Impersonation: Man-in-the-middle attack). A passive attack, like a selfish node attack, eaves-dropping is conducted with a

selfish motivation and does not cause any damage to the network (intentionally). In this dissertation, we mainly focus on the security threat posed by DoS attack and selfish MR; and concentrate on ways to detect and combat these attacks. We also focus on the problem of detecting the source of the DoS attack by using a technique of traceback. We also investigate the detection of these attacks by developing an Intrusion Detection System based on intelligent machine learning techniques.

A second level of security measure needs to be taken for the mesh clients so as to prevent rogue users, prevent aggressive users, and provide end-to-end security for a secure communication. Once, a reliable and strong backbone is built, it is easier to handle end user security issues. It is important to ensure that a good mesh client does not join a rogue MR. Similarly, a MR should prevent a malicious mesh client from freely enjoying the network resources and should be able to bill the user for utilizing the network services. Hence, we develop a lightweight authentication protocol to mutually authenticate single/multihop mesh clients and the MRs. But, authentication by itself is an insufficient defense to block intruders. Hence, we employ various attack prevention tools to protect the mesh backbone.

Thus, though there is a plethora of schemes in literature that address various security issues (like selfishness, DoS attacks, and route attacks) in ad hoc network, they cannot be directly applied in WMN due to their unique architecture and traffic characteristics. As seen from our previous works [9]-[14], we are motivated to analyze potential security threats on WMN and develop a comprehensive, distributed and a scalable solution to safeguard its operation.

1.5 Organization of the Dissertation

The remainder of this dissertation is organized as follows. We first examine different active attacks and passive attacks on the mesh backbone and propose defense mechanism against such

attacks. Then, we examine the problem of developing an efficient light weight authentication protocol.

In Chapter 2, we discuss the vulnerability of WMN architecture to DoS attack and present a cache based defense at the MRs to identify flooding style DoS attacks [11]. We introduce a *most frequently used* cache mechanism to identify and raise an early alert to curb such attack flows. We drop the identified attack flow along the forwarding routers and avert any potential performance degradation. We illustrate the effectiveness of the above mechanism by performing simulations using ns-2 [15].

The best antidote against DoS attack lies not only in taking preventive measures, but also in identifying the true origin of the attacker. Hence, in Chapter 3 we present a novel technique based on MAC identifier to elegantly perform hop-by-hop traceback [13]. We use OPNET [16] as the simulation tool and modify its wireless module to incorporate at each node, a trace table. We also illustrate the effectiveness of our scheme in the attack path reconstruction through simulations and analyze the capability of our proposed scheme in handling different styles of DoS attacks.

Next, in Chapter 4, we study the problem of selfishness in multi-channel WMNs. We highlight the inadequacy of credit/reputation based schemes in promoting cooperation in WMN. We present the distinct motivation for selfishness in WMNs and describe the selfish adversary model of a MR in WMN that exhibits packet dropping, channel riding, radio selfishness and misreporting. We develop a distributed policing architecture called D-POLICE that employs Sink Agents (SAs) to infer selfishness of MRs by gathering traffic reports from all MRs [10]. We develop a system of checkpoints is used by a SA to identity any inconsistency in traffic reports, which are further used to identify the selfish MR. We then study the effectiveness of our

scheme through simulations using ns-2 [15] which reaffirm that D-POLICE successfully prevents performance degradation even in a largely compromised network.

In Chapter 5, we proceed to investigate the detection of a route disruption attack called malicious route flood, in which a malicious node performs frequent unnecessary route discoveries to unknown destinations. We propose to use a perceptron training model as a tool for intrusion detection [14]. We train the perceptron model by feeding various network statistics and then use it as a classifier. We illustrate using an experimental wireless network (ns-2) that the proposed scheme can accurately detect route misbehaviors with a very low false positive rate.

In Chapter 6, we focus on the problem of providing second level of security services to the mesh clients. We outline a light weight authentication protocol called Merkle Tree based Mesh Authentication Protocol (MT-MAP) for mutually authenticating mesh clients and the MR. We employ inexpensive hash operations and avoid the use of public cryptography on the resource constrained mesh clients. We also present a security analysis to prove the robustness of MT-MAP against impersonation and replay attacks.

Finally, Chapter 7 concludes this dissertation with significant inferences and open challenges for future research.

1.6 Key Contributions

The main contribution of our work can be summarized as follows:

- Provided a comprehensive analysis of potential security threats in WMNs and developed scalable and distributed solutions to ensure the safe operation of WMNs.
- Simulation based demonstrations of the impact of attacks (DoS attack, selfish node attack) on the performance of WMNs and the effectiveness of the proposed solutions in detecting and curbing these attacks.

- Investigated the distinct motivation for a MR to act selfishly in a WMN and illustrated its negative impact on the system performance.
- Proposed a simple cache based method to combat DoS attacks such that it is independent of the underlying queuing mechanism at the routers.
- Developed a MAC identifier based traceback mechanism that is scalable and feasible to realize in terms of memory and bandwidth requirements.
- Developed a novel Intrusion Detection System based on intelligent machine learning technique for WMNs.
- Developed a light weight authentication protocol for mutually authenticating mesh clients and MRs.

Chapter

2. Active Cache based Defense against Denial of Service Attacks

2.1 Introduction

The MRs constitute the skeletal backbone of WMNs and highest level of security is required to protect them. As the WMNs run in the unlicensed 2.4 GHz band, a malicious attacker can easily perpetrate a Denial of Service (DoS) attack. The main goal of DoS on a WMN is to deplete the network resources by sending a flood of packets from one or more compromised mesh clients or mesh routers. Unlike a responsive flow like TCP that adapts its rate according to the end-to-end congestion; a DoS attack flow incessantly pumps traffic into the network to cause congestion. It also leads to the starvation of other innocent flows.

A DoS attack poses serious threat to WMNs as it can be easily launched by an attacker by generating spoofed source address. A flooding style DoS attack is difficult to stop by the victim MR. The very architecture of the IP layer is such that any node can originate traffic towards any

other destination and the destination node is helpless in stopping the traffic before it reaches the network. The fundamental characteristics of WMNs also facilitate an easy ground for conducting DoS attack. Firstly, the bulk of the traffic in a WMN is predominantly directed towards the IGW which leads to heavy congestion near the IGW. Thus, it is easy to conduct a DoS attack on these over-burdened MRs which would further degrade the network performance. The proximity of an attacker to the IGW, determines the impact of DoS attack on the network. Secondly, a DoS attack from a near by mesh clients pose a serious threat to longer hop flows which already receive poor service due to the poor performance of IEEE 802.11 in a multi-hop environment.

There are multitude kinds of DoS attacks like Smurf, SYN, and Teardrop attack that vary in their degree of sophistication [17]. A Distributed DoS (DDoS) attack is a more virulent model of DoS attack; where an attack is launched synchronously from several compromised innocent hosts piloted by a remotely hidden attacker. In a SYN attack, the attacker exploits the TCP's 3-way handshake mechanism by opening many fake connections with the server. The attacker then refuses to send the TCP ACK, thereby withholding the resources at the server indefinitely. In a Teardrop attack, the attacker exploits the fragmentation process at a router, by inserting an incorrect offset, thereby causing improper re-assembly. A Smurf attack is a more sophisticated attack, in which an attacker sends forged ICMP (Internet Control Message Protocol) echo request, spoofed with the source address of an innocent victim to large number of innocent nodes in the network; which in turn flood the victim unknowingly by sending ICMP reply packets.

In this research, we propose a novel scheme that addresses the problem of combating DoS attacks by raising an early alert. We employ a simple method that maintains minimal state information at each router and functions independent of the underlying queuing mechanism at the routers. We employ two components at each router: *Active DoS attack detection module* and

DoS attack regulator module. The active DoS attack detection module identifies the high bandwidth attack flow based on *Most Frequently Used* (MFU) cache discipline. The DoS attack regulator module employs a trace notifier that applies rate limit along the upstream routers through which the attack flow passes. Hence, it effectively prevents the starvation of other innocent flows. We implement the proposed scheme just above the MAC layer and do not require any change in the MAC firmware of the routers. Figure 2.1 illustrates the physical positioning of our proposed scheme.

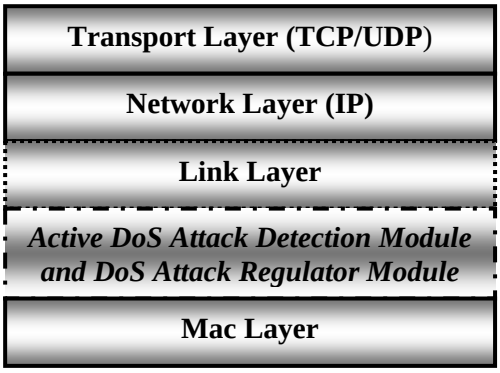


Figure 2.1: Location of our proposed module

The remainder of this chapter is organized as follows: In Section 2.2, we present various existing schemes to detect and control DoS attack flows and then highlight the need for a new scheme in WMN. In Section 2.3, we present the design goals and the architecture of our proposed cache based scheme. In Section 2.4, we provide a comprehensive performance evaluation of our scheme. We finally, conclude with the summary of our scheme and indicate some future research directions in Section 2.5.

2.2 Related Work

In this section, we outline related work in the area of combating DoS attacks and analyze its applicability for securing WMNs from DoS attacks.

Several schemes have been proposed in the literature to combat DoS attacks. The primary line of defense against DoS attacks can be broadly classified into few broad areas like: Filtering scheme, traceback schemes, and rate limiting schemes. The first two schemes defend victim of an attack and a rate based scheme defends the network itself from experiencing a DoS attack.

Ferguson et al. [18] propose to use ingress filtering as a preventive measure against spoofing of IP address. Park et al. [19] propose a distributed packet filtering mechanism to detect source IP address spoofing using BGP (Border Gateway Protocol) routing information. Yaar et al. [20] propose Pi marking scheme to enable the victim to detect spoofed source IP address. Firstly, these schemes are ineffective against bandwidth flooding attacks which affect routers upstream from the victim. Secondly, they do not prevent compromised machines with valid source address in conducting DDoS attack and incur considerable overhead on the ISP. A different style of combating DoS attack involves detecting an attack in progress and constructing the path of attack, leading to the real attacker known as traceback [12]. The trace data can be probabilistically carried in-band within the 16-bit identification field of the IP header of the packets as proposed by Savage et al. [21]. Alternatively, the path information can be logged in the routers for conducting a hop-by-hop traceback later as suggested by Snoeren et al. [22]. Though, these schemes trace the origin of an attack with sufficiently large number of captured attack traffic; they are ineffective in preventing the attack traffic from entering the network itself.

A rate limiting scheme actively monitors the network for high bandwidth attack flows and inhibits network performance degradation before it is too late. It can be further sub-divided based on their motivation as: Buffer management schemes and network congestion control schemes. There are several buffer management schemes in the direction of controlling unresponsive aggressive flows. RED (Random Early Detection) [23] and FRED (Fair RED) [24] are some

flow-based scheme to control bursty flows in a wired network. They randomly drop packet belonging to different flows. But, the idea of randomly dropping packets belonging to different flows does not ensure that an attack flow is controlled to a sufficient extent. Suter et al. [25] propose a scheme called Longest Queue Drop (LQD) that on congestion drops packets from flows that occupy largest buffer space. As it necessitates per-flow state information, it incurs considerable overhead.

A network congestion control in general alleviates the impact of DDoS attacks. Mahajan et al. [26] propose an Aggregate Congestion Control (ACC)/pushback mechanism against high bandwidth aggregates by rate limiting group of similar packets and carrying the rate limit backward towards the source. But, forming an appropriate congestion signature for the aggregates is very challenging. Yaar et al. [27] propose Stateless Internet Flow Filter (SIFF) which enables end-host to selectively stop flows. In SIFF, the sender establishes privileged channels through a capability exchange with the receiver. As the recipient is empowered to stop the traffic by not forwarding capabilities, it effectively mitigates DDoS attack flows. But, it assumes the existence of a secure network environment. In [28], we have illustrated how a cache based mechanism can be used at each MR to control shorter hop length aggressive flows in a WMN. It assumes a benign environment with only unresponsive aggressive flows, while a defense against high bandwidth DoS attack requires more stringent control. Mirkovic et al. [29] propose an automated tool called D-WARD, which is deployed at strategic routers in the Internet to detect DoS attacks based on deviant traffic-flow measurements. Stoica et al. [30] propose Stateless Fair Queuing (SFQ) mechanism that labels the incoming packets at the edge routers based on per-flow arrival rate information. The core routers use the embedded labels in the

packet to drop packets probabilistically for a fair bandwidth distribution. The applicability of SFQ and D-WARD in a WMN, where mesh clients are attached to every MR, is arguable.

In contrast to the above schemes, we present a cache based defense at the MRs that maintains minimal per-flow state information and at the same time empowers the MRs to control attack flows. As the attack flow is throttled at every MR progressively till the source of the attack, it is effective in preventing the attack flows from entering the network.

2.3 Overview of the Scheme

In this section, we first outline the design goals and then provide an overview of our proposed scheme.

Our main goal is to develop a detection scheme that would unequivocally detect attack flows that continuously occupy an unfair buffer space at the MRs. It is crucial to penalize these flows, as admitting such flows would result in no buffer space for other innocent flows when they eventually arrive. We regulate the traffic rate of such incessant flows using our cache based defense mechanism. We pre-empt such misbehaving flows from the IFQ (Interface Queue) of the MRs by dropping them. In a WMN, a distant flow receives low throughput owing to the multi-hop communication. Hence, an attacker can cause total havoc to the far away flows by originating an attack on the MRs from a near-by source. The innocent flows in a network compasses HTTP transfer requests issued from the mesh clients (that are short lived low-bandwidth flows), TCP flows that respond to congestion (that are long-term high-bandwidth flows), and low data rate UDP flows (telnet applications).

Our algorithm consists of two main sub-parts: *Active DoS attack detection module* based on most frequently seen cache discipline and *DoS attack regulator module*. Before we proceed to

describe our scheme in detail, we first introduce the data structures and the variables used at each MR for maintaining the per-active-flow state information.

2.3.1 Data Structure

We maintain a cache table of fixed size at each MR that contains information on all the active flows routed through it. We maintain the following information in the cache table:

*Packet Signature**:*** We uniquely identify the packets of a flow according to the hash of its packet signature, which consists of Type of Service (ToS), source input interface identifier, destination address, and IP protocol type and avoid using the source address as it could be spoofed.

Frequency_count: This gives the number of packets that have been successfully serviced (i.e., that found a space in the IFQ) by this MR node in the current time window.

In addition to the above per-active-flow information, we also maintain a node variable called *drop_probability*. This probability is used to determine whether a packet from a potential attack flow should be admitted into the queue or not. The *drop_probability* is incremented multiplicatively and decremented linearly. Further description of this variable is deferred to the next subsection.

2.3.2 Active DoS Attack Detection Module

Before a rate controller can be applied on the attack traffic, it is important to detect potential attack flows. This module helps in the detection of such high-bandwidth flows that are hazardous to the network.

We propose an active cache based mechanism to identify high-bandwidth flows. The cache table maintains a record of *most frequently seen* flows. Whenever, a packet arrives at a node, its packet signature is compared with the existing packet signature in the cache. If not found, an

entry is created for this new flow; else if the flow is already present in the cache, its frequency counter is incremented. When the cache becomes full, the cache entry with the least count is pre-empted to make room for the new flow. In this way, the cache maintains a list of potential high-bandwidth flows. The attack packets arrive in bulk to flood victim MR and are guaranteed to be present in the cache with a large frequency count.

We can distinguish between a DoS attack and link congestion using application level information [31]. It is possible that our scheme can misclassify bursty flows as an attack flow. In order to avoid such false positives, we compare the frequency count of this flow with the frequency counts of the other flows in the cache. If the frequency count of this flow is greater than half of the maximum frequency count in the cache, the bursty flow is misclassified as an attack flow. But, this labeling is not permanent as the bursty flow would behave normally in another time frame and would no longer be treated as an attack flow. However, an attacker steadily pumping traffic into the network would be consistently classified as an attack flow for the entire duration of the attack.

2.3.3 Active DoS Attack Regulator Module

After the attack traffic is identified, an alert is sent to the DoS attack regulator module to throttle it. The module controls the admission of an attack flow based on a certain dynamically tuned *drop probability*.

Every MR is associated with an initial *drop probability* of 0. In the presence of attack flows, a packet belonging to the innocent flows always finds a full buffer due to which it is forcefully dropped at the IFQ. Thus, whenever a packet belonging to an innocent flow is dropped, we increment the *drop probability* at this MR. Eventually when a packet belonging to the attack traffic arrives it is dropped even if there is space in IFQ depending on this *drop probability*.

Unlike attack flows, an innocent flow has a larger inter-arrival time and hence is not affected greatly by this *drop probability*. As more and more innocent packets are dropped during the attack, the *drop probability* is correspondingly increased, which in turn controls the attack flow.

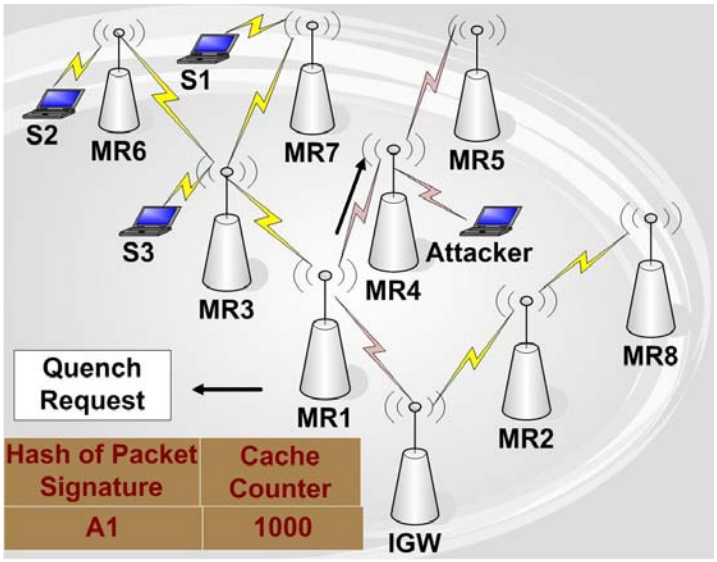


Figure 2.2: Defense against DoS Attack

Once the *drop probability* reaches maximum value DP_{max} and remains so for $T_{monitor\_drop}$, we use a trace notifier that sends a *Quench request* to its upstream neighbor pumping the attack packets. This *Quench request* is sent hop-by-hop backward until the attack source, expunging the attack traffic at each hop as shown in Figure 2.2.

Thus, we save valuable bandwidth near victim by dropping the packets early at the upstream MR. We use a multiplicative increase of drop probability to quickly control an imminent attack from overwhelming the network. Thus, we set the initial value of *drop probability* as 0.05. We however, employ a linear decrease of *drop probability* to ensure that once the attack traffic is effectively throttled, it does not affect the innocent flows. At the same time, it is critical to control the attack flow for an appropriate amount of time and a linear decrease meets this goal. Else, a multiplicative decrease can give undue advantage for the attack flow to resurrect. The step size for the decrement of *drop probability* is set to a small value of 0.05 in our simulation.

The *drop probability* is non-zero as long as the attack flow affects an innocent flow. To handle such instances, we look at the time elapsed since the last increment of *drop probability* and if it is greater than a threshold, we steadily decrement the *drop probability*.

2.4 Performance Analysis

In this section, we evaluate the performance of our scheme using ns-2 (version 2.29). We first illustrate the degradation in the network performance due to the DoS attack traffic and prove the effectiveness of our scheme in improving the aggregate throughput of innocent flows. We then show how our scheme helps in a fair allocation of bandwidth for all the flows in the network. We finally determine the optimal choice of initial value for the *drop probability* through simulations.

We consider a simple IEEE 802.11s based mesh network with 9 MRs deployed in a random fashion as shown in Figure 2.2 (with each MR serving on average 2-3 mesh clients) in the previous section. One of the MR is designated as IGW that provides internet connectivity to other MRs. All mesh nodes and mesh clients communicate using IEEE 802.11 DCF operating at 11 Mbps with RTS/CTS enabled. The two-ray ground model has been used as the radio propagation model with a transmission range of 250 m and carrier sensing range of 550 m. All the traffic is started from the clients under the MRs, and is then aggregated at the corresponding MR and forwarded to the IGW. The simulation duration is set to 150 secs. Without loss of generality, we assume a constant packet size of 1024 bytes for all the UDP flows. The IFQ size is set to be 50. We use the following values for the threshold: $DP_{max} = 0.8$, $T_{monitor\_drop} = 5$ secs. We conduct a UDP flow attack [32] in the above set up in which an attacker generates a high volume of spoofed traffic towards the victim. As UDP has no intrinsic congestion control mechanism,

the attack flow can easily overwhelm the network, thereby depriving internet services for other legitimate users.

2.4.1 Instantaneous Throughput

We first illustrate the effect of an attack on the performance of our scheme using a simple scenario; where in an attack source (a mesh client from MR4) pumps traffic towards the IGW at the rate of 1.2 Mbps (starts at *t = 30* secs). We start a set of three innocent flows from the mesh clients MR3, MR6, and MR 7. *Flow-1* originates from MR7 at the rate 200 Kbps and is started at *t = 0* sec. *Flow-2* originates from MR6 at the rate of 100 Kbps and is started at *t = 10* secs. *Flow-3* originates from MR3 at the rate of 300 Kbps and is started at *t = 20* secs. The three flows are routed via victim MR1 to the IGW as shown in Figure 2.2 in the previous section. Any flows originating from the clients under MR2 or MR8 remain unaffected as they do not lie in the path of the attack and hence we do not consider them in our performance evaluation.

From Figure 2.3, it can be clearly seen that as soon as the attack flow is started, other flows (*Flow-1*, *Flow-2* and *Flow-3*) receive very low throughputs. On the other hand, when we employ our DoS attack regulator, we can observe that (see Figure 2.4) the high rate attack flow is effectively controlled.

Now the throughput of the attach flow is drastically cut down from 1.2 Mbps to 250 Kbps. As soon as the innocent flows start suffering from excessive packet drops, the MR increases the drop probability for the attack traffic. And the trace notifier effectively curbs the attack traffic close to its source. Thus, the innocent flows are effectively shielded from the attack traffic.

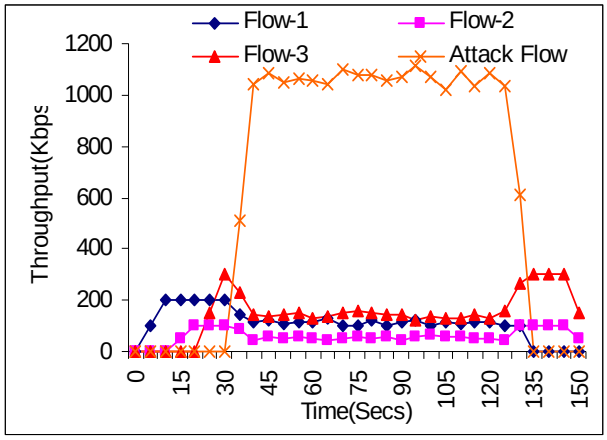


Figure 2.3: Instantaneous Throughput of flows in the default case

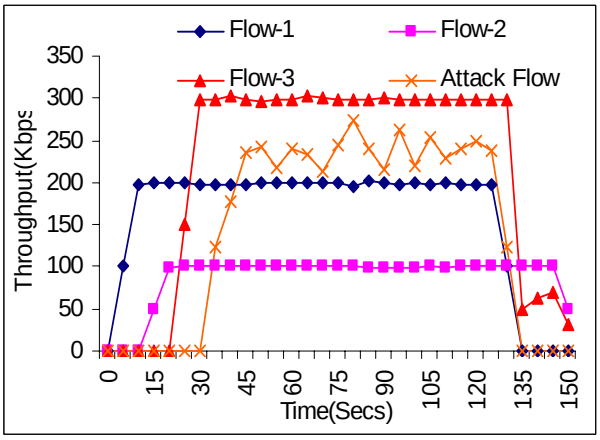


Figure 2.4: Instantaneous Throughput of flows in our scheme

2.4.2 Bandwidth Allocation

We next analyze the bandwidth allocation in the congested link between MR1 and IGW. We consider a similar configuration of flows as in the previous analysis. As the attack traffic is sent through MR1, the buffer at MR1 is pre-dominantly filled with the frequently arriving attack traffic. Usually, routers employ drop tail mechanism in which a new packet is added to the IFQ only if there is space. The normal network traffic being low-rate in nature (e.g., internet browsing) has a higher inter-arrival time, as a result of which it often fails to find space in the congested buffer of MR1 and is frequently dropped. Thus, in the default case, as seen from Figure 2.5, the attack traffic occupies major chunk of the link bandwidth (68%) between MR1 and IGW and enjoys high throughput (100% throughput as seen from Figure 2.3). However, our cache based mechanism continuously monitors the queue build up at the MR1 and increases the *drop probability* at this MR as the innocent traffic is frequently dropped at the IFQ. Thus, our attack regulator module moderates the buffer occupancy of the attack traffic and ensures a fair allocation of the link bandwidth.

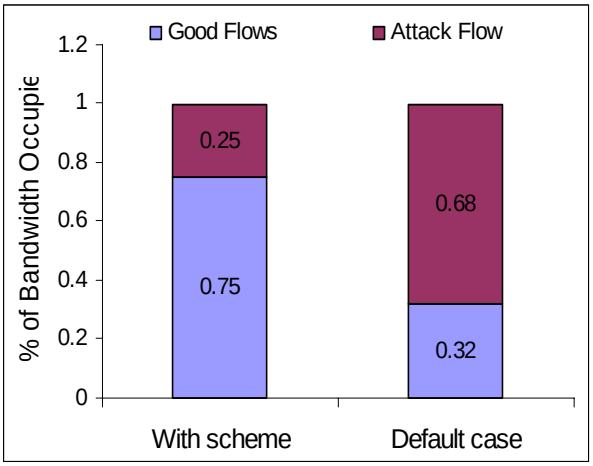


Figure 2.5: Bandwidth allocation at the congested link (MR1)

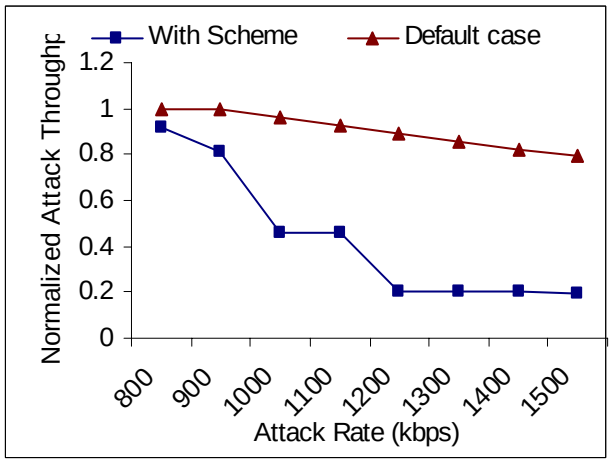


Figure 2.6: Normalized Throughput of attack flow

In the presence of our scheme, the innocent flow gets 75% of the link bandwidth and the share of the attack traffic is reduced to 25%.

2.4.3 Normalized Aggregate Throughput

We also evaluate the effectiveness of our scheme by measuring the normalized throughput of flows, which is the ratio of the throughput obtained to the offered load. We compare the normalized throughput of flows in the default scheme and when using our scheme for the attack flow as well as the innocent flows. We see from Figure 2.6 that in the default case, the attack traffic enjoys maximum throughput. But, when our scheme is applied, it is effectively throttled by the controller module. It is important to note that our cache based scheme rigorously regulates only those flows that are generating excessively high UDP traffic. From Figure 2.6, we see that as the rate of attack traffic is increased from 800 Kbps to 1500 Kbps, the throughput of attack traffic decreases. While, high rate traffic of 800 Kbps is hardly throttled, attack traffic of 1.5 Mbps is drastically cut down by 70%.

We next examine how our scheme protects the innocent flows in the network. We start one attack flow from MR4 (at 1500 Kbps) and about 6 innocent flows (at rates- 50 Kbps to 300

Kbps) from various MRs. The sources are picked at random from the mesh topology. We consider the aggregate throughput of all innocent flows and normalize them with respect to the total offered load. From Figure 2.7, we see that our cache based defense provides a sustained throughput close to 1 irrespective of the attack rate; while in the default scheme the normalized aggregate throughput steadily reduces (from 0.86 to 0.46) as the attack rate increases.

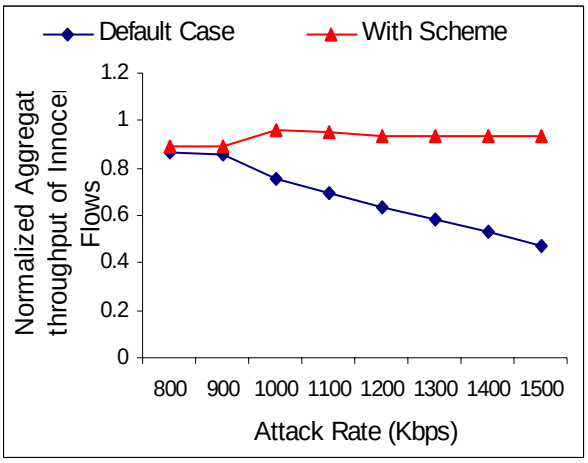


Figure 2.7: Normalized Aggregate Throughput of innocent flows

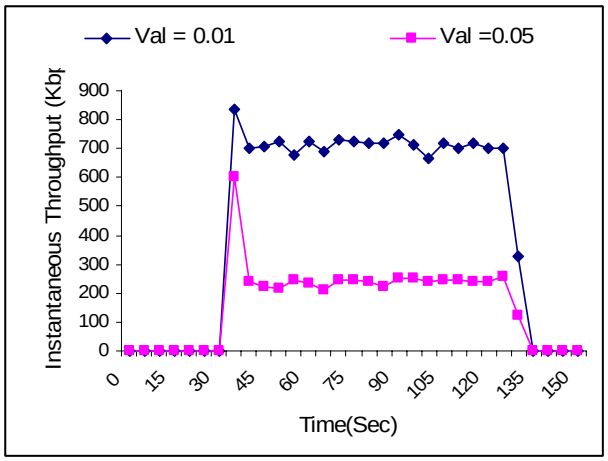


Figure 2.8: Impact of initial value of drop probability on attack regulation

2.4.4 Impact of the Initial Value of Drop Probability

We now study the impact of the initial value of *drop probability* on the performance of our scheme. We consider a similar configuration of flows as in Section 2.4.1. Figure 2.8 shows the impact of different initial value for the *drop probability* on the attack traffic. We see that when the *drop probability* is set to 0.01, the reaction time in controlling the attack is very large (close to 2 secs) and the attack traffic is not regulated effectively. A value of 0.05 for the *drop probability* is an ideal choice with moderate reaction time. It also quenches the attack traffic effectively.

