**The Pennsylvania State University**
**The Graduate School**
**Department of Computer Science and Engineering**

# DISTRIBUTED DENIAL OF SERVICE ATTACKS IN IEEE 802.11S WIRELESS MESH NETWORKS

A Thesis in
Computer Science and Engineering
by
Sudeep Dutt



Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2009

The thesis of Sudeep Dutt was reviewed and approved* by the following:

George Kesidis
Professor of Electrical Engineering and Computer Science and Engineering
Thesis Advisor

Sencun Zhu
Assistant Professor of Computer Science and Engineering

Mahmut Kandemir
Associate Professor of Computer Science and Engineering
Director of Graduate Affairs

*Signatures are on file in the Graduate School.

# Abstract

*IEEE 802.11 Wireless Local Area Networks (WLAN's) have become ubiquitous over the past few years. IEEE 802.11s Wireless Mesh Networks (WMNs) can potentially provide cheap city wide internet connectivity over the next decade. Security is a major concern for wireless networks due to the easily accessibly unbounded communication medium. The focus in this thesis is Distributed Denial of Service Attacks (DDoS) in WMNs. The central problem is that management, control and action frames are not protected in WMNs. Relatively inexpensive equipment can be used to inject malicious management, control and action frames in the WMN's resulting in complete unavailability of the network along with significant depletion of network and power resources. Sybil, Masquerading and Routing DDoS attacks have been focused upon. Solutions have been implemented in the Medium Access Control (MAC) layer. The attacks and the solutions have been simulated in the WLAN network simulator GTNetS. The results show that the solutions render the attacks ineffective with minimum overhead. This thesis reveals the security requirements for WMNs in order to successfully counter DDoS attacks.*

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

I would like to thank my advisor, Dr. George Kesidis, for his guidance and support right through my Masters. He is the prefect mentor and each meeting with him opened my mind to new possibilities and helped me grow as a computer scientist. I am extremely grateful for his patience and support, without which I could not have been successful.

I would also like to thank Dr. Sencun Zhu for introducing me to the process of research in my first semester and for helping me right through. I am grateful for all the support I received from the Computer Science department during the course of my degree. I am also grateful to my student colleague, Hayawardh Vijay Kumar for his advice and insight during the course of my work.

I would like to thank my fiancee Sonia for her immense patience and for being a constant source of encouragement for me. Lastly I would like to thank my father Mr. Nabendu Kumar Dutt, my mother Mrs. Navanita Dutt and my entire family for always being my source of inspiration and strength.

Chapter 1

# Introduction

## 1.1 802.11 Wireless LAN

IEEE 802.11 Wireless Local Area Networks (WLANs) [1] have gained immense popularity over the past decade. Infrastructure wireless networks are providing relatively cheap Internet connectivity in mobile and stationary environments with high performance. The applications for WLAN's are diverse in nature and include applications ranging from home wireless networks to critical military communication in hostile environments.

Figure 1.1 shows the increasing usage statistics of IEEE 802.11 WLAN's over the years [2].

## 1.2 802.11s Wireless Mesh Networks

The next generation of wireless networks includes city-wide multi-hop Wireless Mesh Networks (WMN's) or IEEE 802.11s [3, 4, 5, 6]. The key deployment scenarios for WMN's include large corporate offices, university campuses, residential