The choice of initial value of drop probability is important for the performance of our scheme. An optimal value should be chosen so that the attack traffic is efficiently controlled. At

the same an innocent traffic should not be controlled excessively by mistake and hence a high initial value should not be chosen. On studying the performance of the network with various values, we conclude that an initial value for 0.05 is optimum.

2.5 Conclusion

A DoS attack constitutes a huge umbrella of potential security threats against WMNs hindering their widespread deployment. We propose a cache based method equipped with a trace notifier that helps in securing the network against DoS attacks. We employ a simple method that maintains minimal state information at each router and functions independent of the underlying queuing mechanism at the routers. Simulation results show that our scheme effectively mitigates the effect of DoS attacks by effectively filtering the attack traffic at every intermediate router. In future, we plan to investigate the performance of our scheme in handling DDoS attacks.

Chapter

3. Low Cost Reliable Scheme for Detecting DoS Attackers

3.1 Introduction

The plug-and-play architecture of WMNs makes them a vulnerable target for attackers. A Denial-of-Service attack (DoS) is a potential threat on the backbone MRs, as a malicious intruder can deplete the network resources by overwhelming it with excess traffic. This would inordinately affect the multihop flows, traversing from distant sources, could result in wastage of network resources and could cause total havoc to the system. The attack is even more precarious; especially if the MR is located near the IGW, as these nodes are in charge of forwarding the bulk of network traffic. We discussed in Chapter 2, the DoS attack poses a serious threat to WMNs primarily due to the fact that the attacker can go untraceable by generating spoofed source IP address. Thus, in this chapter, we explore different ways to detect the exact source of a DoS attack.

The best antidote against DoS attack lies not only in taking preventive measures (such as Ingress filtering) ***but also in identifying the true origin of the attacker*** *and in blocking further occurrences of such incidents*. This boils down to the problem of traceback. Traceback involves a forensic analysis of the traffic and identifying the exact source of a packet across the network. Its Internet counterpart called IP Traceback, is a popular way of tracing the source of a packet across the Internet. It is challenging to perform such a traceback in a wireless ad hoc network due to its dynamic topology. But, it is easy to perform in a WMN due to its relatively static topology. The rampant spoofing of source address by the attacker in the packets increases the difficulty in conducting traceback. The attackers, who in general, enjoy their anonymity can now be implicated by traceback and penalized for their malicious act. There are several advantages associated with conducting traceback. As the identity of an attacker could be exposed by traceback, the attacker would think twice before performing a DoS attack. Traceback also helps in providing a better implementation of filtering rules, as the counter-measures can be taken near the originating point of the attacks.

In order to efficiently identify the source of an attack, we propose in this chapter a novel traceback mechanism using MAC identifiers. As IP address spoofing is easy to perform and difficult to identify, we use MAC identifiers for elegantly tracing back to the correct attacker. For every packet it forwards, it pro-actively records or updates a two tuple entry in its trace table for each MR. The MR records for every unique source IP address of the packet, the incoming MAC address on which the packet has been received and the source IP address of the packet. When a request for traceback arrives, the attack path is reconstructed hop–by-hop, by querying each router for the attack packet and retrieving the corresponding upstream router's MAC address from which it originally got forwarded. The querying proceeds backward, until it reaches

the true source. It is important to note that spoofing of MAC address of every intermediate forwarding router is nearly impossible for an attacker. Thus, our scheme can efficiently construct the attack path for any malicious packet.

The rest of this chapter is organized as follows. In section 3.2, we give a brief insight on the science of traceback and various existing IP traceback techniques and also highlight the need for an efficient traceback scheme in WMNs. In section 3.3, we describe our novel technique of MAC identifier based traceback and describe its implementation in OPNET [16]. In section 3.4, we illustrate the effectiveness of our scheme in conducting traceback and present simulation results. We finally conclude with the summary of the work and some future research directions in section 3.5.

3.2 Overview of Existing Traceback Schemes

In this section, we present a generalized attack model and give a brief overview of traceback. Traceback is more popularly known as IP traceback in the internet and there exist plethora of schemes in literature to trace the source of DoS attack in a wired network [33].

There are several challenges associated with conducting traceback [12]. The IP header in all packets contains the source IP address. Unfortunately, due to the limited security features in TCP/IP, it's very easy to spoof a source address by any attacker. Hence, a simple traceback that looks for the source IP address of each packet for obtaining its origin is useless. Routing in IP depends only on the destination address and there is no authority in the internet that validates the source address inscribed in a packet. Due to this stateless nature of internet's routing mechanism, it becomes incumbent on researchers to design a traceback scheme using other guidelines. At one end of the network, an attacker that is responsible for initiating an attack against a distant victim MR could be buried behind several other entities. Hence, it depends upon the intelligence of a

traceback scheme in identifying the true source of attack. A generalized attack model [34] is shown in Figure 3.1. It includes all possible disguise an attacker might use such as a stepping stone/ zombie / reflector.

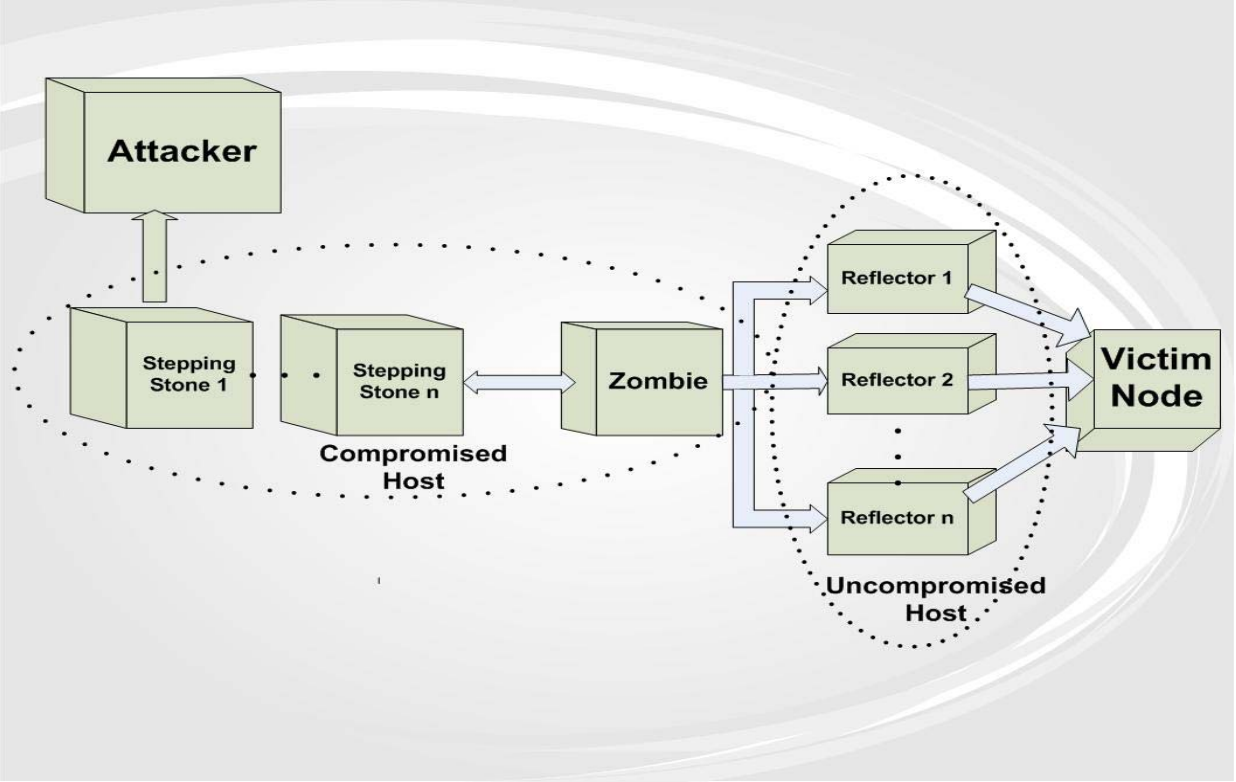


Figure 3.1: Generalized Attack model

Apart from spoofing source address, a more potent attacker might be masked behind *stepping stones,* which are compromised hosts that act as laundering agents. The stepping stones are engineered in such a way that it overwrites its source address on the outgoing packet headers and also applies some packet transformation to conceal the true origin. Most of the traceback schemes are capable of tracing only till the stepping stone [12]. The stepping stones can be identified using specialized techniques only that looks for causality relationship between packets entering and leaving a host [35]. An attacker can also conduct an attack through a *zombie node* by indirectly communicating via stepping stones or by directly installing Trojan programs triggered to execute after certain delay and hide its association. Like a detonating time bomb, a

single command from the attacker is later sufficient to start the attack. In the chain of disguise used by an attacker, the last one is the *reflector node*. Reflectors are innocent nodes that readily send response packets. Large number of zombie nodes with its packet source spoofed as victim's IP address, target a set of reflectors. The innocent reflector send their response and cumulatively flood victim's network. The malicious zombie may initiate a TCP SYN flooding. The SYN packets that are issued by the zombies to a set of reflector nodes contain their source address as the spoofed IP address of victim. The reflectors would then send TCP ACK, second step in a 3 way handshake mechanism and create a deluge of packets in victim's network. The use of reflectors in DDoS attack greatly complicates conducting any traceback [36].

Various existing traceback schemes can be primarily classified into reactive or pro-active schemes, depending on when a traceback is initiated. A *reactive approach* is the one that carries out the traceback on the fly once an attack is detected. In a reactive scheme, traceback is executed in response to an ongoing attack, like a stimuli-response mechanism. The attack is detected by an administrator or an Intrusion Detection System as in Controlled Flooding [37]. A *proactive approach* takes a different orientation in pinpointing the source by proactively recording and logging the traffic packets as they flow through the network. These records are useful indicators for victim in path reconstruction to the actual source and provide timely response on the occurrence of an attack. Probabilistic Packet Marking (PPM) [21] and Source Path Isolation Engine (SPIE) [22] are some pro-active approaches.

The traceback scheme can also be classified depending on where the traceback information of the packet flow is recorded. The resources required for storing this trace data can be distributed across various network components which are as listed as follows:

- On the forwarding routers:

  The packet flow is recorded at various routers in the network and the attack path is extrapolated by examining the router logs. This would necessitate the routers to record information for every incoming packet; which would consume colossal amount of space on the routers and result in unwieldy logs, taking in to account the speed of today's links.

- Inside the packet:

  Each router stamps its IP address probabilistically or deterministically on some special unused fields like ID/Option field of IP header of the packet. As the trace data is carried in-band within the packet, there is limited space available for storing the path of the packet.

- Emitting a separate trace packet:

  Each intermediate router emits a separated trace packet probabilistically. These trace packets like ICMP trace packets collects information on all the routers through which it traverses, before it reaches the destination. The traceback incurs additional overhead in terms of the network bandwidth.

We concentrate on developing a traceback mechanism by placing the burden of logging on the forwarding routers. But, even this approach has its limitations and requires an efficient design. Input debugging is a feature supported by some routers that can identify for a given packet of interest; the incoming link on which the packet arrived. This, however, has an adverse effect on the network performance due to the overhead incurred at the router. SPIE [22] is another solution that reduces the storage requirement by several magnitudes through the use of space efficient data structure called Bloom filters that caches the hash digest of forwarded packets. But, SPIE can trace only packets delivered in the recent past. Also, SPIE is effective in tracing only till the border router which has the attacker and cannot pinpoint the attacker. To

solve this, Baba et al. [38] propose to track the precise location of the attacker by maintaining the incoming MAC address in the router's buffer along with other critical information of the forwarded packets. This scheme however, faces enormous storage requirement at each router as it pro-actively stores information in the router's buffer for every incoming packet. A typical core router would need memory in the order of 1 terabyte to log packets for a minimum length of 5 minutes. Hence, it is not pragmatic to implement a per packet traceback scheme in a real world environment. We propose a modified scheme, which at each router maintains only a two-tuple entry for a given type of attack packets. As most of the DoS attacks (except for few like Teardrop attack) involve a deluge of packets, our scheme guarantees to track the source of the attack by just recording the incoming MAC address and Source IP address of the attack packet.

Apart from the above discussed problem, additional constraints are placed in a wireless environment due to its unpredictable routing topology and limited bandwidth. Hence, there is little research addressing traceback in wireless network. Thing et al. [39] study the feasibility of applying IP traceback schemes like PPM, ICMP Traceback, and SPIE in a Wireless Ad hoc network and conclude that these traceback schemes incur considerable overhead in a wireless network and their performance depends on the network size and the routing protocol. Kim et al. [33] propose an on-the-fly traceback scheme based on small world concept that uses traffic volume technique to trace the path of an attack.

When compared to Wireless Ad Hoc Networks, WMNs ease the constraints on the traceback scheme due to their relatively static topology. However, a traceback scheme for WMN is challenged with different kind of issues, like scalability. As WMNs serve a large community, it requires a huge amount of memory to support the traceback. Our scheme addresses mainly this

issue of memory overhead and proposes a feasible traceback scheme to trace all the attack paths from the attacker(s) to victim.

3.3 Proposed Architecture of MAC Identifier based Traceback

In this section, we first outline the design goals and the assumptions made for implementing our scheme. We then provide details of the proposed system architecture.

3.3.1 Design Goals

Our main design goal is to develop an efficient and feasible traceback scheme for WMNs. The problem of traceback in a WMN is different from internet based traceback due to several architectural differences as enlisted below.

High Volume Traffic: A WMN envisions providing high bandwidth broadband connections to a large community of users and hence, the estimated traffic volume is very high.

Different Traffic Pattern: The traffic in a WMN is predominantly between MRs and the IGW. The MRs are used primarily as an intermediate hop to forward each other's traffic.

While a logging based scheme in a WMN would require humungous amount of memory to store the traceback data; a probabilistic marking scheme would lead to combinatorial explosion at the attack path reconstruction stage. Thus, though there are umpteen IP traceback schemes in the internet, they cannot be directly applied for a WMN. We need a traceback that should be scalable to handle the huge traffic volume in a WMN. In summary, following are our design objectives:

- The traceback scheme should withstand source IP address spoofing and source MAC address spoofing.
- The traceback scheme should be scalable, as WMNs are envisioned to provide high broadband services to a large community.

- The traceback mechanism should be feasible to realize in terms of memory and bandwidth requirements.
- The traceback scheme should be able to handle mobility of mesh clients as it moves from one mesh router to another.

3.3.2 Assumptions

- We assume the attack packets are coming from malicious mesh clients and propose a scheme to trace the same.
- We assume that an attacker is unlikely to compromise an intermediate router and falsify its entry. End hosts are likely to be easy targets for an attacker to compromise than the backbone infrastructure. The assumption of a secure backbone is unavoidable as admitting the possibility of subverted routers can lead to several false positives.
- We assume availability of adequate physical memory at the MRs which can be utilized for storing the lookup table and executing our traceback scheme.

3.3.3 System Architecture

Our proposed traceback mechanism proceeds as a two step process- identifying the attack path along the routers in the mesh backbone and the attack segment to the mesh client. In our proposed approach, we make use of the MAC identifiers as well as packet digest to detect the exact source of the attack. As discussed earlier, it is not always possible to nab the correct attacker using the source IP address due to the rampant spoofing of source IP address. Thus, we propose to use the MAC addresses of all the intermediate routers that a packet passes through. The advantage in using the MAC addresses is that, we can successfully re-construct the attack path, even when the IP address is spoofed. Though, the attacker can conceal or spoof the sources

MAC address, it is nearly impossible for an attacker to forge the MAC identifiers of the intermediate forwarding MRs. Even if the attacker forges the source MAC address, the traceback scheme helps in tracing hop-by-hop the path of attack packet from victim leading back to the attacker. Figure 3.2 shows a sample attack path constructed from victim MR4. Node *A* in Figure 3.2 is the mesh client serviced by MR1 which initiates a DoS attack on a victim router *V*. Our traceback process reconstructs the attack path as {MR4, MR6, MR3, MR2, MR1, A}.

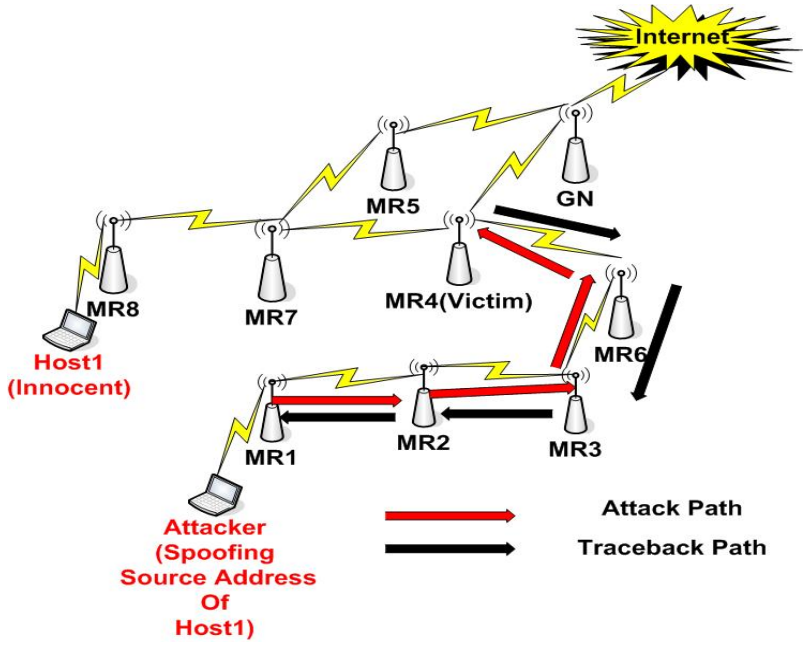


Figure 3.2: Attack Graph

The system configuration consists of the following components that assist in the traceback process:

Intrusion Manager: An IDS agent triggers an alert about the DoS attack and instructs victim router to initiate the traceback process.

Trace Table: Each router maintains a Trace Table that consists of a two tuple entry. A sample snap shot at a router MR4 is shown in Table 3.1. This table is consulted during the traceback.

Table 3.1: Snap shot of Trace Table at MR4

SourceIP Address	Incoming MAC Address	Packet Signature
Host1 (Spoofed IP)	MR6	Null
Host1	MR7	Record an unique Flow Identifier

Each MR, upon receiving a packet examines the source IP address inscribed in the packet and stores it, irrespective of whether or not it is spoofed. It pairs the source IP address with the source MAC address of the incoming packet and stores the tuple in a trace table. As the source IP address could be spoofed, each intermediate router records all the incoming MAC address on which this packet was seen. The multiple incoming MAC address could reflect multiple paths taken by the attack packet due to some re-routing or could reflect two different flows – an attack flow originating from the attacker with a spoofed address of an innocent MR in the network and an innocent flow with correct source IP address originating from the innocent MR.

When a packet with a given source IP address is received from a different source MAC address other than that recorded in the trace table; we say a collision is said to occur. Hence, another entry is created for this new MAC address along with the source IP address. Whenever a clash occurs, a traceback query would have ambiguous paths which would be difficult to resolve without any additional information. To resolve the tie, we begin to record in the trace table some additional information from the packet. This is illustrated in Table 3.1. The first row is the entry recorded upon receiving a flow from the attacker *A* which has a spoofed source IP address of an innocent node *B*. The second row is that of the innocent flow from B. When a packet with source IP address as B is received at MR4 from a different MAC address; a second row entry is created in the trace table of MR4. Henceforth, we record some additional information for this packet flow which will be used later for resolving ambiguities during the traceback.

When we record packet specific information, only a few invariant fields of the IP header of a packet are recorded. Fields that change in transit like time-to-live, header checksum, option field are excluded as they are not unique indicators of a packet flow being originated from a given MR. The invariant fields are chosen such that it would guarantee a distinct differentiation between each flow in the network. Figure 3.3 shows the invariant fields chosen to create packet signature.

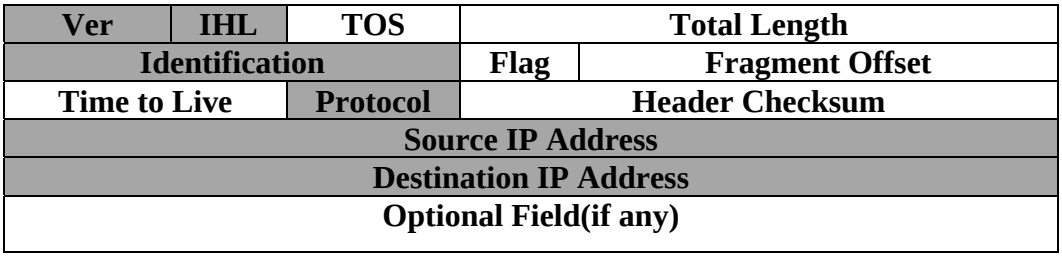


Figure 3.3: IP Header Fields
(Darkened Fields form the packet signature)

The traceback process is started at victim and proceeds backwards from there. When an IDS in the targeted network detects a DoS attack, it raises an alert and victim MR initiates a trace query. The format of the trace query is shown in Figure 3.4. The traceback query consists of the signature of the attack packet, source IP address of the attack packet, source address of the trace request and destination address of the trace query.

Attack Source IP Addr	Attack Packet Signature	Trace Source	Trace Destination

Figure 3.4: Format of Trace Query

The traceback process starts at the router upstream to victim MR (say MR4 in Figure 3.5). The upstream router uses the source IP address provided in the trace request in order to examine its trace table and determine the previous hop MAC address from which the attack packet was received. It then creates and unicasts a new trace query destined to this upstream MR. Thus, traceback proceeds in a hop-by-hop fashion backwards, till the attacker.

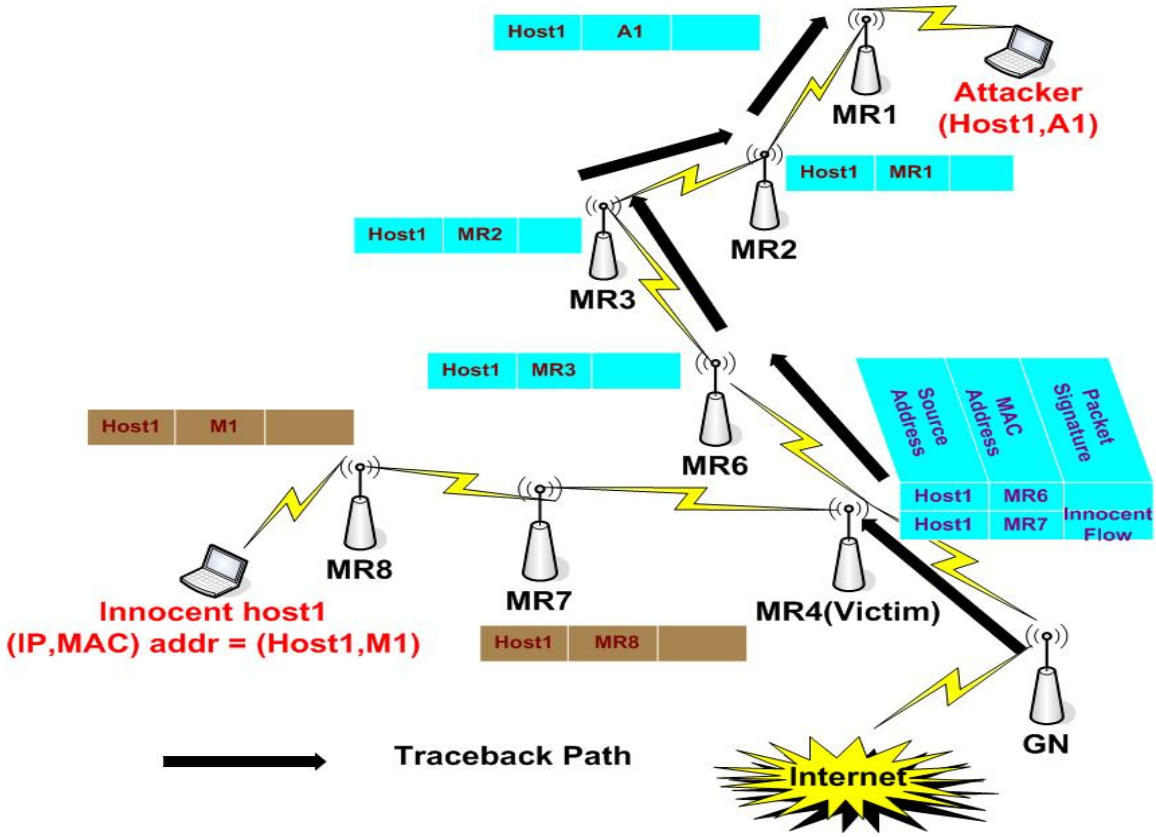


Figure 3.5: MAC Based Traceback Scheme

If a clash occurs during the process of lookup, i.e., there are two MAC addresses corresponding to a given source IP address, the traceback is halted, and additional recourse is required to determine which path to follow. Recall from previous section, whenever a MR detects a clash, it starts storing extra information from the packets. This information is now used to break the tie and is used to identify the exact path of the attacking source with a very high probability. As we see, our reactive approach of storing extra information not only provides a robust way of reducing any false positives but also greatly reduces the overhead of the traceback process.

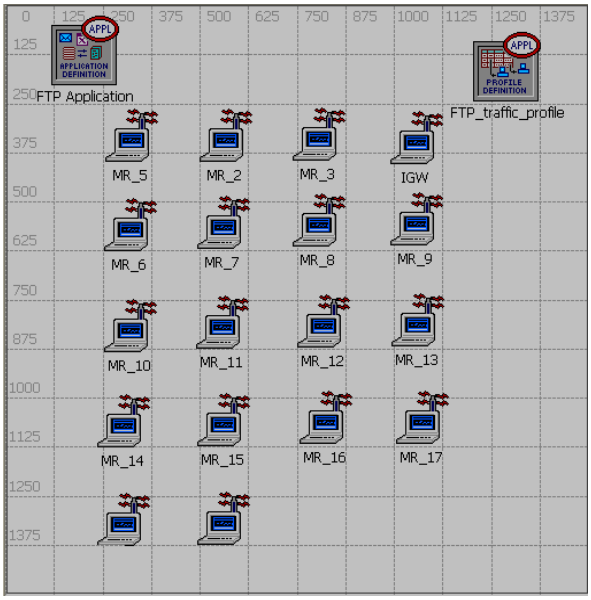
To summarize, in our traceback mechanism, there are two possibilities during the process of traceback:

- If there are no intermediate MRs with clashes, the traceback can proceed in a straight forward fashion, leading to the attacker.
- However, if it results in a clash, it compares the attack packet signature with the stored packet signature corresponding to the two clashing MAC address entries. There are possible cases arising, depending on which entry has been pro-actively recorded first:
- If the attack packet had led to the clash at the time of recording, its packet signature field would be non-empty, which when compared with the trace packet's signature would match exactly and the next upstream router to be queried can be successfully identified.
- If the packet from innocent flow leads to clash, its stored packet signature would not match that of the trace packet. In this case, the other entry with a null value in its packet signature is considered to be the attack path and the next upstream MR given by this MAC address is queried.

We defer an important discussion to the result section, about the possibility of two clashing MAC entries representing multiple attack paths. If the second upstream router had handled the packet, it leads to the third and the querying proceeds backward, until it reaches the true source. This way, the attack path is reconstructed hop by hop; retrieving the corresponding upstream router's MAC address from which it would have been forwarded originally.

3.4 Performance Analysis

We use OPNET's MANET and wireless LAN process model to simulate our proposed scheme and perform a sample attack simulation. We conducted the simulation in a 1400 m x 1400 m area of a campus network conditions as shown in Figure 3.6 and use the wireless traffic setting as shown in Figure 3.7 to evaluate the performance of our scheme. We consider a 4 x 4



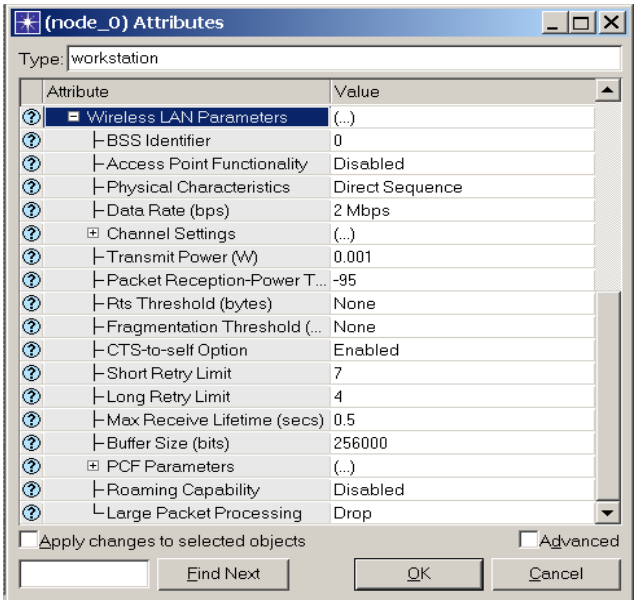


Figure 3.6: Simulation Scenario **Figure 3.7: Wireless LAN Attributes**

grid arrangement of WMN routers placed at a gap of 250 m. The transmission and interference radius of each wireless MR is taken as 250 m and 550 m respectively. We used AODV as the routing protocol though our scheme is generic to be applicable for other routing protocols like DSR. We implement the trace table in the routing layer. In order to make the incoming MAC address available in the routing layer, we modify the packet to include this information before sending upward from the MAC layer. Thus, each MR pro-actively records a two tuple entry in its trace table. When the traceback is initiated, a trace query is dispatched from victim MR to its upstream MR as a separate packet and routed as per the AODV routing protocol. This MR in turn performs a tracetable lookup to determine the next upstream MR which had earlier sent the attack packet and creates a new trace query destined to this MR. Thus, the trace query is propagated towards the attacking MR by consulting the trace table at each upstream router.

In our simulation, the attacking MR uses heavy UDP traffic generating packets of 1500 bytes at 0.001 inter-arrival times to overwhelm victim MR. We can also use FTP traffic for creating a DoS attack for the ease of simulation, rather than a flooding style attack by SMURF, or SYN

packets. As a DoS attack occurs by overwhelming the target network with spoofed packets, we modify the attack source to generate randomly spoofed source address. We study the performance of our scheme in conducting trace back by reconstructing the attack path from victim to the attackers.

The simulations have been run for 150 secs time period and the traffic profile has been activated within the first 5 secs of the start of simulation. As mentioned earlier, an IDS agent is employed to raise an alert and a certain amount of time is allocated for detection of the traceback. The actual traceback process is triggered towards the end of the simulation (i.e., when simulation time is 130 secs).

Figure 3.8 shows the success rate of our proposed traceback scheme in reconstructing the attack path for various percentages of attacker MRs in the network. The victim and the attack sources are chosen randomly to study performance of the traceback. As the number of attackers increases in the network, we observe that the probability of false positives in the traceback process also increases. As we use the signature of the attack packet to resolve the clashing entries in the trace table, the signature comprising of invariant fields of the attack packet, has to be representative of the attack packet.

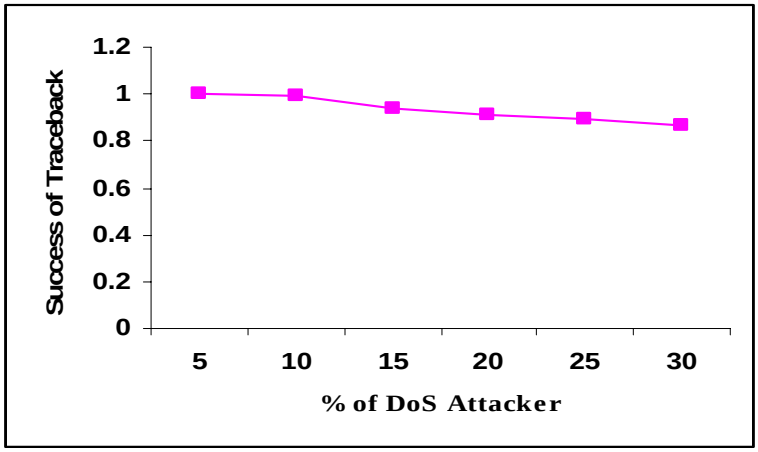


Figure 3.8: Success of Traceback vs. % of Attackers

Hence, there is a possibility of false positives. We do not perform traceback for every single attack packet and instead, assume a flooding style attack and build an attack signature upon examining the flood of attack packets at victim MR.

3.4.1 Analysis of Various Attack

We next examine the efficiency of our scheme under varying attack condition:

Spoofed IP Source Address: A simple attacker might just forge the source IP address to go undetected. Such an attacker is easily detected by our proposed scheme by examining the trace table entries.

Spoofed Source MAC Address: A more intelligent attacker might conceal their identity by forging both the source IP and MAC address. However, even such an attacker can be detected from the trace table information as it is nearly impossible for the attacker to forge the MAC identifiers of the intermediate forwarding MRs.

Single Attacker, Single Path vs. Multiple Paths: The proposed scheme targets to identify the exact location of the attacker, i.e., the MR under which the attacker is serviced. It precisely identifies the attacker and the exact attack path, if there was only a single path taken by the attacker. However, if an attacker sends its packet along multiple paths (that are overlapping at certain routers as shown in Figure 3.9); the proposed scheme can enlist accurately all paths, except one attack path. This is so because, at each intermediate router when a packet with new source IP address arrives for the first time, it is recorded without any packet signature.

The signature of the traversing packet is recorded only when there is already a MAC address present in the trace table for a given source IP address. If the first flow had stopped when a clash occurs, it is not possible to begin recording the signature of this flow also. In such a case, during the traceback process, we would be able to track all except the first old path which was taken by

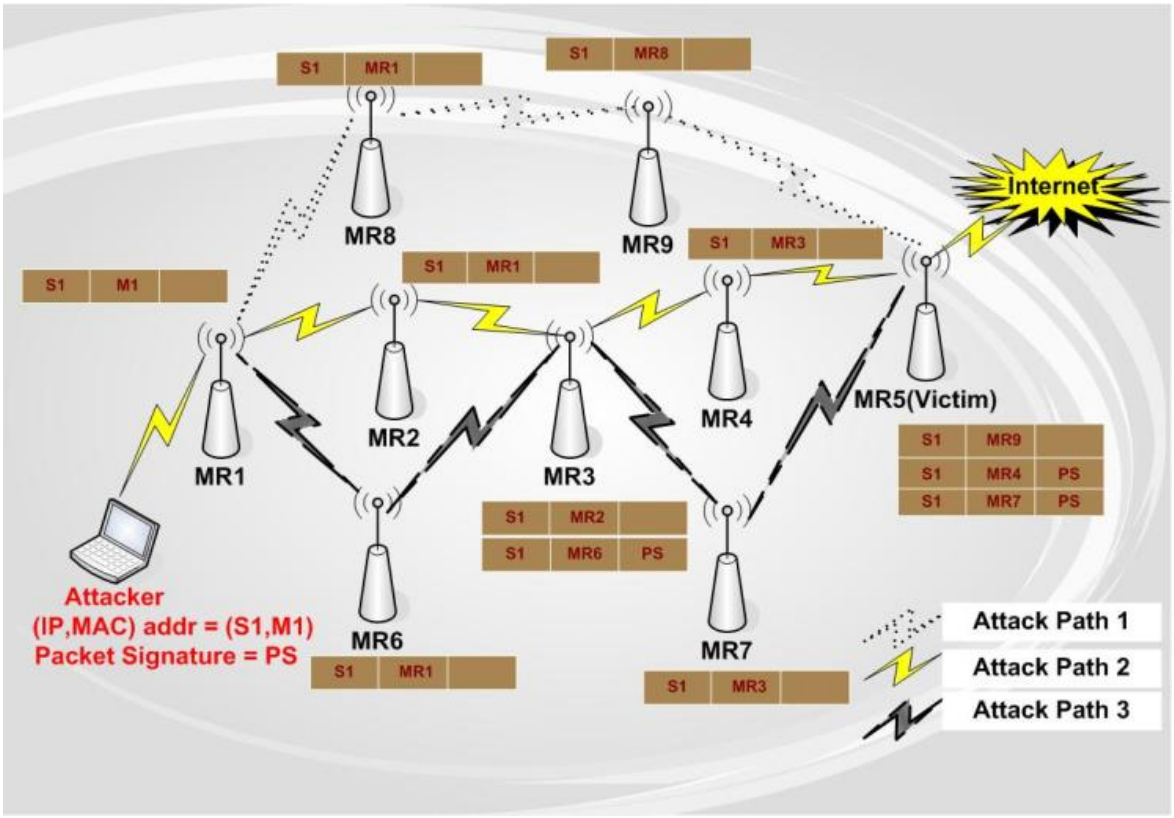


Figure 3.9: Traceback of Multiple Attack Paths

the attacker's packet. This is illustrated in Figure 3.9 in which all the attack paths except the first attack path, i.e., attack path 1 can be traced successfully. This is a reasonable trade-off as we save considerable overhead in recording the packet signature every time.

Single Attacker using different spoofed source IP and MAC address: Our traceback approach can successfully trace the location of the attacker, even if the attacker uses different spoofed source IP and MAC address pairs. In such a case, each intermediate router would record every new pair of source IP and MAC address and all these entries would converge to a single point which is the attacker and hence the identity of the attacker can be traced successfully.

Handling of Distributed DoS Attack: Distributed DoS attack is a more devious attack that falls under the category of reflector attacks. The attacker sends forged request like ICMP Echo Request with spoofed IP address to a large number of innocent hosts that will reply to such

request. The innocent host unicast their reply to the source address given in the received packet, and could unknowingly flood an innocent victim. As such, an attack involves two separate attack paths, one leading to the innocent host and another leading to the actual attacker; they are more difficult to handle. Though, our scheme would require less amount of state information to be maintained at each intermediate router, it would be unsuccessful in identifying the true attacker masqueraded behind the reflector nodes. A significant amount of future work is needed in handling such large scale distributed DoS attacks.

3.4.2 Analysis of Memory Overhead

We further analyze the memory overhead incurred at the routers using our traceback approach. Commercial interest in WMNs is steadily increasing and many companies are coming with proprietary mesh router (EnRoute400, Soekris net4826, Firetide Hotpoint, Tropos 3210) with a limited on-board memory of 64 MB to 128 MB of RAM. Hence, only light-weight applications can be deployed on the mesh routers and our scheme would require only a small fraction of the memory size, about 1 MB (1 %). An entry in the trace table requires a minimum of 10 bytes for the source IP address and previous hop MAC address (32+48 bits). In addition, when a clash occurs an extra 12 bytes is required for storing the packet signature comprising of invariant fields of the attack flow. If p is the probability of a clash and n is the number of flows in the network, our scheme would incur, a constant memory of $n * (10 + 12p)$ bytes at every router. If l, is the length of the attack path, the total memory required along the attack path would be $l * n * (10 + 12p)$ bytes.

In comparison, Baba et al. scheme [38] would incur enormous amount of memory that increases linearly with time and number of flows. An entry in buffer requires approximately 19 bytes (6 bytes for MAC address and 13 bytes for the packet signature). As an entry is created in

router's buffer for every incoming packet, a high bandwidth flow with an inter-arrival time of 0.001 secs (1000 packets per sec) would result in the creation of 1000 entries per second equivalent to 19 KB per sec. And if there are *n* flows in the network, it would incur 18n Kilo Bytes of memory/sec/router. Hence, a flooding style DoS attack would soon result in the exhaustion of the precious router memory (100 flows for 100 secs would incur 190 Mega Bytes at each router).

Thus, our traceback scheme is successful in identifying the source of the attack most of the times, though it might not necessarily identify the current location of the attacker as he/she might have moved to a different router after conducting the attack. Though, the memory of a mesh router is likely to be upgraded with the steady decrease in the cost of memory, it's a good practice to utilize the memory judiciously.

3.5 Conclusion

The open and self-configurable architecture of WMNs makes them vulnerable to DoS attacks. The devious nature of DoS attack camouflages the true source of the attacker due to the use of spoofed source address. Hence, we propose a traceback scheme resistant to source address spoofing. Our traceback scheme based on MAC address identifier has the following advantages:

1. It requires minimal amount of memory at each of the forwarding routers unlike other logging schemes and is hence feasible to implement on the mesh routers.
2. The overhead incurred in tracing the attacker is less even in the presence of spoofed source addresses.

In the future, we intend to test the performance of our scheme in a prototype WMN test bed and analyze the memory overhead and latency of our scheme in real-time. We also plan to extend our traceback scheme to a mesh topology that supports multi-hop mesh clients serviced under a single mesh router.

Chapter

4. Distributed Self Policing Architecture for Detecting Selfishness

4.1 Introduction

One of the salient characteristics of WMNs is self-configurability as in Mobile Ad hoc Networks (MANETs), by which all participating nodes automatically discover routes to the each other and cooperatively forward each other's packets towards the IGW. Even though MRs are not mobile, it is critical to examine the level of cooperation guaranteed in different operational structures of WMN: fully managed, semi-managed and unmanaged.

In semi-managed and unmanaged WMNs, MRs may not belong to a single trusted authority, and each MR can freely decide on how they use their resources. Like traditional wireless networks, a WMN is also characterized by bandwidth-constrained and variable-capacity links which motivates nodes to act selfishly. Also, if the MRs are under the control of different authorities, they are encouraged to misuse the resources for themselves. As the bulk of traffic in

a WMN is oriented towards the IGW, paths near the IGW are heavily congested and a MR can behave selfishly in an attempt to avoid congestion [40]. A selfish MR always favors the traffic originating from its mesh clients; while fully or partly dropping the relayed traffic. It monopolizes the network bandwidth for its own use, dropping any forwarded traffic originating from other MRs. We also call such a MR as a free-rider, as it enjoys the network resources without rendering any forwarding services to the community. In order to simplify our discussion with respect to the selfishness, WMNs are henceforth referred to as semi-managed and unmanaged WMNs.

Selfishness is a common phenomenon rampant in any network that depends on a cooperative framework. WMNs and MANETs exhibit some common characteristics such as limited access to wireless channel, which motivates the nodes to acquire more bandwidth for itself and behave selfishly. Researchers have slowly realized potential impact of this problem in WMNs and in the recent years have started developing collaborative schemes [41]. However, unique properties of WMNs make selfishness yet to be understood fully:

- The WMNs are not power constrained. But, compared to MANETs and traditional wireless networks, they have distinctly different motivations for exhibiting selfishness. For example, a set of MRs connects only to those MRs that are close to the IGW, and is completely dependent on them for forwarding the packets. If the MRs close to the IGW exhibit selfishness, the MRs located further away from the IGW are denied internet services.
- In a MANET, each node is mobile and can change its identity. So, it is a challenging problem to cope with the frequently changing topology and it is relatively hard to maintain the reputation of each node. In a multi-channel WMN, however, it is challenging to develop a reputation-based scheme as it judges the behavior of an adjacent node by overhearing its's

communication. But, it is not always possible to eavesdrop on the near-by communication if adjacent MRs are tuned to operate on distinct orthogonal channels.

In this chapter, we propose a *D*istributed *Polic*ing architectur*e* (D-POLICE) for WMNs, to detect selfish MRs. We employ specialized agents called Sink Agents (SA), which are delegated the task of vigilantly policing the network to detect selfish MRs. These agents are reconfigurable and can be operated in a static or mobile mode. The SA collects special reports known as traffic reports from the MRs. A traffic report is a comprehensive statistical representation of a MR's interaction with its neighbors. In order to expedite the delivery of such reports, we modify the underlying queuing mechanism so that they are enqueued at the head of the MR's interface queue and are always transmitted first. In the static mode, the SAs distributed in the network choose random policing intervals and instructs every MR to submit traffic reports to a near-by static SA. On the other hand, in the mobile mode, the SA retrieves traffic reports by conducting random patrols in the network just like a network cop. The SA applies simple rules on the gathered traffic reports to identify any potential selfish MRs. The element of randomness dictates every MR to operate cooperatively due to the fear of detection. At the same time, it incurs very low overhead as compared to a pro-active periodic monitoring approach.

The remainder of this chapter is organized as follows. We discuss the related work on detecting selfish MRs in Section 4.2, followed by an outline of the network architecture and the adversary model in Section 4.3. We describe the working of the proposed D-POLICE scheme in Section 4.4 and explain in detail various reconfigurable modes of operation of the SA. Section 4.5 discusses the performance evaluation of our scheme. We finally conclude with a summary of our work in Section 4.6.

4.2 Related Work

In this section, we investigate various selfish node schemes in the literature, and then analyze their applicability to WMNs. A more comprehensive study on the problem of selfishness can be seen in our survey work [9]. A selfish node scheme targets to stimulate cooperation between nodes. The selfish node schemes can be generally classified into four categories as shown in Figure 4.1: credit-based, reputation-based, game theory-based schemes, and others.

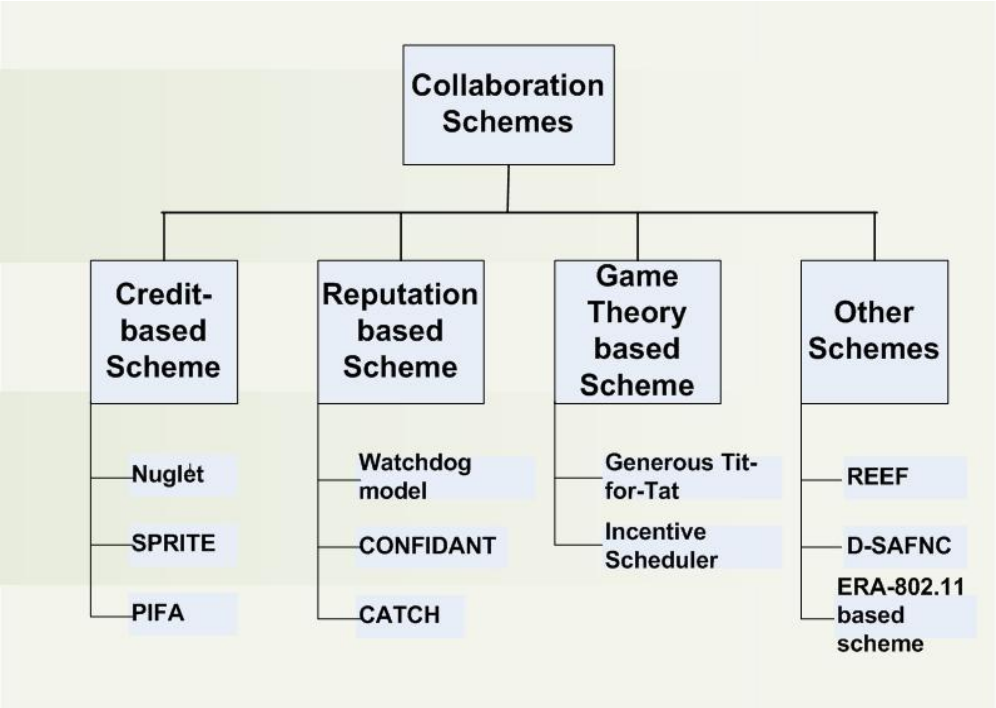


Figure 4.1: Classification of Selfish node Schemes

- Credit-based Scheme:

In a credit-based scheme, nodes providing forwarding services to other nodes are rewarded with virtual currency, and a source node using the forwarding service is charged. A node is constrained by the amount of wealth it possesses and a selfish node is thus forced to behave correctly to avoid being poor. The credit-based schemes primarily differ in their methodology adopted for the allocation of wealth.

Buttyan et al. [41][42] propose two models to reward forwarding nodes, PPM (Packet Purse Model) and PTM (Packet Trade Model). In PPM, a source node loads *nuglets* (virtual currency) into packets, which are later deducted at every hop by the intermediate nodes. It is however restricted to source routing protocol, as sufficient amount of *nuglets* need to be loaded by the source. In PTM, each intermediate node buys packets from its previous hop node and sells it at a higher price to its next hop. It is prone to Denial-of-Service (DoS) attack by a malicious source sending flood of packets. *Nuglets*, however, require tamper-resistant hardware to maintain the authenticity of currency. SPRITE [43] ensures the authenticity of currency by employing a centralized authority called CCS (Credit Clearance Service). Every node keeps a receipt of the packet it receives and submits it to CCS. The CCS then determines the charge and credit for every node in the transmission path from a game-theoretic perspective so that other nodes are motivated to report correctly, even when selfish nodes collude and submit false receipts. SPRITE suffers from scalability problem in a large-scale network, as nodes submit receipts to CCS for every packet. This also incurs considerable communication overhead. Nodes in PIFA scheme [44] avoid this overhead by submitting periodic reports to a centralized credit manager (CM). The CM applies necessary rules to validate the credibility of the reports (by comparing reports of adjacent nodes), and rewards well-behaving nodes. This scheme also suffers from the scalability problem. A centralized agency for auditing and accounting is prone to single point of failure (if the centralized authority is compromised or attacked, the scheme fails).

The static and hierarchical architecture of a WMN makes credit-based schemes largely unfit for it. In a MANET, an isolated node can move to a better location where it is more likely to be selected as an intermediate node and can now earn credits. However, WMNs are relatively static, due to which some MRs in the periphery may never accumulate any wealth; creating large

disparity in the distribution of wealth among the leaf MRs and the central MRs. On accumulating sufficient currency, a wealthy MR can refrain from relaying packets, which is counter-intuitive to our goal. A source node in a WMN might quickly run out of wealth due to the high-data rate of some applications such as multimedia.

- Reputation-based Scheme:

The goal of a reputation-based scheme is to make informed decisions about which nodes cooperate and which behave selfishly. The quality rating (or reputation) of the participant nodes is built based on local observation at the node or based on second-hand observation at other nodes or both.

Marti et al. [45] propose a monitoring agent called *watchdog* at every node that overhears the transmission of its neighboring nodes so as to detect non-forwarding misbehavior. A source node then selects a route with the highest rating. The scheme is prone to a replay attack since a node can pretend to forward by replaying an old cached packet. Buchegger et al. [46] propose CONFIDANT protocol that assigns rating for every node based on the *watchdog* as well as second-rating information gathered from other nodes. The second-rating information prevents spurious rating and detects inconsistencies in the two observations. Accordingly, a path manager selects the best path by avoiding selfish nodes. The process of building reputation in both schemes can be distorted due to collision. They also face the scalability problem due to the centralized validation by the trust manager. CATCH [47] consists of two sub-protocols called Anonymous Challenge Message (ACM) and Anonymous Neighbor Verification (ANV). In order to test the connectivity of a node (*testee*), an ACM message (unknown sender) is unpredictably sent for re-broadcast by all its neighbors (*testers*). As the identity of the sender is unknown, the node is forced to re-broadcast every ACM message. In the ANV phase, a *tester* sends a

cryptographic hash of a random token for re-broadcast and also records other hashes sent by other nodes. If a *tester* considers the *testee* to be normal it releases the secret token to it. Thus, testers exchange each other's opinion by the release of their secret tokens.

The applicability of a reputation-based scheme to a WMN is arguable. The strength of these techniques lies in the assumption that a node can overhear other's transmission which is not possible in a multi-channel WMN. In such a multi-channel WMN, adjacent nodes are assigned non-overlapping orthogonal channels. Hence, promiscuous listening is not always possible. A reputation-based scheme requires a reliable way of distributing reputation to prevent false accusations and spurious ratings. It is also difficult to build a good reputation within a short period of time.

- Game Theory-based Scheme:

The goal of game theory-based scheme is to derive an optimal strategy for every rational player. The operation of the network is modeled as a *forwarding game*, where each node is a player. A game theory-based scheme models the packet forwarding functionality of a node as a strategic game. It derives a *Nash Equilibrium* point for the forwarding rate of a node such that any deviation from the equilibrium results in lesser pay-off for the participant node. A malicious participant cannot increase its pay-off by adopting a different strategy, while the strategy of other participants remains the same.

GTFT (Generous TIT-FOR-TAT) [48] uses the game theory to encourage collaboration in MANETs. GTFT equalizes the percentage of request served by a node *j* for others nodes (*Help-given)* against the percentage of request served by others for the node *j (Help-received).* The node then applies the following check to determine whether or not to relay packet: *Help-received* $+ \Delta >$ *Help- given*. (where Δ is a small positive number to prevent initial system deadlock). If

the above equation is true, a relayed packet is accepted and forwarded by this node. GTFT, however, requires the knowledge about the utility of all nodes in the network. Wei et al. [49] present an incentive scheduler that ensures a fair channel allocation for users in a WWAN/WLAN two-hop-relay system using game theory. A user is assigned time slots by an AP (or a base station) for channel access depending on its instantaneous channel rate, average allocated rate, and the incentive variable (set to a constant value for a relay node and non-zero for others). A node serving as a relay is given a higher priority by the AP during channel allocation. Thus, the relay node obtains a high throughput from the AP.

Recently many game theory-based schemes have been proposed. However, most of the assumptions of game theory-based schemes are not applicable in WMNs. A game theory-based scheme like GTFT assumes all nodes are subjected to energy constraints (i.e., limited node lifetime), which is not true for power-rich MRs in WMNs. A game theory-based scheme like Wei et al. [49], assumes that the AP controls the radio resources for all the network entities in a WWAN/WLAN two-hop-relay system, which is not the case in WMN systems (managed individually by MRs)

- Other Schemes:

Conti et al. [40] propose REliable and Efficient Forwarding (REEF) scheme, in which a node exploits its local knowledge to estimate the reliability of a path. Unlike conventional method of denying services to selfish users, it provides a differentiated quality of service by forwarding their traffic slowly. The node depends on TCP ACKs to update its neighbor's reliability. It is challenging to add an ACK phase to the traditional best of effort UDP model. Cardenas et al. [50] present a solution to prevent MAC layer selfishness. They present a new protocol called ERA-802.11, in which the sender and the receiver commit to a random back off time interval

through a secure message exchange. It ensures an honest and a random back off, only when one of the communicating entities (sender/receiver) is honest. Thus, the scheme can be prudently adapted for detecting new types of selfishness in a multi-channel WMN like radio selfishness, channel riding, etc.

4.3 Network Architecture and Selfishness Adversary Model

In this section, we present the architecture of WMN and then outline an adversary model for identifying different possible kinds of selfish behaviors in WMNs.

4.3.1 Network Architecture

A semi-managed or unmanaged WMN employs a hierarchical architecture [51] consisting of three layers as illustrated in Figure 4.2.

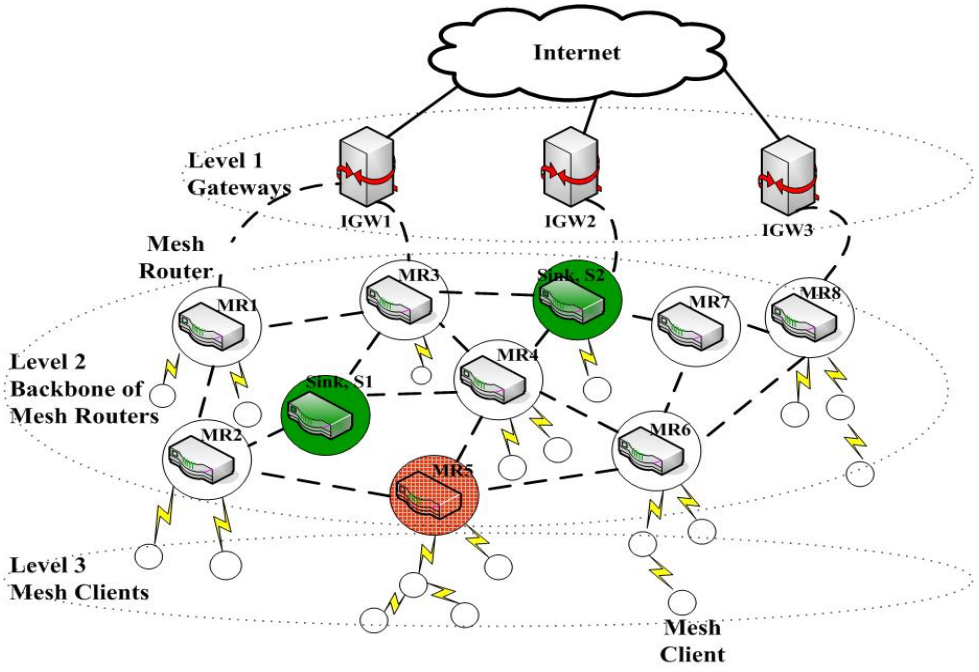


Figure 4.2: Monitoring using Sink Agents in a WMN

On the lowest level of the hierarchy are the mesh clients that connect to its nearest MR in a single hop or multi-hop fashion. The second level of hierarchy consists of a set of n MRs, $\boldsymbol{N}$ =

{MR_1, MR_2, …, MR_N}, forming the wireless backbone. The MR aggregates data traffic from the mesh clients and then forwards it towards to the destination. The destination can either be another mesh client that is attached to another MR, or be a host on the Internet. Each MR is equipped with *m* radios, i.e., interfaces, (R = R_1, R_2… R_m), in order to simultaneously transmit and/or receive packets. Every MR communicates with its neighbors on non-interfering radio spectra, i.e., *l* non-interfering channels (C_1, C_2…C_l such that $1 \leq m < l$). Given a specific channel, $k \in C$, the maximum transmission rate of a link w_k bps.

Let T_R^i be the total transmission rate of $MR_i \in N$ that can be achieved by using all *m* radios, i.e., $T_R^i \leq \sum_{k=1}^{m} w_k$ bps. An un-monitored $MR_i \in N$ can freely decide on how it assigns bandwidth on its *m* radios, to send its own traffic and relay traffic of other MR nodes. Let (x_n, y_n) donate the allocation at MR_n, where x_n is the rate at which MR transmits its own traffic, y_n is the rate allocated for relaying traffic for other MR nodes. Let ($x_n^{'}, y_n^{'}$) donate bandwidth demand at MR_n where $x_n^{'}$ is the demand data rate for its own traffic, $y_n^{'}$ is the demand data rate for relaying other MR's traffic. For every well behaving MR, $0 < x_n^{'} < x_n$ and $0 < y_n^{'} < y_n$ is strictly true.