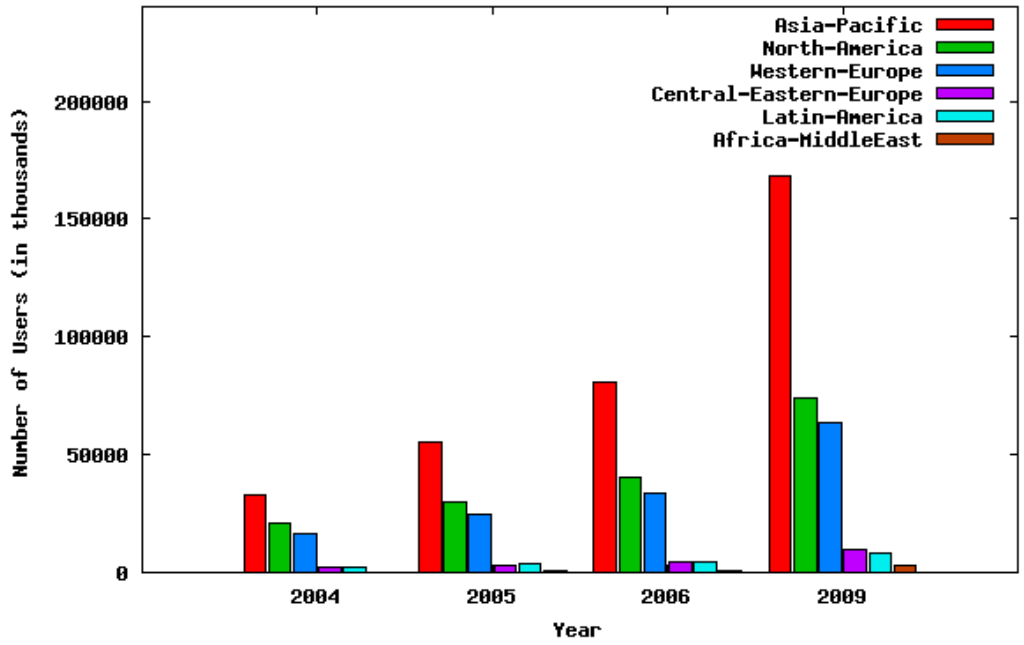


**Figure 1.1.** IEEE 802.11 Users By Region (in thousands)

layouts, military communication and crises management teams. There are several advantages of WMN's. They are inexpensive to deploy in difficult terrains and hostile environments. They consist of self healing robust networks providing fault tolerance. Last mile connectivity is ensured for stations through multi-hop wireless links along with conservation of battery power. WMN's also provide several new challenges in terms of security, route management, Quality of Service (QoS) and interoperability with other protocols.

## 1.3 802.11s - The need for Research

It is a challenging task to secure wireless networks due to the unbounded nature of the communication medium. The focus in this thesis is Distributed Denial of Service (DDoS) [7] attacks in 802.11s WMN's. Security was a focus of 802.11i

[8]. 802.11i has been able to provide confidentiality of data frames. However management, control and action frames are not protected and provide an opportunity for attackers to perform DDoS attacks. These attacks have been seen to result in complete unavailability of the wireless network along with depletion of battery power which is an important resource in mobile environments.

These attacks are particularly devastating in 802.11s WMN's because a single packet could have traversed multiple hops and already consumed a significant amount of network and power resources before it gets dropped due to a DDoS attack. Researchers in academia and industry have been aware of DoS attacks in 802.11 WLAN's [9]. However implementable solutions have not been introduced in the firmware/software of WLAN Station and Access Point solutions since the effects of a successful DDoS attack could till now only affect a single infrastructure Basic Service Set (BSS). With the introduction of 802.11s WMNs it has become mandatory to provide a security architecture which will render these attacks ineffective. Unless this is done attackers will be able to disable city-wide internet connectivity using coordinated DDoS attacks with inexpensive equipment.

Some of the DDoS attacks discussed are as follows:

- Sybil DDoS Attacks. [10, 11, 12]
- Masquerading DDoS Attacks. [13]
- Routing DDoS Attacks. [14, 15]

The solutions for each of these attacks are implemented and explained. The experiments were performed using the Georgia Tech Network Simulator (GTNetS) environment [16]. The solutions are implemented in the Medium Access Control(MAC) layer. The results show that the solutions implemented render the DDoS attacks useless with minimum overhead.

## 1.4 Thesis Contributions

The contributions of this thesis are as follows:

- To explain the working of 802.11 WLAN's and WMN's. This includes important procedures like authentication, association, beacons, scanning, power save mechanisms, Disassociation, Deauthentication, Reassociation, Pre-authentication and Clock Synchronization.