At the top of the hierarchy, there are *g* IGWs that provide the Internet connectivity for the MRs. We assume that all IGWs are critical infrastructure, can not be operated by an individual and are thus trustable. Every MR is connected to atleast one of the *g* IGWs for having Internet

access. The maximal throughput of an IGW is $\sum_{k=1}^{l} w_k$ bps, when an IGW configured with *l* interfaces employs all non-overlapping channels. Therefore, the maximal Internet throughput per MR is $T_R^i(Internet) \leq \sum_{k=1}^{l} w_k \times g / n$ bps, when all *g* IGWs are configured with the maximum number of interfaces *l*. We can see that the throughput T_R^i or $T_R^i(Internet)$ of every MR is limited by the number of channels and the interfaces.

4.3.2 Selfish Adversary Model

In this subsection, we present the selfish adversary model that illustrates the motivation for selfishness in the semi-managed and unmanaged WMNs. As illustrated in the network architecture, every MR has limited throughput of T_R^i and $T_R^i(Internet)$, which motivates a MR to be selfish. In an unmanaged WMN, each MR is an independent entity that can act selfishly to choose a bandwidth allocation such that the bandwidth allocated for its own aggregate traffic originating from its mesh clients is maximized. In a semi-managed WMN, while the MRs forming the core of the wireless network are under the control of an ISP, other MRs operate independently (as in an unmanaged WMN).

- Dropping packets: A MR indiscriminately drops all or some of the relay traffic depending on its available bandwidth: $B_R^i = x_n - y_n$. Whenever MR's residual bandwidth falls below a threshold, it stops forwarding the relay traffic to conserve the bandwidth for itself. In order to hide its selfish behavior, MR can also sometime drop relay packets, while dutifully forwarding at other times. In a similar manner, a MR may intentionally drop the Internet packets from a

neighboring MR to increase its own Internet bandwidth $B_R^i(Internet)$. Such a selfish MR is considered to be malicious as it tries to camouflage its selfish behavior from the outside world.

- Radio Selfishness: A selfish MR can also falsely claim its radios to be busy so that it is not requested for forwarding services by its neighbors. Alternatively, it can also configure the transmit interface to send its own packets only. It greedily devours the channel for itself, rather than symbiotically "using" and "providing" network service.
- Channel Riding: A selfish MR can also hold a channel, thus decreasing the channel utilization. If a fixed channel assignment is not used and a dynamic channel negotiation [51] is employed at the MRs, a selfish MR can indefinitely hold on to a channel for its usage, blocking all transmissions in its neighborhood.
- Misreporting: A selfish MR when submitting traffic reports to the SA can falsely claim to have forwarded the packets, while discretely dropping the packets. This might cast a suspicion on the neighboring MRs during the validation of reports. The MR, thus, is said to exhibit selfish as well as malicious behavior.

The adversarial impacts due to selfishness in a MR degrade the network performance in a different way when compared to MANETs [9]. WMNs are envisioned for high bandwidth broadband applications like voice and video services which mainly use UDP. Due to the absence of end-to-end acknowledgement, the genuine traffic sources remain unaware of the packet drops at an intermediate selfish MR. In a hierarchically structured WMN, a selfish MR near the IGW results in a denial of service to flows originating from other distant MRs. The proximity of a selfish MR to the IGW plays a critical role in the level of deleterious impact on the network. In a sparse network, a selfish MR in an intermediate path could also result in a network partition. Even if a selfish MR is detected, the affected MRs would have to take remedial action to route

their traffic through other alternate paths which could be circuitous. This may result in increased congestion at the neighboring MRs and result in traffic delays that are intolerable for real-time applications.

4.4 Problem Analysis and Design Motivation

We consider an efficient channel assignment strategy for the WMN that would elegantly assign non-interfering channels to neighboring MRs. Here, we focus on the detection of selfish MRs in a multi-channel environment, rather than a distribution scheme for the channels. The self-organizing and collaborative nature of WMN architecture necessitates tight supervision of the network. Detecting and discouraging selfishness in MANETs mainly adopts either credit-based, or reputation-based, or game theory based approaches. However, they cannot be directly applied for WMNs due to several architectural differences [52][53]:

- As WMNs employ multi-channels for simultaneous transmission and reception, reputation based schemes that use promiscuous listening mode cannot be applied here.
- Unlike MANETs, the core of WMN comprises of wireless MRs that are relatively static. The MRs near the IGW are more frequently involved in packet forwarding than the leaf MRs which creates disparity of wealth.

In view of these differences, we feel that a pervasive monitoring is required for WMNs. In this chapter, we describe a distributed scheme called D-POLICE to detect selfish MRs in a WMN. The underlying detection mechanism is based on active monitoring of MR's traffic. Like other traffic monitoring approaches, the basic questions that arises in conducting traffic monitoring are: (i) how many relay packets have been received by a MR, and (ii) how many of these relay packets have been forwarded by the same MR. More specifically, we need to check if the number of relay packets received from other MRs is equal to the number of packets that the

MR forwarded (if we ignore the packet losses due to interference or other factors). Otherwise, the MR acts selfishly. This procedure of checking the inconsistency between the received packets and forwarded packets in MRs is known as an *inconsistency check*. Though this check sounds simple, the malicious nature of selfish MRs complicates the process as it can falsify the source address of packets originating from itself and make it appear as relay traffic. Thus, to truly evaluate the trustworthiness of a MR, we need several vital traffic reports such as the number of packets received from others MRs, the number of packets terminating at this MR, the number of packets originating from this MR, the number of packets sent from this MR, etc. Finally, we also need to cross-verify each of this statistics based on the reports from a given MR and its neighbor. Thus, we propose to collect the above listed statistics between every pair of adjacent MRs.

However, in a distributed network, considerable amount of computational and communication overhead is incurred in performing continuous evaluation of the traffic reports gathered from every MR. This overhead becomes more and more significant as the network size grows. In order to prevent any degradation caused by the traffic analysis, we design a traffic monitoring system that reduces monitoring overhead by investigating the network status at random intervals. Thus, the majority of the network bandwidth is preserved for user traffic rather than just the traffic reports. The proposed D-POLICE scheme does not collect the traffic report at all times. On the contrary, like a police in the highway, the IGW or a trustable agent instructs a set of MR to submit their reports for a random interval of time. Though the MRs pro-actively gathers traffic statistics, they submit reports to the SA only when instructed to do so and thus save valuable network bandwidth. As MR does not know in advance the exact time instance that it will be checked, the traffic monitoring deters selfishness of MRs. Upon retrieving the traffic

reports on-demand, the SA evaluates the reports and if MR is identified to be selfish, the MR will be appropriately punished or even quarantined from the network. When a selfish MR is quarantined from the network, it is no longer tendered any service from other MRs.

In our D-POLICE design, we propose to employ a functional entity called as a Sink Agent (SA) to actively monitor the network. The basic functionalities of a SA is (i) to collect the traffic reports from the MRs, which represents a summary of a MR's transaction for a specific period of time, and (ii) to analyze the traffic report to identify the selfish MR. For performing the first functionality, we design centralized and distributed D-POLICE traffic report schemes. A traffic report of a MR for a given time period represents how many packets are received by it from its neighbors and how many of these relay packets are forwarded. The traffic report also summarizes the source MR, destination MR and the interface of a MR for every traffic flow. For the purpose of the second functionality, we design the D-POLICE traffic analysis scheme. The SA thereafter examines the traffic reports to check if there are any anomalous activities like packet dropping, channel riding, or radio selfishness. It detects inconsistency in the traffic report using simple checkpoints and employs a selfish MR detection algorithm to identify the misbehaving MR. For simplicity, we use the symbol MR_i and MRi interchangeably for representing a $MR_i \in N$. The following are the details of the D-POLICE design.

4.5 Centralized and Distributed Traffic Report Scheme

The SA is a reconfigurable software entity that is deployed on the IGW in an unmanaged WMN (as only the IGW is trustworthy). In the semi-managed WMN, the SA can be deployed on reliable core MRs that are trusted by the IGW. Unlike MANET, the WMN has a static topology and every MR is usually situated on building rooftops. Therefore, we design different traffic reporting strategies to validate the behaviors of the MR in the D-POLICE scheme. Figure 4.3

presents the broad classification of D-POLICE scheme based on the operational style. As shown in Figure 4.3, the SA can be deployed in a centralized mode or a distributed mode.

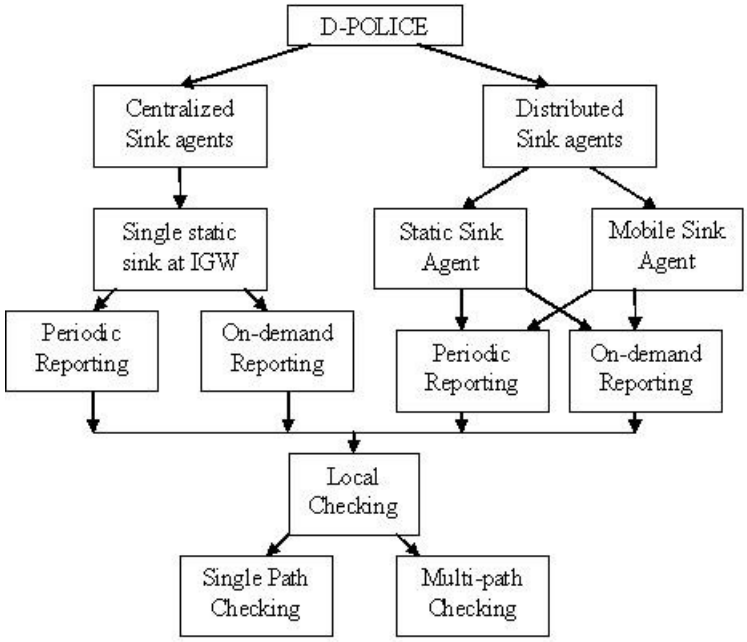


Figure 4.3: Classification of D-POLICE Scheme

In the centralized mode, only the IGW is authorized for managing the SA. On the contrary, in the distributed mode, a WMN has several distributed SAs residing either in the IGW or trustworthy MRs. The SAs can be configured to collect the traffic report from MRs in two modes. The first is the static mode of SA, in which the static SAs residing at the IGW or core MRs request the MRs in its region to submit their traffic report. Alternatively, the SA can also be configured to operate as a mobile SA that is commissioned by the IGW/core MRs to traverse a given region and gather traffic reports from a set of MRs. The mobile mode of operation offers greater flexibility in attack detection and largely mitigates the monitoring overhead. Another important advantage of a mobile SA is that, the IGW can reactively check the traffic flows for a small geographic region (e.g., a path). Furthermore, the traffic report can be gathered in a

periodic or on-demand fashion as shown in Figure 4.3. We further describe these operational modes as follows.

4.5.1 Centralized Mode

The D-POLICE scheme can be configured to operate in a centralized mode with a single static SA at the IGW. This approach can be mainly adopted in an unmanaged network where the mesh infrastructure lacks reliable deployment points and the only trustable node is the IGW. However, a centralized infrastructure incurs considerable routing overhead [44] and increases the level of congestion at the IGW, which is already considered to be a traffic hotspot. Figure 4.4 shows the operation of D-POLICE at the IGW. The static SA deployed at IGW can pro-actively monitor the network by retrieving traffic reports from all MRs in the network, i.e., MR1, MR2, MR3, MR4, MR5, and MR6 at regular time intervals. As traffic reports from all MRs are routed towards the IGW, it increases the level of congestion in the paths towards the IGW, which is undesirable. Alternatively, the SA can request MRs to submit reports at random time intervals.

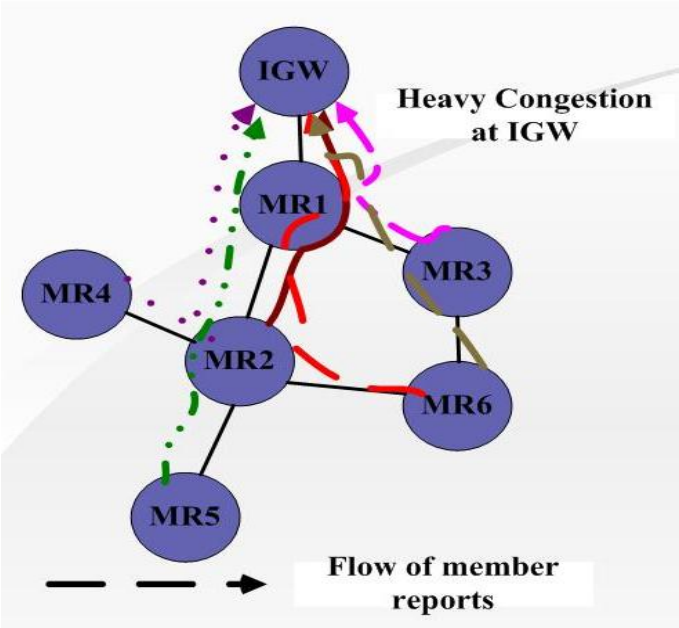


Figure 4.4: Centralized Mode of D-POLICE

4.5.2 Distributed Mode

A distributed approach promotes the idea of dispersing the SAs in the network. The SA assumes the role of a policing agent and monitors the traffic in its region. This reduces the overhead in discovering routes towards the SA. The distributed nature also reduces the detection time and facilitates quick remedial action. The distributed approach can be further classified into two types based on the mobility of SA as: Static SA and Mobile SA.

- **Distributed Static Sink Agent**

A static SA can be deployed at reliable sink nodes operated by the ISP such that all MRs submit traffic report to its nearest SA. For example, in addition to the IGWs, Figure 4.2 has two SAs distributed in the MR: Sink S1 and Sink S2. In the beginning, SAs (e.g., S1 and S2 as shown in Figure 4.2 advertise their presence to the MRs (e.g., MR1, MR2, MR3, MR4) by broadcasting a beacon. Each MR, upon receiving a beacon registers itself under a SA (e.g., S1 or S2) in the following two cases:

- If it has not registered with any SA, or
- If a new SA is found to be nearer than its previously registered SA.

The registration divides the WMN into multiple local monitoring domains and each MR is registered to its nearest SA. To avoid flooding of the beacon, each MR rebroadcasts the beacon only if the hop-count is less than a given number. Once MRs are aware of their respective SA, the SA obtains traffic reports from the MRs either periodically or on-demand. It then applies traffic inconsistency check to evaluate the truthfulness of the traffic reports.

The static SA design is analogous to a client/server structure where the SA acts as the server (i.e., traffic manager) and each MR managed by the SA is a client that submits its traffic reports to the SA (periodically/on-demand). The client/server structure suffers from inefficient

utilization of the network bandwidth. The SA issues the traffic query to the managed MR for the purpose of retrieving traffic information. Thereafter, the registered MR responds to the query by submitting their reports. However, communication overhead increases as large amount of traffic reports flows towards the SA. Thus, a static design is inappropriate and not scalable to a large network. Consequently, to prevent any performance bottleneck at the SA, we are motivated to design a mobile SA as follows.

- **Distributed Mobile Sink Agent**

A mobile SA, like a police patrol moves around the network to collect traffic reports and performs inconsistency check. The mobile SA is commissioned by the IGW or a trustable core MR, to visit a set of MRs and collect local traffic reports from them. The mobile SA migrates from one MR to another MR hop-by-hop according to a predefined itinerary. The mobile SA acts like a network police, conducting inspections from time to time. The mobile SA is an active software agent that works in conjunction with the routing module of the MR to intercept the network traffic. After collecting the required traffic reports, the mobile SA evaluates the traffic reports to check the presence of any selfish MR. If found, the SA then informs it to the IGW. This process avoids unnecessary flow of traffic reports to the IGW, thereby saving considerable communication overhead. The decentralized mobile SA architecture overcomes the limitation faced by the static SA client/server architecture in terms of bandwidth consumption and performs vigilant surveillance of the network.

The mobile SA can be made to move along pre-configured itinerary in the network, which gives a great flexibility in examining specific network segments of interest. Alternatively, the mobile SA can also conduct random walk of the network, starting from the IGW. We assume that the MRs in the network are pre-configured with a label that represents the depth of the node

w.r.t. to the IGW (root of the tree). Hence, every time the mobile SA moves along the outgoing edge of a node, it randomly chooses a MR that lies in the next lower level. This ensures loop free movement and also conducts depth first probing. It probes the network deeper and deeper until it reaches the leaf MR. The mobile SA thus moves along multiple paths in the network and performs an exhaustive systematic search. We assume that the mobile SA keeps track of the return path to the IGW during its tour, so that it eventually returns to the IGW at the end of its tour. In order to reduce the latency in gathering reports from the MRs and to reduce the attack detection time, we employ multiple mobile SA.

It is evident that the use of mobile SA requires some security framework to safeguard its operation against malicious selfish MRs. We incorporate a simple security framework for encrypting the traffic payload inside the mobile SA. We employ a public key infrastructure (E_k, D_k) at the IGW, where E_k is the public key available to all MRs in the network, and D_k is the private key available only to the IGW [54]. Thus, $MR_i \in N$ encrypts the traffic reports M_i using E_k and also includes a random nonce N_0 and its ID as: $< MR_i, N_0, E_k(M_i, N_0) >$. The MR submits the encrypted message to the mobile SA. The IGW or the SA then retrieves the gathered traffic reports by decrypting it, using the private key and checks for foul play such as replay attack [54]. Figure 4.5 shows the operation of a mobile SA, that visits a network segment in the following order: MR1, MR2, and MR5. At the end of its tour, it summarizes the traffic reports in a tabular form. The IGW then decrypts the gathered traffic reports by using its private key and checks for replay attack, selfishness, etc. [55].

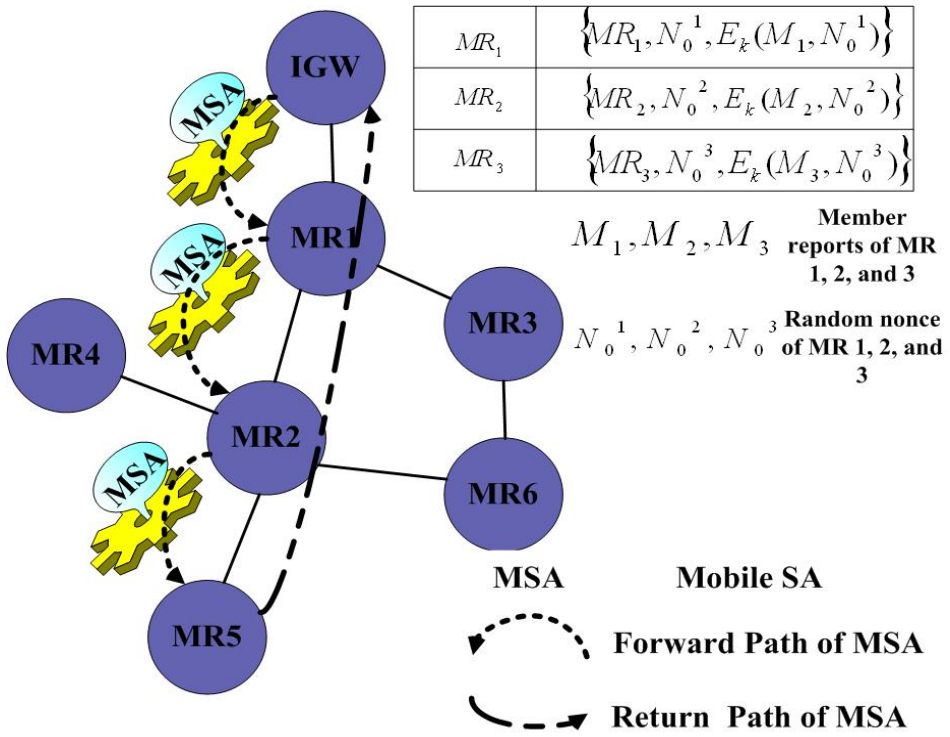


Figure 4.5: Decentralized Mode of D-POLICE

4.5.3 Periodic and On-demand Reporting

In centralized or distributed mode, the traffic reports can be submitted in a periodic or on-demand manner. The MRs pro-actively maintains a historic summary of its traffic transactions at a given time interval. In the periodic reporting, the SA obtains the traffic reports from MRs at fixed time intervals. In static SA mode with periodic reporting, all MRs periodically submit traffic reports to its registered SA every τ secs. In mobile SA mode with periodic reporting, the mobile SA travels the network every τ secs to collect the traffic reports. An adversary can camouflage its operation by behaving correctly during the reporting intervals, while failing to forward at other times. In the view of this behavior, an on-demand reporting is more vigilant in detecting the selfishness of MRs. In this approach, the SA demands the MRs to submit its traffic reports for a specific time interval τ' chosen randomly. The static SA mode with on-demand reporting emulates a police on the highway, who stations himself/herself at a strategic location at

random time periods. On the contrary, the mobile SA mode with on-demand reporting is analogous to a police patrolling the neighborhood and performing random checks. Thus, in a similar fashion, the mobile/static SAs examine the status of network operation at random intervals of time. As MRs are unaware of the policing interval, they are forced to consistently cooperate with other MRs in forwarding each others packets. Therefore, the traffic monitoring is performed at a reduced overhead.

4.5.4 Local PathChecking

In centralized or distributed mode, the SA investigates the status of different network regions. The mobile SA, for example, can be deployed to collect traffic reports from single or multiple paths periodically or on-demand.

- **Single Path Checking:**

Figure 4.6 shows the operation of a mobile SA, that collects reports from a given path (IGW → MR1 → MR2 → MR5). The mobile SA collects traffic reports from MR5 and MR2. It then evaluates it based on simple checks such as checking if the number of packets sent from MR5 to MR2 is same as the number of packets received at MR2. It repeats the above analysis on reports gathered from MR2 and MR1, and so on. It conducts a set of simple test like this at every hop and arrives at a conclusion about the behavior of each MR along the path. In case of a mismatch in the above equation, it is flagged and logged as an inconsistency by the mobile SA, SA1. For example, if MR2 is the selfish node, an inconsistency is guaranteed to be raised in the reports between MR2 and MR1. We defer the discussion of inconsistency checks to the next section.

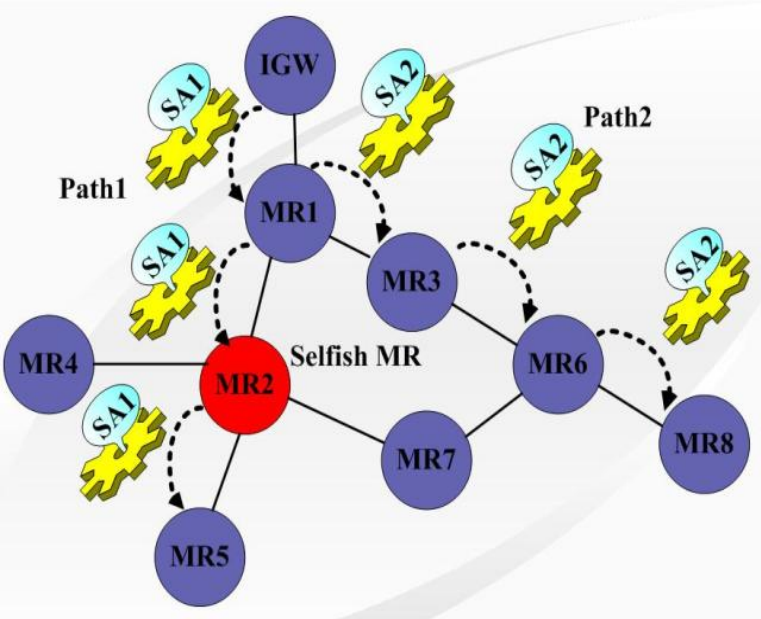


Figure 4.6: Path Checking

- **Multiple Path Checking**

Figure 4.6 shows the functioning of two mobile SAs that retrieves reports from multiple paths simultaneously. We employ multiple agents to retrieve reports, one along the first path and the other along the second path. The mobile SA1 conducts a walk along path1 from IGW → MR1 → MR2 → MR5 and the mobile SA2 conducts walks from IGW → MR3 → MR6 → MR8. The mobile SA retrieves reports and evaluates the MRs along a given path by comparing reports from every adjacent pair of neighbors, i.e., if the number of packets sent from MR2 to MR1 is the same as the number of packets received at MR1 from MR2, and similarly if the number of packets sent from MR3 to MR1 is same as the number of packets received at MR1 from MR3. Thus, localized checking of the network along different paths is guaranteed to identify precise identity of the selfish MRs in the network.

4.6 Traffic Consistency Checking and Selfish MR Detection Algorithm

Before identifying the exact identity of the selfish MR, the traffic inconsistency check analyzes the traffic report from the MRs.

4.6.1 Traffic Inconsistency Check

The MRs submit traffic reports to the SA either periodically or on-demand under the request of the SA. Table 4.1 gives the definition of various fields in the traffic report. Figure 4.7 illustrates the Table 4.1 for a sample traffic scenario. In the table, for a $MR_i \in N$ we use i and MR_i interchangeably. MR_i and MR_j denote two adjacent MRs.

The first four fields (i.e., MR_j, MR_i, SA_j, and Seq_j) represent the MR, its neighbor, the registered SA, and the sequence number of the traffic report respectively. The SA_j is incremented every time MR_j submits a traffic report to the SA. There are two types of statistics collected: ingress traffic (R, γ) that is received by a MR from its neighbor, and egress traffic (S, α, ζ) that is send out by a MR to a neighbor. In particular, we use the subscript → or ← to denote the direction of the traffic flow and use the curved brackets *( )* to denote the MR at which

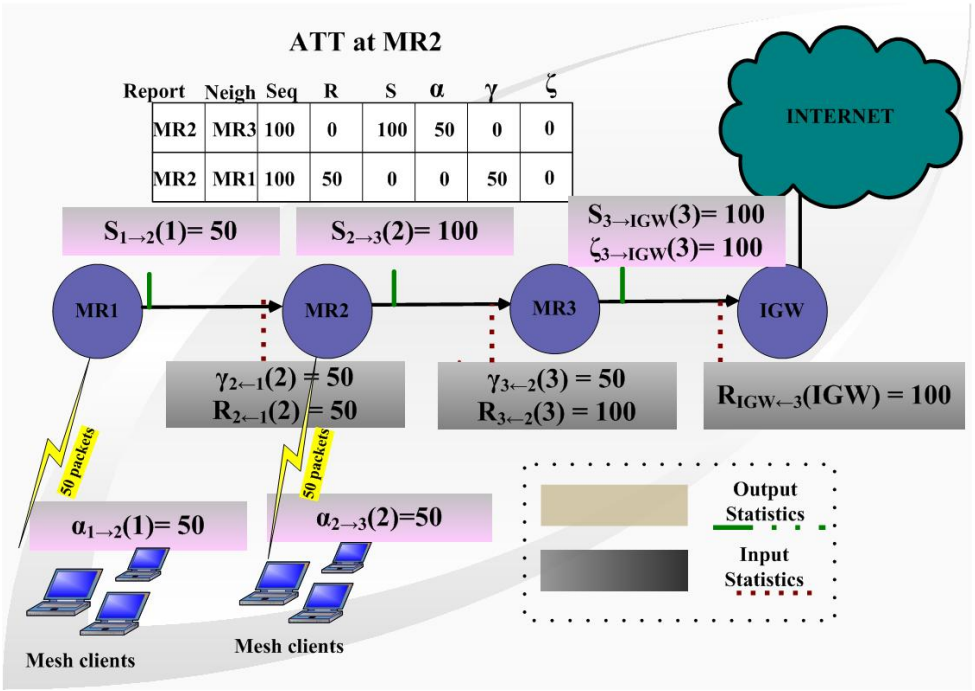


Figure 4.7: Illustration of a Traffic Scenario

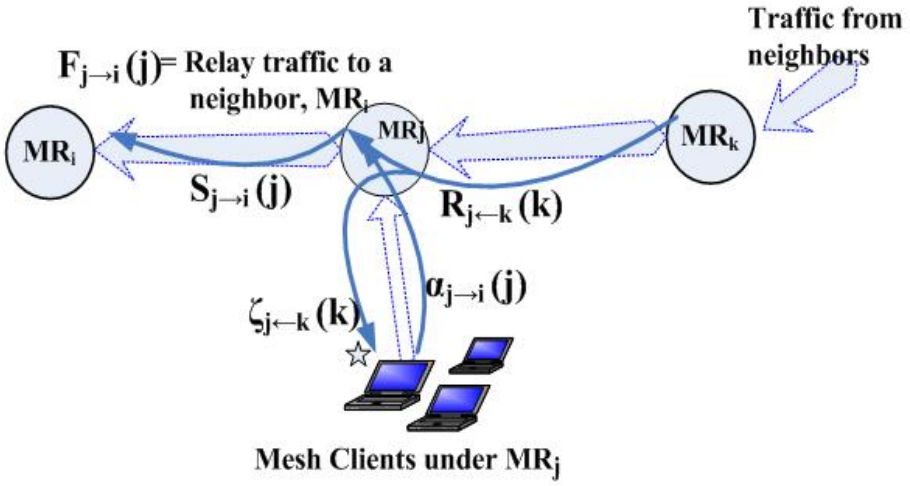
Table 4.1: Format of Traffic Report

Symbol	Description of the Symbol
MR_j	Reporting MR
MR_i	MR_i is neighbor of MR_j
SA_j	SA that MR_j is registered
Seq_j	Sequence number to identify the report of MR_j
$R_{i \leftarrow j}(i)$	Number of packets received at MR_i from MR_j, as reported by MR_i
$S_{j \rightarrow i}(j)$	Number of packets sent from MR_j to MR_i, as reported by MR_j
$\alpha_{j \rightarrow i}(j)$	Number of packets originated from MR_j and sent to MR_i, as reported by MR_j
$\gamma_{i \leftarrow j}(i)$	Number of packets originated from MR_j and sent to MR_i, as reported by MR_i
$\zeta_{j \rightarrow i}(j)$	Number of packets that are sent by MR_j and terminate at next hop MR_i, as reported by MR_j
$F_{j \rightarrow i}(j)$	Number of egress relay packets forwarded by MR_j to MR_i

the traffic statistics is collected. For example, in the symbol $R_{i \leftarrow j}(i)$, R is a statistics which represents the number of received packets. The subscript $i \leftarrow j$ implies that the direction of the traffic flow is from MR_j to MR_i and i after the symbol R denotes that the statistics is collected at MR_i. Similarly, $S_{j \rightarrow i}(j)$ is the number of packets sent from MR_j to MR_i, as reported by MR_j. $\alpha_{j \rightarrow i}(j)$ represents the number of packets originating from MR_j's mesh clients that are sent towards MR_i, as reported by MR_j. $\gamma_{i \leftarrow j}(i)$ denotes the number of packets originating

from MR_j's mesh clients that are sent to MR_i, as reported by MR_i. $\zeta_{j \to i}(j)$ is the number of packets sent from MR_j that terminate at the next hop MR_i, as reported by MR_j.

Using the above statistics, other useful statistics can be derived such as the number of packets of other MR's that is forwarded by a given MR (i.e., relay packets). The number of relay packets can be computed using the output statistics *S* and α. This is illustrated in Figure 4.8.



$F_{j \to i}(j)$: Outgoing relay traffic from MR_j = Incoming relay traffic to MR_j

$$S_{j \to i}(j) - \alpha_{j \to i}(j) = R_{j \leftarrow k}(k) - \zeta_{j \leftarrow k}(k)$$

☆ Sink Traffic

Figure 4.8: Illustration of a Traffic Report Format

We denote this value as *F*. Thus, the egress relay packet, *F*, is calculated by subtracting the total number of packets originating at a MR from the total number of packets sent by a MR. Specifically, $F_{j \to i}(j)$ denotes the egress relay packets forwarded from MR_j to MR_i computed using the traffic report of MR_j. This is again cross-verified against the ingress relay

packets computed using the statistics R and ζ, i.e., the difference between the received packets and the packets terminating at this node.

We further explain this table using the topology as shown in Figure 4.7 in which three MRs, i.e., MR1, MR2, MR3, and the IGW are arranged in a linear fashion. The MR1 originates *50* packets to be sent to IGW and MR2 originates 40 packets to be sent to IGW. If the intermediate MR2 is a good MR, it will forward all *90* packets towards MR3. In the following, we list all non-zero statistics collected at each MR in the above scenario.

At MR1,

$$S_{1 \to 2}(1) = 50 \text{ packets},$$

At MR2,

$$R_{1 \leftarrow 2}(2) = 50 \text{ packets},$$

$$\alpha_{2 \to 3}(2) = 40 \text{ packets},$$

$$S_{2 \to 3}(2) = 90 \text{ packets},$$

$$F_{2 \to 3}(2) = 40 \text{ packets},$$

At MR3,

$$R_{3 \leftarrow 2}(3) = 90 \text{ packets},$$

$$S_{3 \to IGW}(3) = 90 \text{ packets},$$

$$F_{3 \to IGW}(3) = 90 \text{ packets},$$

$$\zeta_{3 \rightarrow IGW}(3) = 90 \text{ packets},$$

Thus, at IGW,

$$R_{IGW \leftarrow 3}(IGW) = 90 \text{ packets}.$$

Thus, we collect traffic statistics such that a report at a MR can be evaluated by cross-verifying against the neighboring MR's traffic report. For example, the number of packets sent from MR1's egress interface to MR2 is compared against the number of packets received at the ingress interface of MR2 from MR1. Unlike SPRITE [43] that sends a report for every forwarded message, only a summary of a MR's traffic is submitted to the SA in the form of a traffic report. Thus, by compressing the statistics into a traffic report, we save valuable network bandwidth. Also, performing per-packet basis check is an onerous task requiring enormous computational capability at the SA. Thus, our structure of traffic reports effectively captures the MR's transactions in a concise and efficient manner for evaluation.

The D-POLICE algorithm is summarized in Table 4.2, which includes three stages: traffic reporting, aggregation of traffic table, and traffic inconsistency checkpoints.

Table 4.2: D-POLICE Algorithm

Traffic Reporting

Γ : $MR_i \in N$ under the request of SA_i {

Submit traffic report to SA_i

}

Aggregation of Traffic Table

Γ^{+} : SA_i upon the reception of traffic report {

Build the Aggregated Traffic Table (ATT) from the submitted traffic reports

Check for the completeness of ATT if it has all required traffic reports

If any missing report, request the appropriate MR for it

}

Traffic Inconsistency Check

Γ: SA_i checks ATT {

Validate the traffic report by using three checkpoints

Apply Check 1: Input-Output Packets

Consistency

Apply Check 2: Originating Packets

Consistency

Apply Check 3: Forwarded Packets

Consistency

}

1. Traffic Reporting

In the traffic reporting stage, all MRs submit their traffic report to their registered static SA or a mobile SA as required.

2. Aggregation of Traffic Table

In the aggregation stage, an Aggregated Traffic Table (ATT) is built at the SA based on the reports gathered from all MRs. The ATT, as shown in Figure 4.6 is designed for facilitating the traffic analysis. The SA groups all reports with the same sequence number (i.e., from same MR) and then examines the ATT to determine if it is complete, i.e., it determines if it has collected reports from all neighbors of a given node with which a transaction has been registered. This is important, as a node's traffic statistics (with a neighbor) can be validated only if the corresponding statistics collected at its neighbor is available. Thus, if there are any missing

traffic reports, it requests the required MRs to submit the missing reports. For this purpose, the mobile SA may need to visit these neighbors of the MR. In the case of static SA, if the MR and its neighbor, i.e., MR_i and $MR_j \in MR_j(i)$, are registered under different SAs, i.e., $SA_i \neq SA_j$, an individual SA cannot validate the consistency of traffic reports. In such a case, a request from SA_i is issued to SA_j for the missing entry.

3. Traffic Inconsistency Checkpoints

The traffic inconsistency check evaluates the trustworthiness of every MR's traffic reports by applying three checkpoints. Let $MR_i(j)$ denote the set of adjacent MRs of MR_j. Given a pair of adjacent MRs, *i* and *j*, the traffic inconsistency check is performed by applying three checkpoints.

Input-Output Packet Consistency

The first check called the Input-Output Packets Consistency, is used to evaluate if the number of packets sent from a MR (i.e., MR_j) is same as the number of packets received at the neighboring MR (i.e., MR_i). The traffic report from MR_j and MR_i should accord with each other. Thus, $S_{j \to i}(j)$ reported by MR_j should be same as $R_{i \leftarrow j}(i)$ reported by its neighbor, MR_i. To determine the trustworthiness of MR_j, such a check would have to be performed between MR_j and all its neighboring MRs, $MR_i \in MR_i(j)$:

$$S_{j \to i}(j) = R_{i \to j}(i) \quad \forall \ (MR_i \in MR_i(j)) \ . \qquad \textbf{(4.1)}$$

This computation prevents MR from dropping packets freely. This check also prevents MR from freely manipulating the value of relayed packets, F_j, which is calculated based on S_j. Continuing to use the same scenario as shown in Figure 4.7, we explain the above check. In order to validate the traffic reports, the SA checks if $S_{1 \to 2}(1) = R_{1 \leftarrow 2}(2)$ and similarly check if $S_{2 \to 3}(2) = R_{3 \leftarrow 2}(3)$ and so on. Say, MR2, behaves selfishly and drops all or part of the 50 packets of MR1. It cannot falsely modify $S_{2 \to 3}(2)$ value to be 90 packets and remain undetected. A mismatch is immediately flagged when $S_{2 \to 3}(2)$ is compared against $R_{3 \leftarrow 2}(3)$ (which will be 40 packets only).

Originating Packet Consistency

The second check called the Originating Packet Consistency, determines if a selfish MR is manipulating the statistics on the number of packets originating from its mesh clients. To hide its selfish behavior, a selfish MR can drop other's traffic and in its place generate corresponding amount of traffic from its mesh clients so that the total number packets at its egress interface remains the same as the previous case (when no packets are dropped). In such a case, Eq.(4.2) fails to determine this misbehavior of a selfish MR. In order to prevent any manipulation of self-originating traffic statistics, we cross verify the number of packets originating at a MR using the observation of its neighbor to which these packets are forwarded. For every neighboring MR that received the traffic of a given MR, we determine the total number of packets that originated from this MR (i.e., from its mesh clients). The neighboring MRs can count this value by examining the source address of the packets arriving at its ingress interface. Thus, we check if the number of

packets originating from MR_j (denoted as $\alpha(j)$) is equal to the number of received packets at MR_i that originated at MR_j (denoted as $\gamma(i)$) $\forall MR_i \in MR_i(j)$.

$$\alpha_{j \to i}(j) = \gamma_{i \leftarrow j}(i) \ \forall MR_i \in MR_i(j). \quad \textbf{(4.2)}$$

Hence, a selfish MR cannot originate traffic at the cost of dropping other's traffic and cannot freely manipulate the value of packets originating from its mesh client, α. We use Figure 4.6, to explain the above check. In order to determine if MR2 is a behaving selfishly or not, the SA checks if $\alpha_{2 \to 3}(2) = \gamma_{3 \leftarrow 2}(3)$. Say, if MR2 behaves selfishly and drops all the 50 packets of MR1. It cannot manipulate the value of $\alpha(2)$ to be 100 packets as $\gamma(3)$ would be registered as 50 packets.

Forwarded Packet Consistency

Eq (4.2) cannot prevent a MR from misreporting the number of packets originating from its mesh clients. The MR_j can always forge the source address in the packets and forward it to MR_i, which will results in an incorrect value of $\gamma_{i \leftarrow j}(i)$. For this, a check called the Forwarded Packet Consistency is applied to determine the number of relay packets forwarded by a MR. As explained previously, the number of relay packets transmitted by a MR_j towards a neighbor is $S_{j \to i}(j) - \alpha_{j \to i}(j)$, i.e., the difference between the total number of transmitted packets and the total number of packets originating form itself. In other words, this is the number of egress relay packets at MR_j. If MR is behaving correctly, this value should be same as the number of ingress relay packets at MR_j. We can compute the number of ingress relay packets obtained at

MR_j based on an egress statistic observed at MR_j's neighbor MR_k namely $\zeta(k)$ and the ingress statistics at MR_j, namely *R(j)*. The value $\zeta_{k \to j}(k)$, as reported by MR_k represents the number of packets that will terminate at MR_j, among the packets sent from MR_k to MR_j. Thus, the accumulated sum of the number of relay packets received at the ingress interface of MR_j should always be greater than the accumulated sum of the number of relay packets sent at the egress interface of MR_j as shown below:

$$\sum_i R_{j \leftarrow k}(j) - \sum_i \zeta_{k \to j}(k) \geq \sum_i S_{j \to i}(j) - \sum_i \alpha_{j \to i}(j) \qquad \textbf{(4.3)}$$

$$\forall MR_i \in MR_i(j),\ k \in MR_k(j).$$

We explain the above check using the linear scenario in Figure 4.6. The SA computes the number of relay packets at the ingress interface of MR2 using $R_{1 \leftarrow 2}(2)$=50 packets and $\zeta_{1 \to 2}(1) = 0$ packets. Thus, the total number of relay packets received at ingress interface of MR2 is 50 packets. Next, the SA checks this quantity against the number of relay packets sent at the egress interface of MR2, using $S_{2 \to 3}(2) = 100$ packets and $\alpha_{2 \to 3}(2) = 50$ packets. Thus, the total number of relay packets at the egress interface of MR2 is 50 packets. If MR2 behaves selfishly and misreports on the value of $\alpha(2)$, it is immediately caught when the above check is applied. Hence, the above check stringently enforces a MR from misreporting. Hence, the SA concludes that MR2 is dutifully performing packet forwarding.

If the traffic reports satisfy the aforementioned credibility tests, the SA infers that all MRs in the network are cooperatively forwarding each other's traffic. As there is a possibility of some

packet loss occurring due to interference/channel degradation/queuing overflow in a wireless channel, there should be a provision not to misinterpret this as selfishness. Hence, when the three checkpoints are applied, we always consider a certain degree of permissible packet drop for a given network condition.

4.6.2 Selfish MR Detection Algorithm

The purpose of free-rider detection algorithm is to identify and punish the free-riders. The SA gathers κ sets of reports from each MR every τ_{report} time, i.e., $\tau_{check} = \kappa * \tau_{report}$ According to the inconsistency of these reports, the free-rider detection algorithm applies some pre-defined rules to identify the selfish MRs as follows.

A. Number of Alleged Manipulation

After applying the three checkpoints on its traffic reports, each SA checks if the reports from two adjacent MRs accord with each other. If not, the MR and the neighbor involved in the transaction are added to a suspicion list called NAM (Number of Alleged Manipulation) list. In the static mode of operation, there exist multiple SAs. Hence, examining the inconsistencies at a single SA the identity of the free-rider is unclear, as member MRs involved in the alleged manipulations might belong to other SAs. In this case, one SA is chosen as a Sink Manager (SM) to which all other sinks unicast their NAM list and a master NAM list is created at the SM. As only the suspicious node list is passed to the SM, D-POLICE incurs lesser overhead in evaluating reports and lesser congestion at the SA. In the mobile mode of operation, the SM and mobile SA remain the same, as the mobile SA is the only repository of all gathered traffic reports.

B. Inconsistency Record Table

The Inconsistency Record Table (IRT) is used for the purpose of identifying the selfish MR by imposing penalty on defaulting MRs found in the master NAM list. When a SM registers an

inconsistency about a suspicious neighbor transactions (MR_i, MR_j), it is hard to determine if MR_i is selfish or MR_j is selfish.

However, if MR_i repeatedly continues its selfish behavior, it is bound to be caught in another inconsistency with some other neighbor MR_k, where $k \in MR_k(i)$. Hence, the IRT defines a rule called Additive Increase and Multiplicative Decrease (AIMD) to identify the selfish MR. The AIMD scheme additively penalizes the selfish MR and multiplicatively decrease any false penalty imposed on an innocent MR. This way, we systematically eliminate the innocent MRs and identify the repeatedly cheating selfish MR(s). Table 4.3 shows an IRT built at the SM using a master NAM list.

Table 4.3: Inconsistency Record Table

	MR_1	MR_2	MR_3	…	MR_n	Total Penalty
MR_1	-	$\omega_{i,j}$	$\omega_{i,k}$	…	$\omega_{i,n}$	$\sum_{l=1}^{n} \omega_{i,l}$
MR_2	$\omega_{j,i}$	-	$\omega_{j,k}$	…	$\omega_{j,n}$	$\sum_{l=1}^{n} \omega_{j,l}$
MR_3	$\omega_{k,i}$	$\omega_{k,j}$	-	…	$\omega_{k,n}$	$\sum_{l=1}^{n} \omega_{k,l}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
MR_n	$\omega_{n,i}$	$\omega_{n,j}$	$\omega_{n,k}$	…	-	$\sum_{l=1}^{n} \omega_{n,l}$

Let *n* be the total number of MRs in the network. Let $\omega_{i,j}$ denote the penalty value on a pair of MR (MR_i, MR_j). Let ψ and ψ' denote the update functions of $\omega_{i,j}$. Each entry in the IRT

represents the alleged manipulations (NAM) in the transactions between MR_i and MR_j. The last column denotes the total penalty imposed on each MR. If this is greater than a certain threshold *Upper threshold* and the MR has not defaulted before, the MR is blacklisted and added to a blacklisted node history. This threshold is further lowered to *Lower threshold* for a MR in the blacklisted node history. We have two different thresholds as our design gives the selfish MR a second chance to enter the network after sometime. This will be explained further in the later section.

Each entry in the IRT is updated by an AIMD function. For example, if there is an inconsistency in the reports of MR_i and MR_j, we increase the penalty according to Eq. (4.4).

$$\psi(\omega_{i,j}) = \omega_{i,j} + 1 \text{ and } \psi(\omega_{j,i}) = \omega_{j,i} + 1. \quad \textbf{(4.4)}$$

As other innocent MRs, MR_k involved in a transaction with a selfish (MR_i or MR_j) are penalized, penalty on other MRs are reduced by half as given by Eq. (4.5) and Eq.(4.6).

$$\psi'(\omega_{k,i}) = \left\lfloor \frac{\omega_{k,i}}{2} \right\rfloor \ \forall k \notin \{i, j\}. \quad \textbf{(4.5)}$$

$$\psi'(\omega_{k,j}) = \left\lfloor \frac{\omega_{k,j}}{2} \right\rfloor \ \forall k \notin \{i, j\}. \quad \textbf{(4.6)}$$

Figure 4.9 shows the changes in the IRT table whenever an inconsistency is detected in the transactions in the topology shown in Figure 4.2. Say, MR5 is the selfish MR which misreports on the number of relay packets forwarded and the upper threshold for selfish node detection is

Threshold: 5

Inconsistency Record Table	
No.	MR
1.	*MR2-1*
2.	*MR5-2*
3.	*MR4-5*
4.	*MR5-4*
5.	*MR5-2*
6.	*MR2-5*

#1	5	2	4	1	Total
5	-	0	0	0	0
2	0	-	0	**1**	**1**
4	0	0	-	0	0
1	0	**1**	0	-	**1**

#2	5	2	4	1	Total
5	-	**1**	0	0	**1**
2	**1**	-	0	1	**2**
4	0	0	-	0	0
1	0	**0**	0	-	**0**

#3	5	2	4	1	Total
5	-	1	**1**	0	**2**
2	**0**	-	0	**1**	**1**
4	**1**	0	-	0	**1**
1	0	0	0	-	0

#4	5	2	4	1	Total
5	-	1	**2**	0	**3**
2	0	-	0	1	1
4	**2**	0	-	0	**2**
1	0	0	0	-	0

#5	5	2	4	1	Total
5	-	**2**	2	0	**4**
2	**1**	-	0	1	**2**
4	**1**	0	-	0	**1**
1	0	0	0	-	0

#6	5	2	4	1	Total
5	-	**3**	2	0	**5**
2	**2**	-	0	1	**3**
4	**0**	0	-	0	**0**
1	0	0	0	-	0

Figure 4.9 Inconsistency Record Table Computations

set as five. While the first inconsistency penalizes MR2 and MR1, the second report removes the penalty imposed on MR1 and increases the penalty for MR2.

However, with the aid of further inconsistency reports, the ambiguity on the identity of selfish MR is resolved. We steadily increase the penalty on selfish MR5 with each inconsistency report and at the same time remove the false penalty imposed on the innocent MR2. Thus, this example illustrates the accuracy of AIMD approach in punishing the selfish MR.

C. Penalty on Selfish MR

Once a blacklisted MR is detected, the affected MRs should have a policy for rerouting their traffic and the selfish MR should be penalized. Upon receiving this message, for example, the MRs that have a route through this blacklisted MR, invalidate their entries and take appropriate rerouting action. This can be either performed by a local repair or by rediscovering a new route.

Thus, the affected MRs now reroute their traffic through the alternate paths. Selfish behavior is discouraged as all legitimate MRs collectively refuse to forward any traffic originating from the blacklisted MR. SM maintains the list of all previously blacklisted MRs along with the observed time of its misbehavior. A free-rider can be immediately isolated from the system or allowed to re-enter the network after an interval of time called *Forgiven time*. The Forgiven time imposes conservative punishment and avoids misjudgment in the network. On the expiration of this timer, the system temporarily pardons the isolated MR essentially to give a second chance. Other MRs henceforth resume routing through this MR. However, if the MR begins its selfish activity again and is found in the NAM list, its threshold for the IRT table computation is lowered to *Lower threshold* as a precautionary measure. Using the IRT computation, if it is found to default again, it is blacklisted and the *Forgiven time* is exponentially increased.

4.7 Performance Analysis

In this section, we first illustrate the impact of selfishness on the network performance. We evaluate the performance of D-POLICE using open source network simulator (ns-2) [15]. Although D-POLICE can be run over any routing protocol, we choose AODV (Ad hoc On-demand Distance Vector routing) as the routing protocol. We start flows from the clients that are being serviced by the MRs. Henceforth, when we say a flow is started from a MR, we imply that the clients registered under the MR has started a flow. The flow at a MR represents the aggregate flow from all clients registered under it. The IEEE 802.11 standard is used for channel arbitration with the transmission range and channel capacity set to 250 m and 11 Mbps respectively. The total simulation time is set to 200 sec. In order to reduce the variance in statistics we take an average of 3 runs per scenario. We set D-POLICE specific parameters as

follows: *Check Round* (28 sec), Report Round (7 sec), Lower Threshold (1 sec), Forgiven Time (14 sec), Beacon Max Hops (2), and Upper Threshold (3 sec).

4.7.1 Selfishness Exposed

In this sub-section, we illustrate the problem of selfishness and its deleterious impact on the network performance. We consider a hierarchical architecture of WMN as shown in Figure 4.10 and demonstrate the inherent dependencies in the routing paths that can be exploited by a selfish MR.

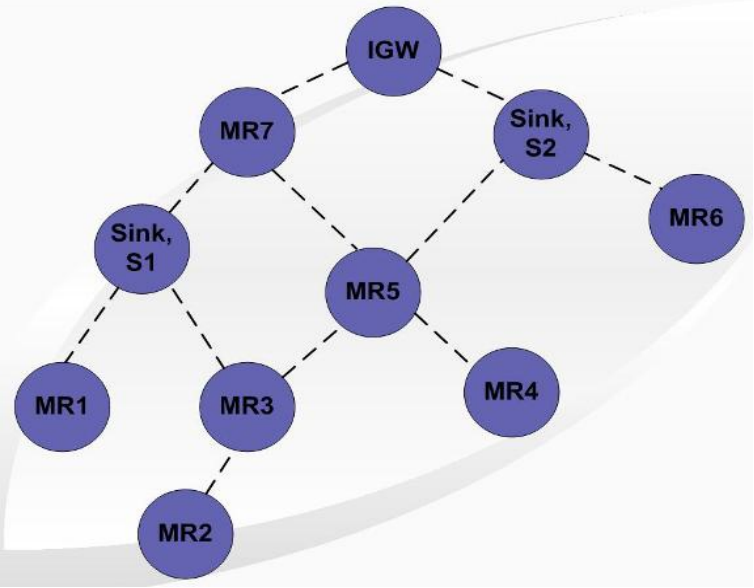


Figure 4.10 WMN Scenario

Figure 4.11 shows the instantaneous throughput of four UDP flows of 160 Kbs with a packet size of 512 Kbytes (Flow 1: MR5 → IGW, Flow 2: MR3 →IGW, Flow 3: IGW → MR4, and Flow 4: MR7 → IGW). We place a selfish node (MR7) along the path to the IGW, which probabilistically drops packets from other MRs, with a probability of 50%. We clearly see that when the selfish node MR7 starts forwarding its traffic at $t = 20$ sec (Flow 4), other flows receive only 80 kbps throughput. The proximity of a selfish node to the IGW determines its level of deleterious impact on the WMN. We next place a selfish node in the periphery of the network

and determine its level of impact. We set up 4 UDP flows of 160 Kbps in the network (Flow 1: MR2→IGW, Flow 2: MR3→IGW, Flow 3: IGW→MR4, Flow 4: MR7→ IGW). The MR3 begins to behave selfishly at *t = 20* sec and Flow 2 originating from the selfish MR3 enjoys full throughput. As the selfish node is in the periphery of the network, it affects only some flows that are routed through it (Flow 1: from MR2). We see from Figure 4.12, that other flows (Flow 3: to MR4, Flow 4: from MR7) remain unaffected.

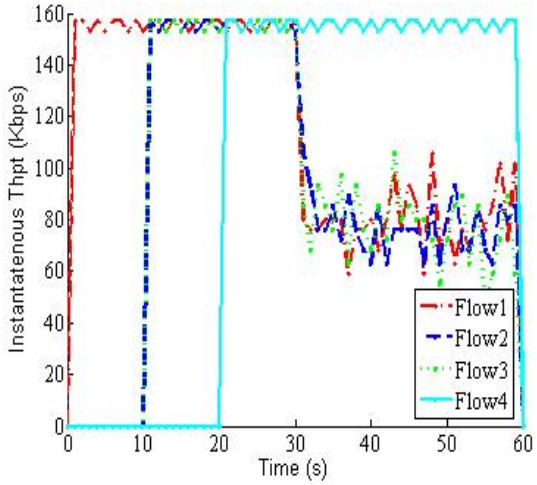


Figure 4.11: Selfish node near IGW

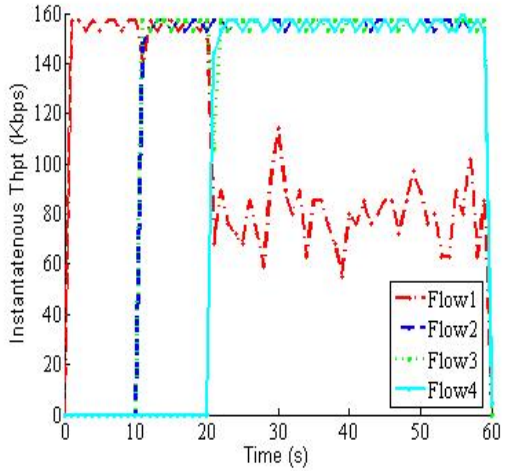


Figure 4.12: Selfish node in the periphery

4.7.2 Instantaneous Throughput

We next illustrate the instantaneous throughput of flows in the presence of selfish MRs. We operate the D-POLICE in the static mode and consider a hierarchical architecture of WMN with static SA uniformly deployed in the network as shown in Figure 4.10. We begin a set of 4 UDP flows of rate 150 Kbps from 4 different MRs (Flow 1: MR5→IGW, Flow2: MR3→IGW, Flow 3: IGW→MR4, and Flow 4: from MR7→IGW). The flow duration is taken as 190 sec. MR7 is considered as the selfish node which fully drops other's traffic (100 % drop). As seen from the Figure 4.13, during the time period 10-30 sec, Flow 1 from MR5 suffers from 100% packet loss,

while Flow 4 from MR7 enjoys good throughput. Flow 2 and Flow 3 also suffer significant losses. However, this free-riding does not continue for a long period of time. After four rounds (nearly 30 sec) of continued misbehavior by MR7, the D-POLICE confirms MR7 as a free-rider and broadcasts its identity to the entire network. The IGW now stops forwarding the traffic of selfish MR7 traffic to the Internet.

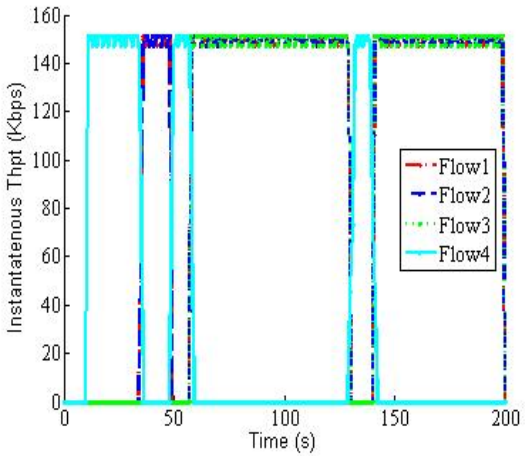


Figure 4.13: Instantaneous throughput with D-POLICE

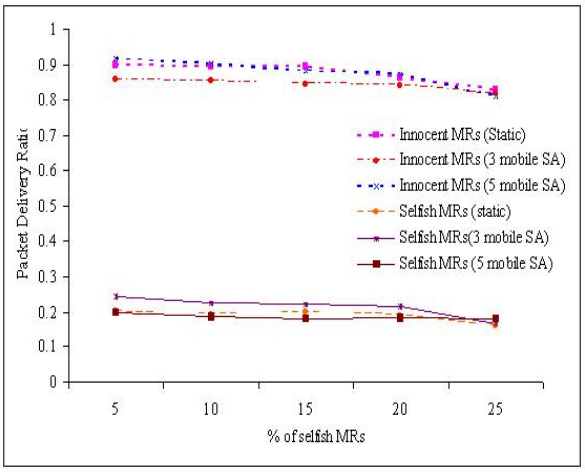


Figure 4.14: PDR vs. % of selfish MRs

In order to illustrate the punishment for prolonged misbehavior, we start another flow from IGW to MR4 (Flow 3) shortly after the selfish MR is forgiven (t = 50 sec). The IGW routes Flow 3 to MR7. However, as MR7 continues its misbehavior, Flow 3 suffers 100 % packet loss. This can be seen during the time period 45-58 sec in Figure 4.13. The D-POLICE quickly identifies the misbehavior of MR7 (based on reports gathered within one round) and blacklist it for a time that is chosen with an exponential backoff timer. It notifies the entire network to re-route traffic through other alternate path. Now, MR7 which is blacklisted will not be able to route any traffic in the network until t = 129 sec. We conservatively allow the selfish MR to re-join the network and at the same time disconnect it incase of any misbehavior. This can be seen from the Figure 4.13 during the time period 141-200 sec.

4.7.3 Packet Delivery Ratio

We now evaluate the effectiveness of D-POLICE in the presence of multiple selfish MRs. In all our subsequent evaluations we consider a network of 25 MRs in a 5 x 5 grid spread over an area of 1500m x 1500m. We randomly pick about 10 distinct source MRs and start UDP flow of 100-150 Kbps from the mesh clients under the MR. We place the IGW in the center of the grid. We operate the D-POLICE scheme in the mobile mode and the static mode. In the mobile mode, the mobile SA is configured to perform random walks of the network. We study the performance of our scheme by employing varying number of mobile SA. In the static mode, we uniformly deploy four static SA at reliable core MRs in the network. We perform on-demand evaluation of reports, with the reporting instance decided randomly by the SA. For the clarity of the plots, we do not show periodic reporting which also yields similar performance as obtained by on-demand reporting.

We measure the Packet Delivery Ratio (PDR), which is the ratio of the number of packets received at the destination to the number of packets generated at the source. Figure 4.14 shows the PDR of the innocent and selfish MRs under varying percentage of selfish MRs for different operation modes of the D-POLICE scheme. We see that the PDR of innocent MRs is well maintained and the PDR of selfish MRs is kept at a low value. Even though innocent MRs may occasionally loose packets because of the presence of selfish MRs in their path, they quickly reroute their traffic, and maintain a steady PDR of close to 90 % in all modes of D-POLICE. As D-POLICE gives a second chance to the misbehaving MR, the PDR of free-riders is low, but non zero (about 20%). We also see that the performance of D-POLICE in the static mode and mobile mode (with 5 mobile SA) is identical. The mobile mode with just 3 mobile SA yields slightly lower performance as compared to the other two modes. This is because three mobile SAs are

insufficient to systematically cover the network. Hence, some parts of the network might be uncovered by the mobile SA, resulting in the selfish MRs going undetected for some time. We also note that as the number of selfish MRs increases, the PDR of the innocent MRs gradually decreases. This is because, with increased number of selfish MRs, the paths leading to the IGW become longer and more circuitous. However, D-POLICE latches the PDR of innocent MRs at 80% even when 25% of the MRs in the network behaves selfishly.

4.7.4 Average Delay of Flows

In this section, we discuss the average delay experienced by the flows in the network due to the presence of selfish MRs. The average delay represents end-to-end delay experienced by each packet traversing from its source to its destination. We study the average delay experienced by all innocent flows and do not include a flow from selfish node in our measure. This metric encapsulates the detection time to determine the selfish MR and announce its identity to the network so that rest of the MRs re-route their traffic through other alternate paths.

As we see from Figure 4.15the average delay experienced by the flows is very negligible in the order of few msecs. But, as the number of selfish MRs increases to 25%, the average delay of flows increases to a maximum of 14 msecs. This is because in the presence of large number of selfish MRs (after the D-POLICE identifies the selfish MRs), the packets are re-routed along circuitous paths to the IGW. This increases the average end-to-end delay of flows. We see from Figure 4.15 that the average delay of flows in the mobile mode is less than the delay in the static mode. This is because mobile SA performs localized detection in each region. It gathers the traffic reports from all MRs as it traverses along a given path, evaluates them, and determines if there is any selfish MR lying in this path. This results in smaller processing and quicker

detection and enables quicker remedial actions to remove the selfish MR. In the static mode, however, the traffic reports are routed towards the SA and the MRs might be 2-3 hops away from the SA depending upon the deployment of static SA. Thus, SA has to wait until all reports from all the MRs arrive. The figure also illustrates the average delay of flows when using periodic reporting and on-demand reporting mode of D-POLICE. As on-demand mode gathers reports for a specific random interval of time, it incurs lesser routing time in the submission of reports and lesser time in evaluating the reports and thereby enables quicker detection of selfish MRs. Thus, the average delay experienced by flows reduces. The mobile on-demand mode of D-POLICE incurs the least end-to-end delay in routing flows.

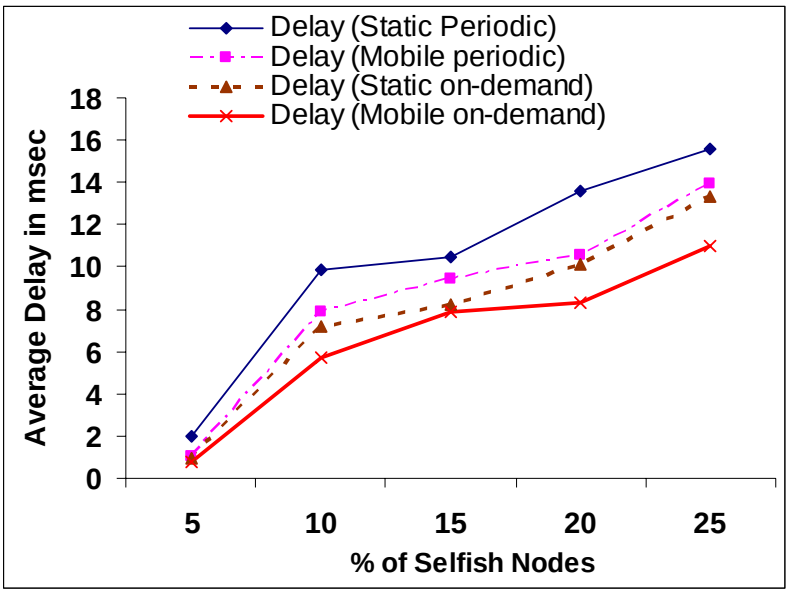


Figure 4.15: Average Delay of Flows vs. % of Selfish MRs

4.8 Conclusion

The MRs form the backbone of a WMN. As cooperative relaying is the fundamental theme of WMNs, a secure framework needs to be built in the backbone to combat selfishness. We have highlighted inadequacy of credit/reputation based schemes in promoting cooperation in WMN. We have presented the distinct motivation for selfishness in WMNs and have described the

selfish adversary model of a MR in WMN that exhibits packet dropping, channel riding, radio selfishness and misreporting. We have developed a distributed policing architecture called D-POLICE that employs Sink Agents to infer selfishness of MRs by gathering traffic reports from all MRs. We designed three effective checkpoints that are guaranteed to raise inconsistency in traffic reports due to selfish behavior. We have also designed a dynamic penalty mechanism that accurately identifies the selfish MR based on the recorded inconsistencies.

We have also described various modes of operation of D-POLICE to efficiently gather traffic reports. Specifically, D-POLICE can be operated in a static or mobile mode and can be made to gather reports periodically or at random intervals. The simulation results indicate that it is beneficial to operate the D-POLICE scheme in the on-demand reporting mode due to the lower communication overhead in sending traffic reports. The results also fortify that all modes of D-POLICE effectively improve the network throughput, even in a largely compromised network. D-POLICE effectively isolates the selfish MRs and re-routes the traffic through other alternate paths. It also tries to re-accommodate a quarantined selfish MR after some time and hence imposes conservative punishment to avoid misjudgment.

Chapter

5. Perceptron based Intrusion Detection System

5.1 Introduction

WMNs are becoming increasingly popular as they provide cheap and ubiquitous broadband Internet access. WMNs consist of a group of static MRs that are connected by wireless links. It obviates the need for extensive wired backhaul network by connecting only a small subset of MRs (known as IGWs) to the wired backbone. Other MRs forward their traffic in a multihop fashion towards the IGW. On the other hand, open wireless communication channel and the multihop nature of communication can pave way to malicious intruders. A malicious intruder can exploit the hidden loopholes in the routing protocol [56], to conduct various kinds of attack such as route disruption attacks (e.g., blackhole attack, selfish node attack, malicious route flood, route table poisoning, route looping, modification of route requests) and packet forwarding attacks (e.g.,

selfish node attack, Denial of Service (DoS) attack). Bhargava et al. [57] demonstrated that a false distance vector and a false sequence number attack can drastically reduce the network throughput by 75%. Building a secure routing protocol (SEAD [58], Ariadne [59]) is not a complete solution to thwart attacks on the routing protocol, as MRs are often deployed in public locations (like rooftops, streetlights, poles etc.) that are easily accessible to potential adversaries.

In this chapter, we study the problem of detecting malicious nodes that imitates a normal node in all respects, except in performing unnecessary route discoveries. These malicious nodes frequently initiate route discovery to unknown destinations with intent to flood the network with route request packets. As it is difficult to distinguish between a route discovery initiated with a malicious intent and a legitimate route discovery for repairing broken/stale routes, this type of attack is hard to detect. Though, several broadcast management techniques exists that try to alleviate redundant broadcasts [60]; they fail to address the problem of bogus broadcast packets. Desilva et al. [61] propose a statistical method to mitigate malicious route floods; in which when the generation rate of a node exceeds a certain threshold, all the route requests of that node are dropped by its neighbors. We however, detect the same based on the examination of the network.

In order to detect malicious route floods in a WMN, we propose to deploy an Intrusion Detection System (IDS) on every MR in the network. As WMNs don’t suffer from power constraints, MR can be made as a monitoring agent. We propose an IDS based on a machine learning technique called perceptron training. The IDS collects a set of statistics on few route and data parameters such as RREQ (Route Request), RREP (Route reply), and RERR (Route error), data packets originated, and data packets forwarded. The above listed parameters are then fed to the perceptron model in order to learn the normal routing behavior and establish a normal routing profile. Typically, it is difficult to judge a safe threshold value for the number of RREQs packets

that can be generated by a MR in a WMN as it depends on several other network conditions like link failure, stale route, network congestion, etc. Thus, we propose a threshold independent scheme called *perceptron based IDS* to detect malicious route flood attack. It adaptively modifies the detection model based on the training data and detects attack in the face of new data.

The remainder of this chapter is organized as follows: Section 5.2 describes the related work. Section 5.3 presents various loop holes in the route discovery phase of Ad hoc On-Demand Vector (AODV) routing protocol. We study the impact of a malicious route flood attack through simple simulations. Section 5.4 delineates the architecture of our perceptron based IDS. Section 5.5 illustrates the effectiveness of our proposed scheme through extensive simulations. We finally conclude this chapter and present the future work in Section 5.6.

5.2 Related Work

Most of the traditional IDS, detect attack by processing audit trails for deviations from normal activity (anomaly detection [62]) or by matching typical intrusion instances (misuse detection [63]). The main drawback of misuse detection is that, it cannot detect new attacks. On the other hand, an anomaly detector successfully detects unknown intrusions also. But, if an attacker changes its profile slowly, even well-known attacks remain undetected.

Intelligent machine learning techniques like perceptron, artificial neural networks (ANN), and decision trees can be used to train the IDS accurately. ANN has innumerable advantages in the detection of intrusions [64]. Zhang et al. [65] propose to detect DoS attacks using a hybrid ANN classifier based on perceptron and back propagation. It derives a statistical similarity decision vector between the observed profile and the established normal profile. As in the presence of a DoS attack the similarity vector is very small, an alert is raised by the IDS. But, the detection metrics for a route flooding attack and DoS attack are different. Hence, such a scheme is not

suitable for detecting misbehavior in the routing protocol. Ghost et al. [66] employ a neural network to detect anomalous intrusions on a software system (Buffer overflow attack). Siaterlis et al. [67] employ a Multi-Layer Perceptron (MLP) to detect Distributed DoS attacks by monitoring several passive flow statistics. Thus, most of the works in the literature focus on developing an IDS for detecting DoS attacks only and are unsuitable for detecting route floods. We however, focus on the problem of detecting malicious route floods by using the intelligence provided by our perceptron model. Huang et al. [68] propose a cooperative detection system by electing Custer Heads (CHs). The CHs apply simple rules to identify various route attacks. As the CHs are elected randomly, there is a possibility that a malicious attacker can be elected as a CH which would be detrimental to the network. We however, employ a distributed approach to detect route attacks that also decreases the detection time when compared to [68].

5.3 Malicious Route Flood Attack and its Impact

In this research, we use an AODV routing protocol for our analysis. In this section, we explain the route discovery phase in the AODV protocol and highlight the vulnerabilities in it that can be exploited by an attacker. We then investigate the impact of a malicious route flood attack on the network.

5.3.1 Vulnerability of AODV Routing Protocol

AODV is a reactive ad hoc routing protocol. A source node in AODV initiates a route discovery to a destination only if its route entry is unknown or old. It broadcasts a route request packet (RREQ) requesting for a route to the required destination. This is repeatedly re-broadcasted by other nodes until it reaches the destination itself or an intermediate node with a fresh enough route. However, a malicious node can periodically generate large number of false RREQ packets to non-existent destination nodes so that it is repeatedly re-broadcasted till the

lifetime of the packet. This attack is known as a *malicious route flood attack*. This creates a deluge of packets which drastically affects the achievable throughput at the application layer. If the network is saturated, a malicious flooding further aggravates the network performance [61]. An attacker can intensify the effect of a malicious route flood by further manipulating the protocol. Typically, each RREQ packet is associated with an identification number (ID) to prevent redundant broadcast. A malicious node can modify the ID of a RREQ packet so that it always appears to be a fresh request to other nodes and is repeatedly re-broadcasted by them. A malicious route flood attack results in route invasion, packet dropping, and network congestion.

5.3.2 Impact of Malicious Route Flood Attack

In this sub-section, we study and illustrate the impact of a malicious route flood attack in WMNs through simulations in ns-2 [15]. We consider a simple IEEE 802.11s based mesh network with 49 MRs (7 x 7) deployed in a grid like fashion in an area of 1500 x 1500 meters. We randomly attach 2-3 mesh clients to each of the MRs. One MR is selected as the IGW. The MRs communicate with each other using legacy IEEE 802.11 based interfaces. We start flows from few mesh clients (attached to MRs). We initiate 30 UDP flows, sending traffic at a constant rate of 200 Kbps. We use a constant packet size of 512 bytes. We use IEEE 802.11 as the channel arbitration protocol. The transmission range and the channel capacity are set to 250 meters and 11 Mbps respectively. AODV is used as the routing protocol. The total simulation time is set to 500 secs. Each simulation is repeated with 10 different traffic profiles containing randomly chosen traffic sources. We randomly choose one MR as malicious, which periodically generates (every 1 sec) large number of RREQ packets to arbitrarily chosen unknown destinations. We consider the case when all MRs are backlogged with traffic.

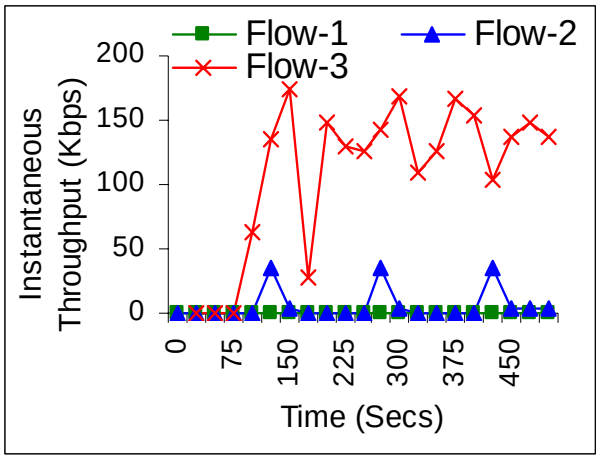


Figure 5.1: Instantaneous throughput of flows during attack

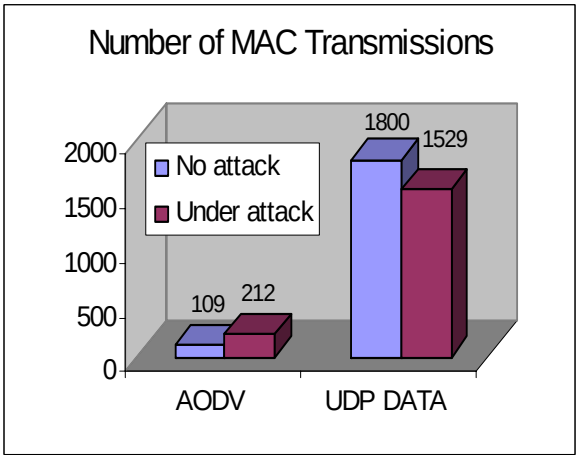


Figure 5.2: Average Number of Packets Transmitted by each MR

Figure 5.1 shows the effect of a malicious route flood (1 RREQ/ sec) on the instantaneous throughput of flows at the IGW (all flows not shown for clarity), under a randomly chosen traffic profile. We see that while the throughput of *flow-3* is affected by a small percentage (45%) only, the throughput of *flow-1* and *flow-2* are drastically reduced by 99% in the presence of a malicious route flood attack. Figure 5.2 illustrates the average number of packets transmitted at each MR in the presence and the absence of a malicious route flood attack. It can be seen from Figure 5.2 that during an attack, the average number of RREQs packets received at each MR is abnormally high (212), while the average number of data packets transmitted by each MR is cut down from 1800 to 1529. This is due to the fact that control packets are given higher priority over data packets to prevent link breakages.

We next study the effect of a malicious MR generating RREQs at different rates (1-10 RREQs /sec), for different offered load. We clearly see from Figure 5.3 that as the generation rate of malicious requests increases, the aggregate throughput of flows drastically decreases. Thus, a malicious route flood attack is a serious threat.

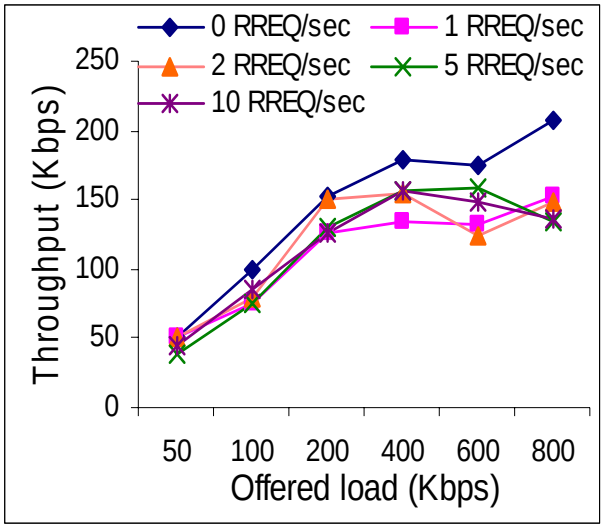


Figure 5.3: Throughput obtained vs. offered load

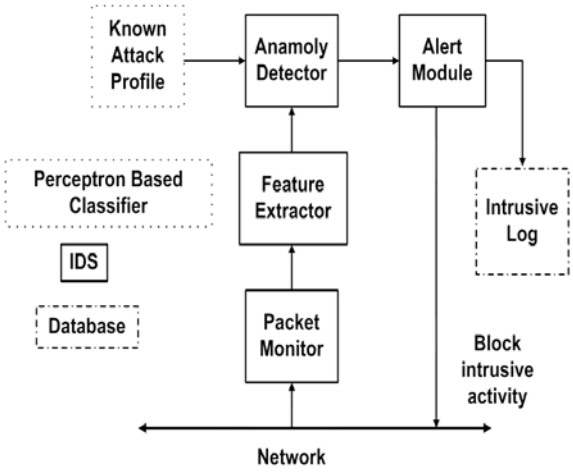


Figure 5.4: Architecture of perceptron based IDS

5.4 Perceptron Based IDS

In this section, we present the application of a machine learning technique called perceptron in IDS. We first present the high level architecture and then discuss the configuration of our perceptron based classifier.

5.4.1 System Architecture

An IDS is deployed at every MR in the WMN. The architecture of the proposed IDS is shown in Figure 5.4. It consists of the following components:

- Packet Monitor & Feature Extractor:

Packet monitor at every MR intercepts the incoming traffic and passes it to the feature extractor. Feature selection plays an important role in determining the accuracy of any classifier. The feature extractor gathers a set of statistics for each of its neighbors $|N_j|$ (where $|N_j|$ represents the set of neighbors of a node N_i). A sample snapshot of the features collected is shown in Table. 5.1.

For broadcast packets like RREQ, we maintain only the number of packets sent by a node's neighbor. But, for unicast packets like RREP, we gather two statistics, i.e., the number of packets sent by node's neighbor with destination as the node itself and with destination as some other node; the latter being collected by listening in promiscuous mode.

Table 5.1: Sample snapshot of nodal table

# RREQ sent (N_j to N_i)	number of route request packets
# RERR sent (N_j to N_i)	Number of route error packets
# RREP sent (N_j to N_i)	number of route reply packets
# Packets originated (N_j to N_i)	number of data packets originated with source address as self
# Packets forwarded (N_j to N_i)	number of data packets forwarded with source address as self

- Anomaly Detector:

  It consists of the trained perceptron model. The monitored statistics is fed as input to the perceptron classifier, which gives a binary output (t). A positive output is indicative of a normal condition and a negative output is indicative of an attack. An alert is issued by the alert module on detecting attack to block the intrusion and a report is lodged in the intrusive log.

- Known Attack Profile:

  The training data for the perceptron model is generated by conducting experimental simulations using ns-2 [15]. The known attack is grafted into the simulation and the accumulated network statistics gathered during this period are labeled as attack instances.

5.4.2 Perceptron Training Algorithm

We model the detection of malicious flood attack as a typical classification problem. Perceptron is used as a classifier to assess the network status based on the monitored network activity. Perceptron is the simplest form of a linear classifier [69]. The configuration of a linear perceptron is shown in Figure 5.5. As seen in Figure 5.5, a perceptron accepts a linear combination of real-valued inputs; and outputs 1, if the result is greater than a certain threshold and outputs -1, otherwise. For a given set of n inputs x_1, x_2, …, x_n, the output is,

$$o(x_1, x_2, ..., x_n) = \{1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n > 0$$

$$\text{-1 otherwise}\}, \qquad \textbf{(5.1)}$$

where each w_i denotes the contribution of each input x_i to the perceptron output. Here, w_0 denotes the threshold that the linear combination of inputs must surpass for an output of 1.

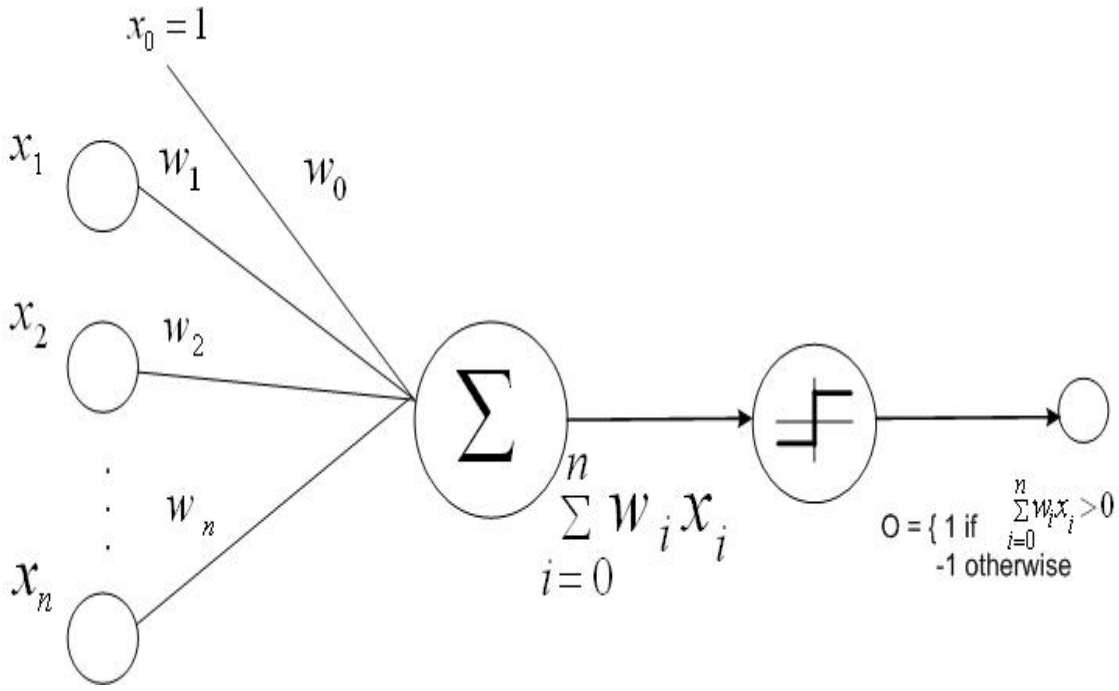


Figure 5.5: Linear Perceptron Model

For simplification, we augment an additional unity constant $x_0 = 1$ to the original input vector so that the above equation can be rewritten in a simpler form as,

$$o(x_1, x_2, ..., x_n) = \{1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0$$

$$\text{-1 otherwise}\}, \quad \textbf{(5.2)}$$

The perceptron training algorithm involves the evolution of the weight vector $\vec{w}$, which would correctly classify all the instances presented to it in the training stage. We input a feature vector of route and data parameters to the perceptron along with the label (ω_1 or ω_2), which represents the class of the instance (where ω_1 represents a normal instance and ω_2 represents an attack instance). We begin the training with some random assignment of weights. When the perceptron misclassifies an example, the weights are actively adapted according to the following perceptron training rule,

$$w_i = w_i + \eta x_i \text{ if } w_i x_i \leq 0 \text{ and } x_i \in \omega_1, \text{ and}$$

$$w_i = w_i - \eta x_i \text{ if } w_i x_i \geq 0 \text{ and } x_i \in \omega_2, \quad \textbf{(5.3)}$$

where η is the learning rate of the algorithm which moderates the extent to which weights are changed. We set η to a small value of 0.02. Thus, N training examples consisting of normal and attack instances are presented to the algorithm iteratively. The algorithm is repeated until all the examples are correctly classified. It is important to note that a perceptron can be applied for linearly separable data points only.

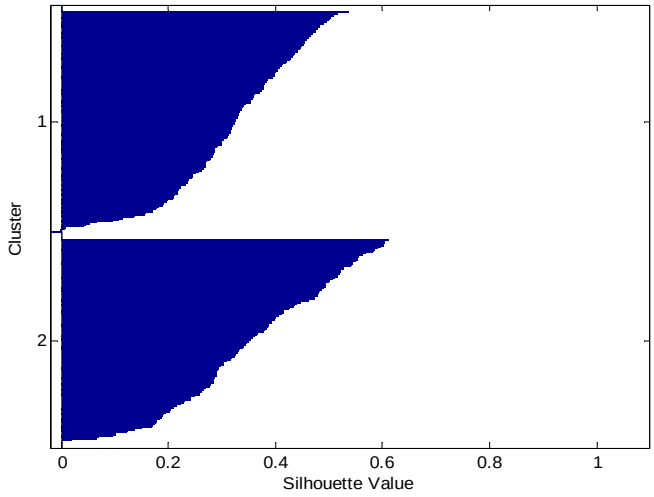


Figure 5.6: Linearly separable training data

To highlight the fact that the attack instances and the normal instances are linearly separable in the space of detection metrics, we tested the data by performing k-means clustering test [69]. We see from the silhouette plot (MATLAB) in Figure 5.6, that our training sample is perfectly classified into two clusters (attack and non-attack instances).

5.5 Performance Analysis

We next study the performance of our proposed perceptron based IDS using simulations performed in ns-2 [15]. We use the same scenario and attack configuration as described in Section 5.3. Each simulation is run for 600 secs. We generate a sample size of 33,500 feature points, comprising of 50 % of attack instances and 50 % of normal instances. The training period is set for 500 epochs so that the weights of the perceptron model converge for sure. We use about 50 % of the generated data for our training model and the other half is used later for testing and validation purposes. We evaluate the network performance based on the following metrics:

- True positives (TP): Number of times an alert is raised, when an attack is present.
- False negatives (FN): Number of times when no alert is raised, but attack is present.

- False positives (FP): Number of times when alert is raised, but attack is not present.
- True negatives (TN): Number of times when no alert is raised, when no attack is present.

The performance of our classification algorithm is thus based on TPR (True positive rate) and FPR (False positive rate). TPR = $\frac{TP}{TP + FN}$, which is the ratio of number of alerts when there is an attack to the total number of attacks. Similarly, FPR = $\frac{FP}{TN + FP}$.

We illustrate the detection accuracy (TPR) of the trained perceptron model for various sizes of training sample in Figure 5.7. We find that the accuracy of the model is smaller for just few sample size. The model achieves best results for a training size of 4000-8000 instances. We next study the Receiver Operating Characteristics (ROC) curve (TPR vs. FPR) which reflects the tradeoffs in the sensitivity of the detection algorithm. Figure 5.8 shows the ROC curve for our detection scheme. We observe that in our scheme very less number of normal instances are misclassified as anomalies (as seen by 0 FPR value) and all attack instances are correctly identified as intrusions (as seen by the high TPR value close to 1).

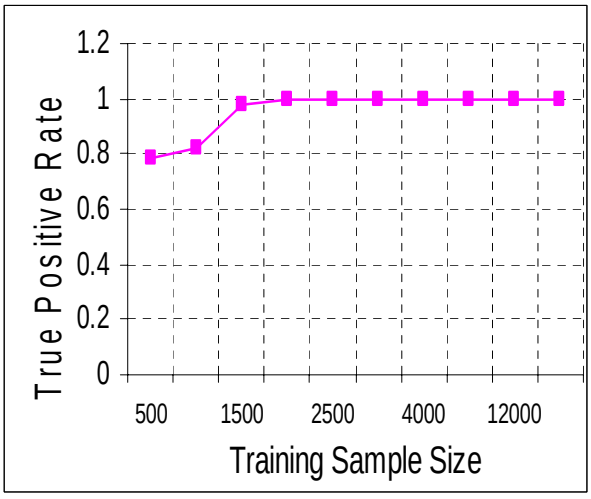


Figure 5.7: Effect of training sample on TPR

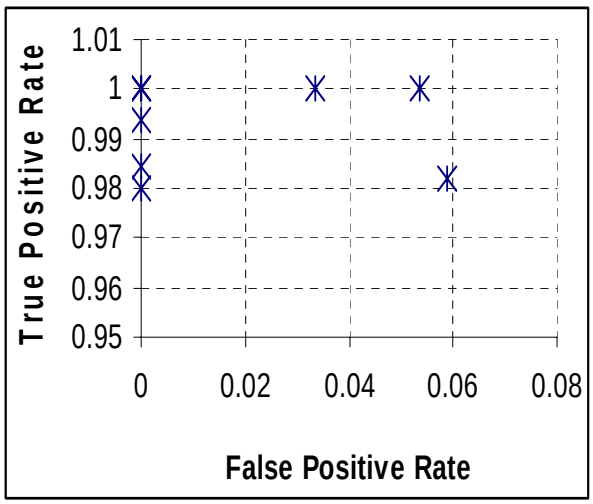


Figure 5.8: ROC Curve

Table 5.2 summerizes the misclassification rate of our algorithm for different generation rate of RREQs by a single malicious MR. Misclassification rate is defined as the percentage of inputs misclassified during one training epoch (include FP and FN).

Table 5.2: Misclassification rates of perceptron

Bogus RREQ generation rate	Misclassification rate
1 RREQ/sec	0.9811 %
5 RREQ/sec	0.9881 %
10 RREQ/sec	0.9995 %

Figure 5.9 demonstrates the detection ability of our system when the number of malicious MRs generating malicious requests is increased. Even if only a small percentage of the MRs are malicious, we see that our perceptron based IDS accurately detects them with a high TPR of 98% and has close to 100% detection rate for largely compromised network. This is because for a high attack rate, the data is easily separable in the space of detection metrics and is thus easily classifiable by the perceptron model. Similarly, Figure 5.10 shows the FPR for varying number of good MRs. A large number of good MRs imply very few MRs are compromised. It can be seen that the maximum value of FPR is within 23%, for a largely compromised WMN. We also see that as the attack rate increases, the FPR increases indicating that a small percentage of false alarms would be raised from time-to-time.

Both graphs prove that the proposed perceptron based IDS has a very high TPR (100%) and a low FPR (23%). The proposed IDS, thus detects malicious route discoveries accurately and efficiently. It also illustrates the fact that the features (detection metrics) selected as inputs for training the perceptron model, are accurate predictors of the attack.

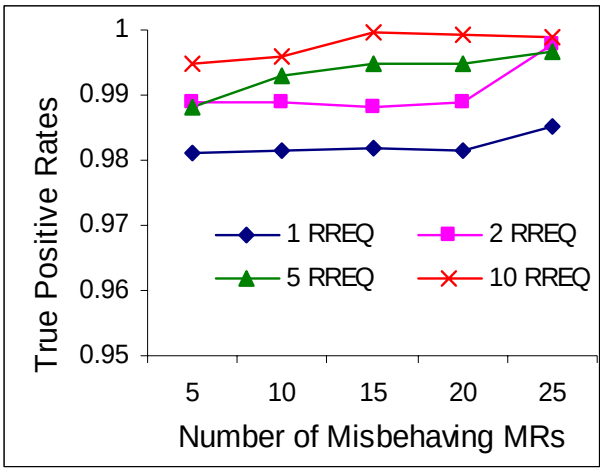


Figure 5.9: TPR under varying number of misbehaving MRs

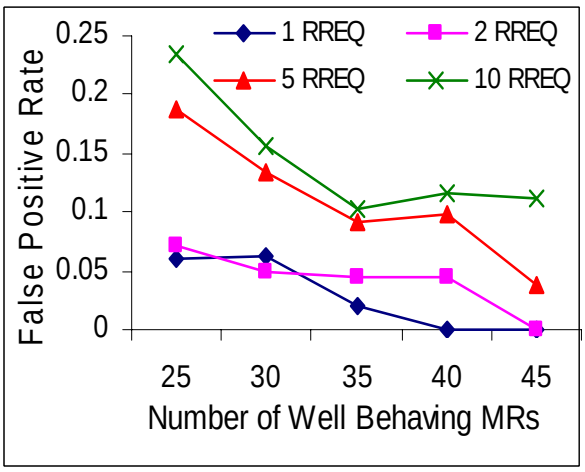


Figure 5.10: FPR under varying number of well behaving MRs

We next study the effect of learning rate (η) on the detection rate of our proposed IDS. It is important to choose a small value for the perceptron to correctly classify the data with a low FPR and a high TPR. We see from Figure 5.11, that a lower learning rate has a higher detection rate over higher learning rate. Hence, we choose an ideal learning rate of 0.02 for our model.

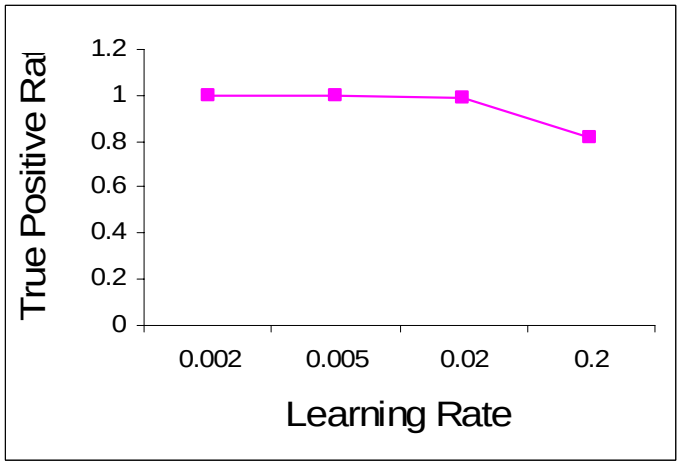


Figure 5.11: Effect of learning threshold on the detection rate

5.6 Conclusion

In this chapter, we proposed an intelligent machine learning technique based on linear perceptron classifier to detect malicious route discoveries. We model the detection of malicious route flood as a classification problem. As malicious route flood attack has a global effect at every MR in the WMN, the IDS raises a timely alert on the occurrence of the attack. We prove through extensive simulations, that our perceptron based IDS model has a high detection rate and a low false positive rate. As a part of our future work, we plan to use other machine learning techniques like multi-layer perceptron model to detect attacks like selfish node and blackhole attack, which fail to get classified by a linear perceptron model.

Chapter

6. Secure and Fast Authentication using Merkle Trees

6.1 Introduction

With rapid proliferation of WMNs, it is of great concern as to how to incorporate security mechanisms so as to prevent fraudulent users [70]. The open wireless communication channel and the multihop nature of WMNs can pave way to malicious intruders. Such an intruder, if unnoticed, can disrupt the network activity by conducting various attacks such as impersonation attack, anti-integrity attack, anti-confidentiality attack etc. [71]. An adversary can conduct impersonation attack by introducing a rogue MR that sends forged/replayed registration messages to entice mesh clients (MCs) who assume that it is connected to the Internet. A rogue MR can conduct anti-integrity attack by poisoning the route tables. It can cause other MRs to redirect their traffic towards itself by advertising a higher rate link/less congested link to the IGW. On the other hand, a malicious MC can conduct anti-confidentiality attack by revealing

sensitive cryptographic keys obtained by eavesdropping. It is important to ensure that a good MC does not join a rogue MR. Similarly, MR should prevent a malicious MC from freely enjoying the network resources and should be able to bill the user for utilizing the network services. To summarize, the above attack raises several concerns as outlined below:

- How to enforce mutual authentication between a MC and the MR to which it wants to connect?
- How to establish trust relationships amongst multi-hop MRs?
- How to uniquely identify a MC and prevent it from changing its identity to conduct different types of attacks?

In this chapter, we propose a hierarchical security model for WMNs that addresses these concerns and facilitates fast and secure authentication of MCs. We focus on a network layer authentication mechanism so that it is independent of the underlying network topology. Owing to its hierarchical structure, the task of provisioning security in a WMN is two-dimensional. As the MRs form the wireless backbone, highest level security is required here. As a first stage of authentication, it is important to establish a security association between a newly joining MR and the IGW. We describe how the IEEE 802.11s standard [72] achieves this. A second stage of authentication involves establishing a security association between the MR and the MC (single/multi-hop). The final stage of authentication involves verification of the integrity of the control messages exchanged between a MR and a MC. We achieve the last two requirements using a low cost hash tree based scheme called MT-MAP (Merkle Tree based Mesh Authentication Protocol). We employ Merkle Tree [73] based authentication certificates in which a user commits to the root of the Merkle Tree and authenticates by presenting proofs to regenerate the root value. It is a well known fact that the hash based algorithms (MD5, SHA-2)

are computationally inexpensive than the symmetric key algorithms like (DES, 3DES, AES) which is in turn, are inexpensive than the public key algorithms (RSA) [74]. Thus, in our design, we avoid the use of public key certificate on the resource constrained wireless MCs [75]. We also outline a methodology to refresh the authentication tokens in a timely manner. The following are the key contributions of our research:

- We provide an insight into the problem of authentication in WMNs. To the best of our knowledge, this is the first comprehensive investigation that outlines various existing research efforts to incorporate authentication mechanism in a typical WMN architecture.
- We investigate efficient alternatives to replace public key certificate based authentication and outline a low cost authentication mechanism to validate the identity of MCs.
- The proposed MT-MAP facilitates authentication of all MCs connected to the MR by single/multiple hop.
- We provide a detailed overview of IEEE 802.11s security proposal and show how the proposed mechanism tightly adheres to the hierarchical security paradigm of IEEE 802.11s.
- We present an integrated authentication suite to generate, refresh, and validate the user certificates in a systematic fashion.
- We discuss the resiliency of the proposed authentication mechanism in detail against various possible security attacks such as impersonation, anti-integrity attack and replay attacks.

The rest of the chapter is organized as follows. We analyze various existing research proposals for authentication in WMNs in Section 6.2. In Section 6.3, we discuss at length the hierarchical security paradigm of IEEE 802.11s standard and provide the fundamental security goals of this standard. Section 6.4 provides an overview of the proposed MT-MAP in detail.

Section 6.5 shows the security analysis of the proposed MT-MAP scheme. Finally, Section 6.6 concludes with a brief summary and future work.

6.2 Approaches in Literature

The problem of secure authentication is a well known one in Wireless Local Area Networks (WLANs) and Ad hoc networks. The growing interest in WLANs, prompted the IEEE Task Group to define 802.1X [76] framework that restricts network access to authorized client only by three party authentication. Thus, a simple way of securing access to MRs is by adapting 802.1X to WMNs. Before, we dwell into the details of the protocol, we introduce some basic terminologies:

- Supplicant: the MC that wishes to join the network
- Authenticator: the MC seeking network services from a directly connected AP
- Authentication server (AS): The backend central server which acts as AAA (Authentication, Authorization and Accounting) server and maintains all user credentials like secret keys, public key certificates, and passwords.

The 802.1X framework offers port based network access control at Layer 2. When a MC attempts to connect to an AP, it is initially restricted to send authentication traffic only. It sends authentication request messages through an unauthorized port in the AP. The AP acts as a proxy between the supplicant and the backend AS, and relays the authentication requests to the AS. The AS and the client mutually authenticate each other and generate a secret Pairwise Master Key (PMK).

The IEEE 802.1X framework employs Extensible Authentication Protocol (EAP) [77] to execute the client authentication, using various cryptographic suites such as public key certificate based (EAP-TLS: EAP Transport Layer Security [78]), secret key based (EAP-TTLS: EAP

Tunneled Transport Layer Security [79]), one-time password ased (EAP-MS-CHAP: EAP Microsoft Challenge-Handshake Authentication Protocol), smartcards, etc. EAP offers a great degree of flexibility by running any authentication method on top of the IEEE 802.1X framework. Thus, the client and the AS mutually authenticate each other and compute a session key called PMK. We have so far explained only the client authentication. The AS authenticates the AP using Remote Authentication Dial in User Service (RADIUS) protocol [80] and establishes a Security Association (SA) between the two. The logical secure channel between two communicating entities is known as SA. Unlike EAP, RADIUS provides per-packet authentication and integrity. The AS delivers previously generated PMK to the AP. Thus, the MC and the AP mutually authenticate and derive a transitive trust through a common trusted friend which is the AS [81]. The IEEE 802.11i standard in conjunction with the IEEE 802.1X framework further defines how the PMK generated through EAP exchange can be used to derive transient keys (PTK- Primary Transient Keys) for the communication between the MC and the AP [82].

However, one of the principle hindering factors in extending the IEEE 802.1X framework to WMNs is the multihop nature of MCs [83] [84]. The IEEE 802.1X standard is operated at the link layer to establish a point to point link and upon successful authentication the MC and the AP register each other's Medium Access Control (MAC) address. This mandates that the MC should be directly connected to the MR. Thus, the IEEE 802.1X enables only single hop authentication of MCs. Unfortunately, there is no standard protocol to authenticate mobile clients in a multi hop network. Realizing the deficits of a link layer authentication, IETF Task Force manifested a solution to authenticate clients using IP protocols called PANA (Protocol for carrying Authentication for Network Access) [85]. However, PANA cannot provide link layer security

and requires the use of protocols such as IPSec. Also, PANA does not define a new authentication protocol and the incorporation of EAP messages inside an IP protocol is left unaddressed. Maknavicius et al. [83] solve the problem of authenticating multihop MCs by using PANA with EAP-TLS [78] (which is a based on public key certificate). But, the use of public key cryptography incurs considerable overhead in signature verification and is an expensive operation on the resource constrained MCs. Cheng et al. [84] also propose a network layer authentication mechanism that reduces this overhead by adopting EAP-TTLS authentication [79]. The primary advantage of EAP-TTLS is that the MC is not required to carry certificates. The TLS server uses its public key certificates to authenticate itself to the MC using EAP TLS. And the secret/passwords of the MCs is encapsulated inside the EAP messages and tunneled to the TTLS server. However, certain amount of overhead is still incurred in the establishment of a TLS tunnel. Also, it would require vendors to support tunneling and TLS capability.

Zhang et al. [86] propose a secure authentication architecture that uses ID based cryptography, thereby reducing the size of the certificate to few bytes. They proposed a hierarchical security architecture in which MCs are registered to a broker and MRs are registered to a WMN operator. The MC obtains an authentication token called UPASS from the broker which authorizes it to connect to MRs. As the brokers and the WMN operators share security association, the MC and the MR mutually authenticate. However, primary disadvantage of this scheme is that the user obtains its UPASS private key through the WMN operator which is a third party.

6.3 Overview of IEE802.11s

An increasing commercial interest in WMNs has prompted the IETF to setup a new task group called IEEE 802.11s [72] to formalize the Physical, MAC layer and other security issues.

It is expected to be released by 2008. In this section, we primarily focus on the security addendum to this standard. As a WMN is envisioned for a large community of user, scalability of the security architecture is very important. The IEEE 802.11s standard proposes hierarchical security architecture for authentication and key distribution management. More precisely, 802.11s extends the 802.11 EAP security proposals by establishing a key hierarchy [72]. It should, however be noted that 802.11s offers link level security of MRs and not end-to-end security of the mesh path.

A typical WMN trust hierarchy consists of a backend server, AS at the top level, a key distributor, MKD at the next lower level, and new MRs joining the network at the lowest level as shown in Figure 6.1. The role of central AAA server is delegated to a set of Mesh Key Distributors (MKD) in the network, which are special MRs or IGW with secure physical link to a backend Authentication Server (AS) for AAA services. Initially, the trust hierarchy is established by authenticating MKD to the AS, using the IEEE 802.1X authentication framework. Henceforth, the authentication henceforth progresses in an orderly fashion, starting at the MKD. A new MR that joins the network becomes a supplicant and the MKD acts as an authenticator. The supplicant MR and the MKD mutually authenticate using the AS. Once, this MR is authenticated, it assumes the role of authenticator called Mesh Authenticator (MA). The MA is authorized to authenticate any other supplicant MR through a series of 802.1X authentication to the MKD and subsequently to the AS. Thus, the key distribution progresses in a hierarchical fashion as shown in Figure 6.1. Though the 802.11s standards achieves establishment of a secure mesh backbone, the authentication of MCs is left as an open issue. Here, we focus on the design of a secure authentication scheme for the MCs.

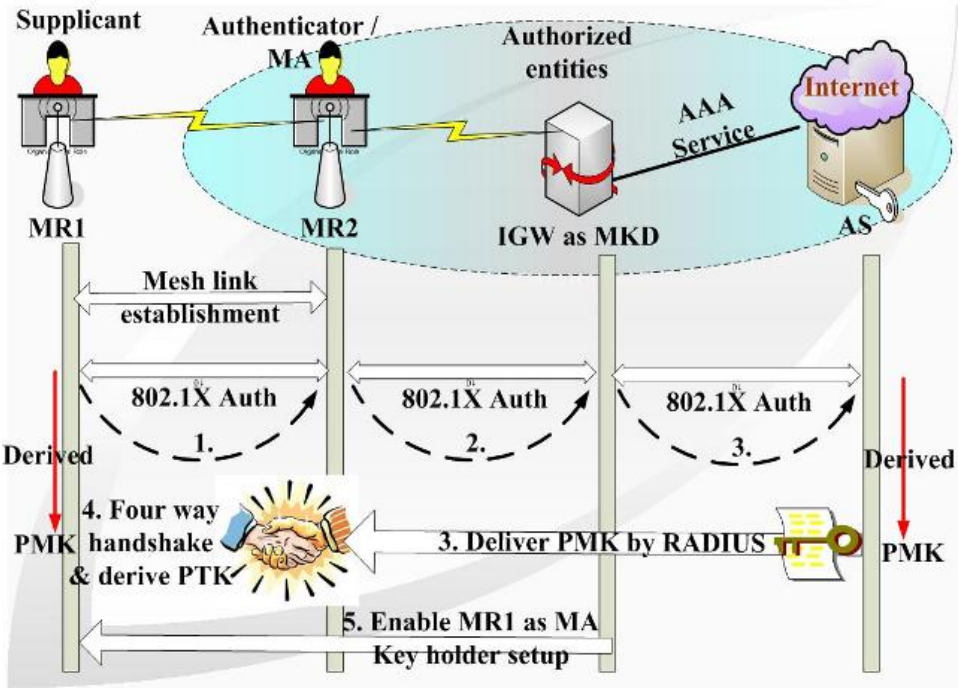


Figure 6.1: Hierarchical Authentication of Mesh Points

6.4 Proposed Merkle Tree Based Mesh Authentication Protocol

In this section, we give a brief summary of the design goals of the proposed authentication protocol and then introduce the concept of Merkle Trees construction. Finally, we outline the architecture of our proposed MT-MAP protocol.

6.4.1 Design Goals of MT-MAP

We propose a light weight hash tree based authentication scheme to validate the identity of a MCs and prevent fraudulent Internet access. The following summarizes the design goals of the proposed protocol:

- Authentication of multihop/single MC: We design an authentication mechanism that is independent of the underlying network topology and enables authentication of MCs connected by single/multiple hops.

- Auto-refresh of authentication tokens: We design a secure mechanism based on one-way hash chains to update the authentication tokens in an efficient and timely manner.
- Authentication of control messages: We design an authentication protocol that also addresses the issue of maintaining the integrity of control messages such as route update, disassociation request to prevent Denial of Service (DoS) attacks.
- Minimal use of Public Key Infrastructure (PKI): As PKI incurs considerable overhead; we intend to employ it only for the authentication of the MRs and do not load the MCs with bulky public key certificates. Instead, we authenticate MCs using low cost hash trees called Merkle trees.

6.4.2 MT-MAP Approach

As MC can connect to WMN in a multihop fashion, we implement the MT-MAP in the network layer. A public key certificate is much bulkier than a hash digest. Also, it incurs considerable communication and computational overhead in the verification and refreshing of the certificate. Hence, we incorporate low cost hash trees called Merkle trees to authenticate the MCs. We integrate the proposed approach with the hierarchical security framework of IEEE 802.11s. In addition to the entities defined in the IEEE 802.11s standard, we use additional entities in our proposed MT-MAP protocol. For clarity of our proposed protocol, we briefly describe the security functionality of these entities as follows:

- Mesh Client (MC): The MC is a wireless device like laptop, PDA seeking Internet access through the MR/MA We henceforth refer MR as Mesh Authenticator (MA) and use it interchangeably. The MC acts as a supplicant and in order to authenticate itself to the attached MA, it presents a secret token and authentication proof (together used for computing the root of Merkle tree).

- Mesh Access Restrictor (MAR): The MAR is an agent built into the MR to restrict illegitimate MCs from accessing the network. It is similar in operation to the 802.1X control ports. Until the MC is authenticated, it allows authentication traffic only and blocks all data traffic.
- Mesh Authentication Agent (MAA): The MAA is an integral part of the mesh backbone management center called Mesh Key Distributors (MKD). The MAA is an agent that resides inside the MKD and is responsible for authenticating new MCs joining the network.
- Mesh Authentication Server (MAS): MAS is the backend Authentication Server (AS) that delivers the credentials of MCs to the querying MAA. We assume the existence of a secure physical link between the MAS and MAA.

It should be noted that in our authentication framework the above entities (MAA, MAS, MAR) are logical entities and hence may or may not coexist in the same MR. We integrate the above logical entities into the IEEE 802.11s security hierarchy in order to utilize the trust hierarchy defined by the standard. Figure 6.2 illustrates the logical location of the entities.

The authentication process involves three phases as shown in Figure 6.2. A newly joining MC first initiates Mesh Authenticator Discovery phase in order to establish an association with the MA and procures a temporary IP address from a local DHCP server for the purpose of sending authentication messages. In the second phase, the newly joining MC authenticates the MA and at the same time MA authenticates the MC. The MAA delivers the result of the authentication to the MC and assigns a dynamic IP address to it.

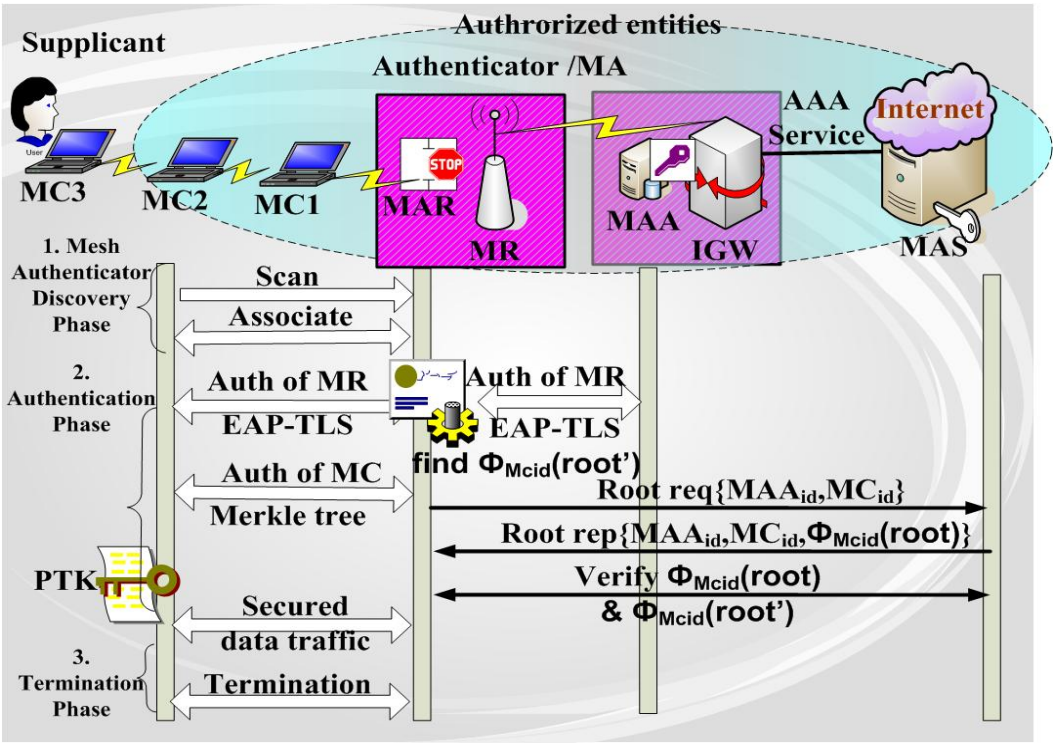


Figure 6.2: Phases of MT-MAP protocol

But, if the authentication fails or the MC moves away from this MKD domain, the termination phase is initiated to de-authenticate the MC.

A. Mesh Authenticator Discovery Phase

A newly joining MC scans through the advertised list of MA and request for an association with a suitable MAs. The MA advertises its presence by sending periodic beacons and accepts incoming association request from the MC. MC that is outside the coverage range of MA, relays its association request to other MC. Other MC that know the path to the MA, respond to this newly joining MC and relays its request in a multihop path towards the MA. As shown in Figure 6.2, MC3 connects to the MA by relaying it request through MC2 and MC1. As it is important to validate the identity of each intermediate MC forwarding the request, we assume an existence of secure multi-path routing discovery protocol [71]. Such a scheme utilizes prior security association available at the MA to validate the intermediate MCs.

B. Authentication Phase

In this phase of the MT-MAP protocol, an EAP-TLS scheme is used to authenticate the MA and a hash based authentication scheme is used to authenticate the newly joining MC.

We adopt a hash tree based approach to authenticate the MC because it is more secure than one time password and quicker than public key certificates. The Merkle trees offer a powerful way of binding a set of secret tokens to a single public root value by applying one-way hash functions. The secret tokens are stored in the trusted components of MC. Due to the one-way nature of hash function; it is computationally infeasible to compute the secret tokens from the published root values. The MC releases one secret token at a time and presents it to the MA, which further securely delivers it to the MAA. The MC also sends along with the secret token some authentication proofs, which is used by MAA to compute a root value. It validates MC by comparing this computed root value against the published root value retrieved from the MAS. In order to have mutual authentication, the MC also validates the accessing network. MC tunnels a router authentication request to the MAS through the MA to validate the identity of the MA. Now, MC trusts MAS (SA3), MAS trusts MAA (SA1), and MAA inturn trusts MA (SA2). Thus, MC trusts MA in a transitive manner (SA4). Figure 6.3 illustrates this transitive nature of trust derived between the MC and MA. As we are primarily focused on authentication, we refrain from the discussions on computing data encryption keys. We assume protocols like IEEE 802.11i handshake can be used here to compute link layer encryption key between MA and MC.

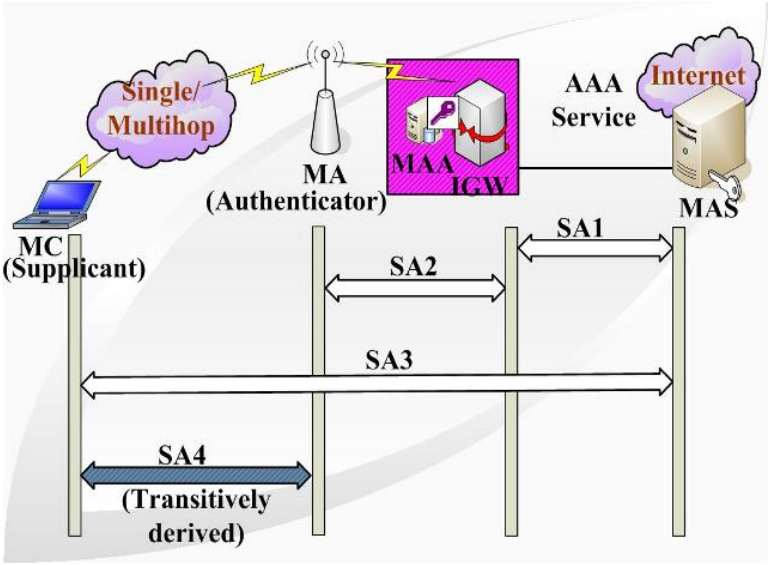


Figure 6.3: Establishment of Transitive Security Association

In our protocol, we employ Merkle hash trees that serve as a varied range of cryptographic applications, including authentication [87] [88]. Merkle tree is characterized by a hash function $\hbar$ (like MD5, SHA-2) and a mapping function, φ used to map each node to a fixed size string. Applying these functions on a set of secret tokens, it then generates a complete binary tree structure. The leaves of the Merkle tree, consists of a set of m ($=2^H$) randomly generated values, where H is height of the Merkle tree. These leaves serve as multiple private keys and the root of the tree is a known public key. In order to ensure the secrecy of other leaf values, the φ values of the leaf nodes are obtained from a pre-image ($leaf_i'$) such that $\varphi(leaf_i) = \hbar(leaf_i')$. MC then performs a Merkle tree construction to commit to the set of secret tokens [73]. For each internal node, n_p, the φ value of the node is obtained by applying a hash function on its left (n_l) and right child (n_r) as shown below

$$\varphi(n_p) = \hbar(\varphi(n_l) \,||\, \varphi(n_r)), \text{ where } || \text{ represents the concatenation of two string.}$$

Thus, in this manner MC recursively applies a hash function and generates a root value $\varphi(root)$ as shown in Figure 6.4. We assume that the MC pre-registers this value with the

MAS [87] and the value of the root is publicly known. When a MC joins the network, it authenticates itself by the following steps:

a. The MC discloses to the MA the pre-images of the leaf along with an authentication path. The authentication path denotes all the φ values from the tree that are essential to verify the path from that released leaf value to the root of the tree. For example, as shown in Figure 6.4, when a MC tries to authenticate itself with a secret token, $leaf_1'$, the authentication path consists of the following values $<\varphi(leaf_2), \varphi(n_{34}), \varphi(n_{58})>$.

b. The MA acting as a verifier computes

$$\varphi(root') = \hbar(\hbar(\varphi(n_{01}) || \hbar(\hbar(leaf_1')) || \varphi(leaf_2))) || \varphi(n_{58})).$$

c. The MA then submits a Root Request (Root-Req) to MAA which forwards it to the MAS in order to obtain the published value of the root, $\varphi(root)$ of the associated MC.

$$MAA \rightarrow MAS \quad Root\text{-}Req\{MAA_{id}, MC_{id}\},$$

where MAA_{id} and MC_{id} are the unique ID for the MAA and the MC respectively.

d. The MAS responds with Root Reply (Root-Rep) message by performing a lookup in its database and sends a reply to the MAA which inturn forwards to MA.

$$MAS \rightarrow MAA \; Root\text{-}Rep\{MAA_{id}, MC_{id}, \varphi_{MCid}(root)\}.$$

The authenticity of $\varphi_{MCid}(root)$ is guaranteed by a pre-existing security association between MAA and MAS (SA1) and between MAA and MA (SA2).

e. The MA next compares the published root against the computed root $\varphi(root')$ to see if the two are equal.

(i) If equal, the MC is authenticated and MAR opens the port for sending data traffic.

(ii) Else, the connection with the illegitimate MC is terminated. The MAR closes the port for further traffic and initiates termination phase.

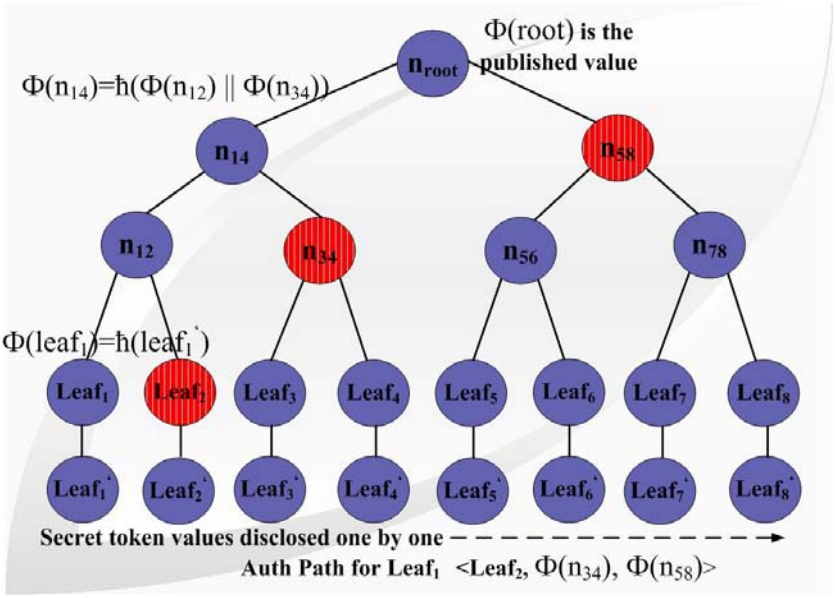


Figure 6.4: Merkle Tree Construction

Thus, hash trees offer a flexible alternative to the bulky public certificates. As only the secret tokens are stored in the MC, it is ideal for mobile devices such as PDA with limited storage. Also, efficient Merkle Tree Traversal techniques can be employed to generate authentication paths on a need basis, instead of storing the entire tree in the MC [89]. A typical Merkle tree traversal would require log (*m*) hash operations to reach up to the root. Hash chains also offer a flexible way of performing authentication [74] by applying a hash function repeatedly on a secret nonce (N_0) and using the hash chain in the reverse order (N_i, N_{i-1}, N_{i-2}, …, N_0). But, hash chain based authentication, incurs considerable computational overhead in the recursive application of the hash function. To validate a value, say N_{i-1}, it requires *i-1* hash operations.

6.4.3 Authentication of Control Messages

Every MA maintains a binding for all authenticated clients registered under it. The MA maintains a client cache table which consists of the IP address of the local MC and the IP address

of the peer entity (IGW/another remote MC) with which it is communicating [90]. When a MC moves out of the range of the current MA, it sends a termination message to end its current association. The MC can also change its location in the multihop path leading to the MA. Thus, the entry in the client cache table of MA needs to be updated to reflect these changes. These are important control message and if unauthenticated, it could create a potential loop hole. A malicious attacker can generate spurious control message, imitating the genuine MC and initiate a false disconnection and/or false binding in the client cache table at the MA. This could lead to a potential DoS attack [71]. Though, we can secure the control messages by encrypting it, repeated use of the same key results in inadequate security. Hence, we propose to use previously outlined Merkle tree construction for authentication of control messages between the MCs and the MA.

As a Merkle tree facilitates multiple one time authentication paths to validate the publicly known root, we propose to use the multiple secret leaf tokens for authenticating the control messages. The leaves of the merkle tree and their associated authentication paths are revealed in a pre-determined order, from left to right and sent along with the control packet. In order to reduce the communication overhead in the authentication of control packets, we move the published root of the client, φ_{MCid} (root), from the MAS to the MA using the pre-existing security key. Thus, the MA acts a verifier and caches the public root in its *Authentication Cache* (AC). When an authenticated MC sends a control packet such as route update, termination packet, etc., it presents the next unused secret token (leaf_i) and the corresponding authentication proofs for it (Auth_i).

$$\text{MC} \rightarrow \text{MA} \{\text{MA}, \text{MC}_{\text{id}}, \text{leaf}_i, \text{Auth}_i\}.$$

Thus, the MC is authenticated by the MA using the same public root key. The MA also caches in AC all leafs released so far and the time of their release to avoid a replay attack using an expired leaf value. Thus, if at time $T_{K,}$ the last released leaf value is $leaf_i$, for all time $T > T_K$, the MC should disclose a new leaf value $leaf_j$ to authenticate itself such that $j > i$.

6.4.4 Auto-refresh of Hash Tree based Certificates

The primary advantage of employing hash tree based authentication is that it facilitates easy refresh of secret tokens and is scalable to a large networks [75]. On the other hand, public certificates apart from being bulky; are extremely tedious to revoke. The Certificate Authority (CA) should perform the onerous task of tracking the validity of the certificates of each client and should notify all entities using this certificate about its expiry. Revocation of certificates is typically performed by sending a large Certificate Revocation List (CRL) [91] to the MCs, which incurs considerable overhead.

We propose an auto-refresh mechanism that is performed with the help of MA. The MA maintains a counter in the AC that is incremented for each leaf value presented to it. And when the counter value reaches its limit ($m=2^H$), it implies the MC has exhausted set of all leaf values and the cached public root of the MC automatically expires. An expiration notification is sent to the MC to re-construct a new Merkle tree. The MC generates a new set of pseudo random secret tokens and re-constructs the root of Merkle tree as explained earlier. It then securely registers the newly computed root with the MA using the pre-negotiated session key (IEEE 802.11i PTK). The MA inturn registers the newly computed root of MC with the MAS. As the number of control messages is considerably lower than data messages, the re-initialization of Merkle tree occurs less frequently. It should be noted that only a previously authenticated MC performs re-initialization upon the expiration of its authentication tokens.

6.5 Security Analysis

In this section, we investigate various possible security attacks on the proposed authentication protocol and outline the counter measures that are taken into consideration.

6.5.1 Protection against Denial of Service (DoS) Attack

A malicious attacker can conduct a DoS attack by sending a flood of packets to the MR. This results in denial of service to legitimate MCs. Another well known security flaw that can be exploited by an attacker is that the de-authentication/termination message from the 802.11 based wireless clients is unauthenticated [71] [84]. Thus, a malicious attacker can send false termination messages on behalf of a legitimate client. Our protocol averts such dangerous situations, by authenticating all control messages exchanged between the MC and its peer communication entity. A malicious attacker can also conduct DoS attack by repeatedly submitting false credentials. Such an attack can be prevented by appending a cookie generated from IP address of MC to the authentication request. The MAR at the router keeps an upper bound on number authentication failures associated with a MC and blocks all further traffic.

6.5.2 Protection against Replay Attack

In a replay attack, a malicious attacker caches a legitimate authentication request (by eavesdropping or interception) and replays it at a later time. Thus, if a MA fails to recognize the authenticity of the request, it could be granting access to malicious attackers. Our scheme prevents such an attack as MC is required to present fresh token every time. If a malicious attacker replays an old secret token, it is immediately identified by the MA. When a MC presents a token, the MA checks the AC to determination its freshness. Thus, without implementing a complex sequence number mechanism for each token, we exploit the one-time validity of Merkle tree tokens to detect such an attack.

6.5.3 Protection against Spoofing Attack

IP address spoofing is a rampant problem in WMNs due to multihop and dynamic nature of communication between the MR and MCs. Spoofing is the act of forging a false IP address in the packets originating from an attacker. Using such a spoofed address, a malicious attacker attempts to disrupt network activities. It can perform a man-in-the-middle attack by cleverly intercepting a MT-MAP termination request and thus hijacks the session. The proposed protocol prevents such an attack as it is impossible to derive the link layer security keys. Also, the MA maintains in its AC, a binding of the IP address of the MC and its associated authentication tokens. Thus, it cannot replay an old token and authenticate the control messages. Also, it is infeasible to present the right authentication paths for any other secret token that it intercepts.

6.6 Conclusion

Security of WMNs is very much in its infancy, impeding their wide spread deployment. We have proposed an efficient network layer mechanism to mutually authenticate a multihop MCs and the MR. We employ expensive public key certificates to authenticate MRs and authenticate the MCs using light weight Merkle trees. A Merkle tree is a low cost tree built by the repeated application of a hash function on a set of secret tokens of a client. The client commits to a root value and publishes it in the authentication server. A client wishing to authenticate itself releases a secret token and an authentication path, which are then used to compute the root of the tree. If this is similar to the published root, the client is authenticated. The other secret tokens are used to authenticate all control messages between the client and the MR. We also outlined a mechanism to perform timely refresh of the authentication tokens. In the future, we plan to extend this framework for conducting secure and fast handoff of MCs.

Chapter

7. Conclusion and Future Research

As seen by the surge in its popularity, WMNs have rapidly evolved within a short span of time, surpassing other well known peer technologies. There are several research projects underway in the academia and industry to develop wireless mesh architecture. Increasing commercial interest in WMNs has prompted the IEEE to set up a task group (IEEE 802.11s) for formalizing the MAC and PHY standards. However, IEEE 802.11s is still an unapproved standard for WMNs that is targeted to be formalized by the end of 2008. Since its inception, it has become the limelight of all researchers. Nokia's Rooftop Mesh [92], Radiant Networks [93] , and MIT's roofnet [94] are some known efforts in this direction. Industrial giants like Motoroala, Intel, Nokia, Microsoft and other wireless mesh vendors like BelAir, Strix systems, Firetide, LocustWorld, Mesh Dynamics, Tropos, etc., have come up with several proprietary mesh architectures.

WMNs have emerged as a novel disrupting technology, with myriad applications in community internet access, public safety and public work operations. MRs can be deployed at strategic locations in the highway and can be combined along with wireless IP video camera to provide convenient and economical way of monitoring highways, without the need for laying expensive cables. WMNs mounted at strategic locations like buildings, poles, towers, rooftops, traffic signals throughout the city offer an easy way of video surveillance for firefighters, law enforcement personnel and other medical emergency operations so that real time road conditions and other vital statistics at the scene of accident can be retrieved quickly. Though the enlisted applications of WMNs seem luring, considerable degree of security measures have to be taken to provide a reliable Internet service. As WMNs are deployed in such a public environment, they can be easily tampered by an adversary who can modify/disable the routing functionality of the MRs.

In the earlier chapters, we have seen that the open and self-configurable architecture of WMNs makes them vulnerable to different attacks such DoS attack, route disruption attack, and selfish node attack. We have analyzed these attacks that degrade the performance of the overall network and proposed solutions to prevent infiltration of such attacks in the network. We have proposed several schemes that should be incorporated in MR to monitor and safeguard the network. We have implemented a cache based traffic regulator at the MR to throttle the infiltration of DoS attack and quench the flood of packets. We have developed a trace back tool at each every MR to pro-actively record a signature of the packets forwarded by it, so that this information can be later utilized to reconstruct the attack path. In order to handle potential selfish behavior of MR, we deploy policing agents in the network that periodically collects traffic reports of adjacent MRs. We also study the selfish adversary model in a multi-channel WMN

that exhibits radio selfishness, channel riding apart from dropping other's packets and propose a scheme to detect these malicious behavior of a MR. Finally, we proposed a perceptron based IDS to detect malicious route discovery that attempts to exhaust precious network resources. The state of the network is modeled as a classification problem and is judged by an IDS that is trained by using a perceptron algorithm. Results indicate that our newly developed algorithms greatly improve the ability of the network in preventing these attacks in a timely manner and in avoiding any performance degradation. Finally, we have proposed a light weight authentication protocol of enabling secure authentication of mesh clients. We use inexpensive hash operations in the place of public cryptography for authenticating mesh clients and also outline a scheme to perform timely refresh of authentication tokens.

7.1 Future Work

In this dissertation, we have thoroughly investigated key security threats hindering the large scale deployment of WMNs. We have proposed several security mechanisms for handling these problems. We have also evaluated the integrated security architecture using extensive simulations. A promising direction for extending our dissertation is the implementation and evaluation of our proposed security mechanisms in an experimental/real time test bed that is currently being built in the CDMC (Center for Distributed and Mobile Computing) Lab.

Further work can be done to analyze the problem of DDoS attack in which attacker uses several low bandwidth traffic aggregates to bring down the victim. An algorithm to throttle this would necessitate a reactive traceback mechanism to infer the multiple attack paths and an attack regulator module to collectively apply rate limiting rules on each individual attack path.

As a part of the future work, we can extend the capabilities of the IDS proposed in Chapter 5, so as to detect various route disruption attacks like black hole attack, selfish node attack, worm

hole attack etc. A perceptron model that we proposed earlier is a linear classification tool and is limited to the detection of simple flooding attacks. But, devious attacks such as black hole attack, worm hole attack have a localized impact on the network and are not linearly classifiable. In order to detect such attacks, intelligent mobile agents like Honeypots [96] can be used to closely study the attacker's strategy. Honeypots have been largely used in literature to identify Buffer overflow attack, DoS attack. We can employ Honeypot as a feedback module to judge the authenticity of reply packets generated in the network.

Another interesting direction is designing a fast re-authentication mechanism at handoff. A mesh client using 802.11i exchanges considerable management frames to authenticate itself and establish an encryption key. If it were to re-authenticate itself at every visiting MR, this would incur substantial overhead and larger delays in the order of few hundred milliseconds. The proposed MT-MAP protocol in Chapter 6 can be potentially extended to conduct fast re-authentication for the free roaming of mesh clients.

Bibliography

[1] C. Cordeiro and D. P. Agrawal, *Ad hoc & sensor networks: theory and applications*,World scientific publishing, Spring 2006, ISBN No. 81-256-681-3; 81-256-682-1 (paper back).

[2] D. P. Agrawal and Q. A. Zeng, *Introduction to wireless and mobile systems*, Brooks/Cole (Thomson Publishing), 2003.

[3] N. B. Salem and J.-P. Hubaux, "Securing Wireless Mesh Networks," In IEEE Wireless Communication Magazine, 13(2), pp. 15-55, April 2005

[4] A. Zimmermann, "Wireless Mesh Networks: An Overview," Technical Report. Mobile Communication Group, Informatik Research Lab, 2006.

[5] R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop MANETs," In IEEE Communications Magazine, 43(3), pp. 123–131, March 2005.

[6] R. Poor and E. Corp, "Wireless Mesh Network," In Intelligent System – Wireless Magazine, 2003.

[7] N. Nandiraju, D. Nandiraju, L. Santhanam, B. He, J. Wang, and D. P. Agrawal, "Wireless Mesh Networks: Current Challenges and Future Directions of Web-in-the-sky," In IEEE Communication Magazine, 2007.

[8] P. Kyasanur and N. H. Vaidya, "Capacity of Multi-channel Wireless Networks: Impact of Number of Channels and Interfaces," In the Proc. of ACM Mobicom, August 2005.

[9] L. Santhanam, B. Xie, and D. P. Agrawal, "Selfishness in Mesh Networks: Wired multihop MANETs," In the IEEE Communication Magazine Special Issue on Ad Hoc and Sensor Networks, October 2007.

[10] L. Santhanam, N. Nandiraju, Y. Yoo, and D. P. Agrawal, "Distributed Self-policing Architecture for Fostering Node Cooperation in Wireless Mesh Networks," In the Proc. of the Personal Wireless Communication, Vol. 4217, pp. 147-158, 2006.

[11] L. Santhanam, D. Nandiraju, N. Nandiraju, and D. P. Agrawal, "Active Cache based Defense against DoS attack in Wireless Mesh Networks," In the Proc. of International Symposium on Wireless Pervasive Computing, 2007

[12] L. Santhanam, A. Kumar, and D. P. Agrawal, "Taxonomy of IP traceback," Journal of Information Assurance and Security, Vol. 1, pp. 79-94, 2006.

[13] L. Santhanam, D. Nandiraju, N. Nandiraju, and D. P. Agrawal, "Low cost Reliable Traceback based on MAC Address Identifier in Wireless Mesh Network," OPNET Workshop, August 2006.

[14] L. Santhanam, A. Mukherjee, R. Bhatnagar, and D. P. Agrawal, "A Perceptron based Intrusion Detection System for Detecting Malicious Route Floods in Wireless Mesh Networks," In the Proc. of Third International Conference on Wireless and Mobile Communications, 2007.

[15] Network Simulator (ns-2), http://www.isi.edu/nsnam/ns/index.html.

[16] OPNET Modeler v11.0, http://www.opnet.com

[17] CERT Advisory CA-2000-01, "Denial-of-Service Developments," CERT, 2000; www.cert.org/advisories/CA-2000-01.html.

[18] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," In Internet Eng. Task Force RFC 2827, May 2000; www.ietf.org/rfc/rfc2827.txt.

[19] K. Park and H. Lee, “On the Effectiveness of Route-based Packet Filtering for Distributed DoS Attack Prevention in Power-law Internets,” In the Proc. of ACM SIGCOMM, pp. 15-26, 2001 DoS.

[20] A. Yaar, A. Perrig, and D. Song, “Pi: A Path Identification Mechanism to Defend against DDoS Attacks,” In the Proc. of IEEE Symposium on Security and Privacy, 2003.

[21] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Practical Network Support for IP Traceback,” In the Proc. of ACM SIGCOM 2000, IEEE/ACM Trans. Networking, Vol. 9, No. 3, pp. 226-237, 2001.

[22] A. C. Snoeren, C. Patridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W.T. Strayer, “Hash-based IP traceback,” In the Journal of IEEE/ACM Trans. Networking, Vol. 10, No. 6, pp. 721-734, 2002.

[23] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” In the Proc. of IEEE/ACM Transactions on Networking, Vol. 1, No. 4, August 1993.

[24] D. Lin and R. Morris, “Dynamics of Random Early Detection,” In the Proc. of ACM SIGCOMM 1997.

[25] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhary, “Design Consideration for Supporting TCP with Per-flow Queuing,” In the Proc. of INFOCOMM 1998.

[26] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, “Controlling High Bandwidth Aggregates in the Network,” In Computer Communication Review 32(3), July 2002.

[27] A. Yaar, A. Perrig, and D. Song, “SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks,” In the Proc. of the IEEE Symposium on Security and Privacy, May 2004.

[28] N. Nandiraju, D. Nandiraju, L. Santhanam, and D. P. Agrawal, “A Cache based Traffic Regulator for Improving Performance in IEEE 802.11s based Mesh Networks,” In the Proc. of the RWC, 2006.

[29] J. Mirkovic, G. Prier, and P. Reiher, “Attacking DDOS at the Source,” In the Proc. of the 10th IEEE International Conference on Network Protocols, pp. 312-321, 2002.

[30] I. Stoica, S. Shenker, and H. Zhang, “Core-stateless Fair Queuing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks,” In the Proc. of SIFCOMM, 1998.

[31] J. Jung, B. Krishnamurthy, and M. Rabinovich, “Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites,” In WWW, May 2002.

[32] A. Ramanamathan, “WADeS: A Tool for Distributed Denial of Service Attack Detection,” Thesis, University of Texas, A &M, 2002.

[33] Y. Kim, and A. Helmy, “SWAT: Small World-based Attacker Traceback in Ad-hoc Networks,” In the Proc. of the IEEE/ACM Mobiquitous Conference, pp. 85-96, 2005

[34] S. C. Lee and C. Shields, “Tracing the Source of Network Attack: A Technical, Legal and Societal Problem,” In the Proc. of the IEEE Workshop on Information Assurance and Security, pp. 239–246, 2001.

[35] Y. Zhang and V. Paxson, “Detecting Stepping Stones,” In the Proc. of the 9th USENIX Security Symposium, pp. 171-184, 2000.

[36] V. Paxson, “An Analysis of using Reflectors for Distributed Denial-of-Service Attacks,” In the Proc. of the ACM Computer Communication Review, Vol. 31, No. 3, pp. 3-14, 2001.

[37] H. Burch and B. Cheswick, “Tracing Anonymous Packets to Their Approximate Source,” In the Proc. of the 14th Conference on Systems Administration, Usenix Association, pp. 313-322, 2000.

[38] T. Baba and S. Matsuda, “Tracing Network Attacks to their Sources,” In IEEE Internet Computing Magazine, Vol. 6, No. 3, pp. 20-26, 2002.

[39] V. L. L. Thing and H. C. J. Lee, “IP Traceback for Wireless Ad-hoc Network,” In the Proc. of the 60th IEEE Vehicular Technology Conference, 2004.

[40] M. Conti, E. Gregori, and G. Maselli, “Reliable and Efficient Forwarding in MANETs,” In Journal of MANETs, Vol. 4, pp. 398-415, 2006.

[41] B. Wang, S. Soltani, J.K. Shapiro, and P.-N. Tan, “Distributed Detection of Selfish Routing in Wireless Mesh Networks,” Technical Report MSU-CSE-06-19, University of Michigan, 2006.

[42] L. Buttyan and J.-P. Hubaux, “Enforcing Service Availability in Mobile Ad hoc WANs,” In the Proc. of the IEEE/ACM MobiHOC Workshop, 2000.

[43] S. Zhong, Y. Yang, and J. Chen, “Sprite: A Simple, Cheat Proof, Credit-based System for Mobile MANETs,” In the Proc. of IEEE INFOCOM, 2003.

[44] Y. Yoo, S. Ahn, and D. P. Agrawal, “A Credit-payment Scheme for Packet Forwarding Fairness in Mobile MANETs,” In the Proc. of IEEE ICC, 2005.

[45] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating Router Misbehavior in Mobile MANETs,” In the Proc. of Mobi-Com, 2000.

[46] S. Buchegger and J.-Y. L. Boudec, “Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes Fairness in Dynamic Ad hoc Networks,” In the Proc of MobiHOC., 2002.

[47] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, “Sustaining Cooperation in Multi-hop Wireless Networks,” In the Proc. of NSDI, 2005.

[48] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, “Cooperation in Wireless MANETs,” In the Proc. of IEEE INFOCOM, 2003.

[49] H.-Y. Wei and R. D. Gitlin, “Incentive Mechanism Design for Selfish Hybrid Wireless Relay Networks,” In the Proc. of Mobile Networks and Applications, Vol. 10, pp. 929-937, 2005.

[50] A. C. Cardenas, S. Radosavac, and J. S. Baras, “Detection and Prevention of MAC layer Misbehavior in MANETs,” In the Proc. of the 2nd ACM workshop of Security of Ad hoc and Sensor networks, pp. 17-22, 2004.

[51] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu, “A New Multi-channel MAC Protocol with On-demand Channel Assignment for Multi-hop Mobile Ad hoc Networks,” In the Proc of Int'l Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN), 2000.

[52] P. Michiardi and R. Molva, “Simulation-based Analysis of Security Exposures in Mobile MANETs,” In the Proc. of the European Wireless Conference, 2002.

[53] E. Huang, J. Crowcroft, and I. Wassell, “Rethinking Incentives for Mobile Ad hoc Networks,” In the Proc. of ACM SIGCOMM PINS, 2004.

[54] L. Venkatraman and D. P. Agrawal, “A Novel Authentication Scheme for Ad hoc Networks,” In the IEEE WCNC, Vol. 3, pp. 1268-1273, 2000.

[55] B. Xie, A. Kumar, D. P. Agrawal, and S. Srinivasan, “Secured Macro/micro-mobility Protocol for Multi-hop Cellular IP,” 2006, pp. 111–136.

[56] P. Ning and K. Sun, “How to misuse AODV: A Case Study of Insider Attacks against Mobile Ad hoc Routing Protocols,” In Ad Hoc Networks, 3(6), pp. 795-819, 2005.

[57] B. Bhargava, W. Wang, and Y. Lu, “On Vulnerability and Protection of Ad hoc On-demand Distance Vector Protocol,” In the Proc. of International Conference on Telecommunication, 2003.

[58] Y. Hu, D. B. Johnson, and A. Perrig, “SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks,” In Ad Hoc Networks, pp. 175-192, 2003.

[59] Y. Hu, A. Perrig, and D. B. Johnson, “Ariadne: A Secure On-demand Routing Protocol for Ad hoc Networks,” In the Proc. of ACM Mobicom, pp. 12-23, 2002.

[60] S.-Y. Ni, Y.–C. Tseng, Y. Shyan, and J.-P. Sheu, “The Broadcast Storm Problem in a Mobile Ad hoc Networks,” In the Proc. of IEEE MCN, pp. 152-162, 1999.

[61] S. Desilva and R. V. Boppana, “Mitigating Malicious Control Packet Floods in Ad hoc Networks,” In the Proc. of IEEE WCNC, Vol.4, pp. 2112-2117, 2005.

[62] A. Valdes and D. Anderson, “Statistical Methods for Computer Usage Anomaly Detection using NIDES,” Technical report, SRI International, January 1995.

[63] W. Lee, S. J. Stolfo, and K. W. Mok, “A Data Mining Framework for Building Intrusion Detection Models,” In the Proc. of IEEE Symposium of Security and Privacy, pp. 120-132,1999.

[64] J. Cannady, “Artificial Neural Networks for Misuse Detection,” In the Proc. NISSC, pp. 443–456, 1998.

[65] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: A Heirarchial network intrusion detection system using statistical preprocessing and Neural Network classification," In the Proc. of Workshop on Information Assurance and Security, pp. 85-90, 2001.

[66] A. K. Ghosh, J. Wanken, and F. Charron, "Detecting Anomalous and Unknown Intrusions against Programs," In the Proc. of Computer Security Applications Conference, pp. 259-267, 1998.

[67] C. Siaterlis and V. Maglaris, "Detecting Incoming and Outgoing DDoS attacks at the Edge Using a Single Set of Network Charactersitics," In the Proc. of 10th IEEE Intl. Symposium on Computers and Communication Systems, pp. 469-475, 2005.

[68] Y.-A. Huang and W. Lee, "A Cooperative Intrusion Detection System for Ad hoc Networks," In the Proc. of 1st ACM Workshop on Ad hoc and Sensor Networks, pp. 135-147, 2003.

[69] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd Edition, Academic Press, 1997.

[70] L. Santhanam, B. Xie, and D. P. Agrawal, "Selfishness in Mesh Networks: Wired Multi-hop MANETs," To Appear in IEEE Wireless Communication Magazine Special Issue on Security in Ad hoc Networks, 2007.

[71] B. Xie, and A. Kumar, D. P. Agrawal, and S. Srinivasan, "Securing Macro/micro mobility for Multi-hop Cellular IP," Elsevier Journal on Security in Wireless Mobile Computing, Volume 2, Issue 2, Pages 111-136, April, 2006.

[72] G. R. Hiertz, S. Z. Max, D. Rui, L. Berlemann, "Principles of IEEE 802.11s," In the Proc. of ICCCN, pp. 1002-1007, 2007.

[73] R. Merkle, “Protocols for Public Key Cryptosystems,” the IEEE Symposium on Research in Security and Privacy, 1980

[74] H. Tewari and D. O'Mahony, “Lightweight AAA for Cellular IP,” In the Proc. of European Wireless 2002, Florence, Italy, February 25-28, 2002,pp 301-306

[75] D. Berbecaru, “MBS-OCSP: An OCSP based Certificate Revocation System for Wireless Environments,” In the Proc. of the IEEE Signal Processing and Information Technology, 2004.

[76] IEEE Std., “Local and Metropolitan Area Networks Port based Network Access Control,” 2001.

[77] A. Aboba and L. Blunk, “Extensible Authentication Protocol (EAP),” RFC 3748, 2004.

[78] A. Aboba and D. Simon, “PPP EAP TLS Authentication Protocol,” *RFC 2716*, 1999.

[79] P. Funk and S. Blake -Wilson, “EAP Tunneled TLS Authentication Protocol version 0,” RFC 4017, 2005.

[80] C. Rigney and A. Rubens, “Remote Authentication Dial in User Service (RADIUS),” RFC 2138, 1997.

[81] R. Fantacci, L. Maccari, and T. Pecorella, “Analysis of Secure Handover for IEEE 802.1X-Based Wireless Ad Hoc Networks,” In the IEEE Wireless Communications Magazine, October 2007.

[82] E. Perez, “802.11i: How We Got Here and Where are We Headed,” SANS Institute 2004.

[83] C. L.-Maknavicius and H. Chaouchi, “Security Architecture in a Multi-hop Mesh Network,” In the Proc. of the Safety and Architecture Networks, 2006.

[84] Y. Cheng, B. Xie, D. Wang, and D. P. Agrawal, “Secure Authentication and Data Communication in Wireless Mesh Networks,” Technical Report, Univ of Cincinnati, 2007.

[85] D. Forsberg, Y. Ohba, B. Patil, and H. Tschofenig, “Protocol for Carrying Authentication and Network Access (PANA),” IETF Draft (work in progress), March 2006.

[86] Y. Zhang and Y. Fang, “A Secure Authentication and Billing Architecture for Wireless Mesh Networks,” In the Journal of Wireless Networks, Vol. 13, No. 5, October 2007.

[87] Y. Hu, D. B. Johnson, and A. Perrig, “SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks,” Ad Hoc Networks, pp. 175-192, 2003.

[88] L. Zhou and C. V. Ravishankar, “Dynamic Merkle Trees for Verifying Privileges in Sensor Networks,” In the Proc. of IEEE ICC, Vol. 5, pp. 2276-2282, 2006.

[89] D. Stebila, “Slightly Improved Merkle Tree Traversal for User Authentication Using Psuedorandomly-Generated Leaves,” University of Waterloo, ON, Canada, 2006.

[90] B. Wehbi, W. Mallouli, and A. Cavalli, “Light Weight Client Management Protocol for Wireless Mesh Networks,” In the Proc. of the MDM, 2006.

[91] R. Housley, W. Polk, W. Ford, and D. Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL profile,” IETF, RFC-3280, 2002.

[92] Nokia RoofTop Wireless Routing. White paper.

[93] Radiant Networks Website – www.radiantnetworks.co.uk

[94] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, “Link-level Measurements from an 802.11b Mesh Network,” In the Proc. of SIGCOMM, 2004.

[95] IEEE Standard, “Wireless LAN-Medium Access Control and Physical Layer Specification,” P802.11, 1999.

[96] S. Khattab, R. Melhem, D. Mosse, and T. Znati, “Honeypot back-propagation for mitigating spoofing distributed Denial-of-Service attacks,” In the Proc of IPDPS, 2006.