- To explain how DDoS attacks can be performed easily in 802.11 WMN's. Most of the DDoS attacks discussed involve Sybil attacks, Masquerading Attacks and Routing DDoS attacks. The specific attacks discussed involve Shared Key Authentication, deauthentication, disassociation, attacks against stations in power save, routing attacks and jamming attacks [17].

- To provide MAC layer solutions to prevent some of the DDoS attacks mentioned. The solutions are designed so that they require changes mostly to the MAC layer at the Access Point. This is an important requirement to ensure easy upgrades to existing Access Point firmware in the field.

- To highlight the security requirements of WMN's so that they can be robust towards DDoS attacks. Permanent robust solutions are required to counter the threat from DDoS attacks. The overhead involved is minimal since most of the solutions are triggered as counter measures which are deployed only on detection of a DDoS attack.

## 1.5 Thesis Format

The rest of this thesis is organized as follows. Chapter 2 provides a brief background about the working of WLAN's and WMN's. This chapter first details the working of 802.11 WLAN's. This is followed by the enhancements in 802.11 WMN and its features. Chapter 3 focuses on the related work on DoS attacks in 802.11 WLAN's and wireless adhoc networks.

Chapter 4 details the DDoS attacks and the solutions. The attacks are broadly classified in three categories which are Intra BSS attacks, Routing Attacks and Attacks on the Mesh Portal Point. Sybil, Masquerading and Routing attacks have been focused upon.

Chapter 5 details the GTNetS Simulation framework. This is followed by an analysis of the experimental simulation results. The implementation overhead is discussed as well.

Chapter 6 discusses the security requirements of WMN's to successfully counter DDoS attacks. The management, control and action frames of importance for each category of DDoS attacks are highlighted in this section.

Chapter 7 presents the conclusion of this thesis along with the future work in this area. Appendix A contains the source code for 802.11 MAC modifications required for feature additions and packet injections to the GTNetS simulation framework.

Chapter 2

# Background

## 2.1 802.11 WLAN Overview

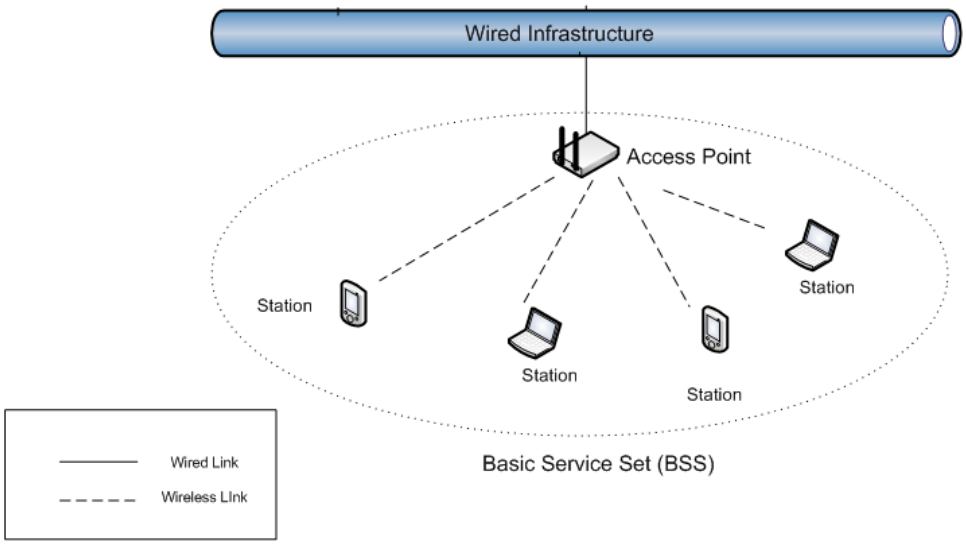


**Figure 2.1.** 802.11 Basic Service Set

### 2.1.1 802.11 Basic Service Set

Figure 2.1 shows a Basic Service Set in a 802.11 WLAN with one Access Point and numerous mobile Stations. The Access Point is connected to wired infrastructure

**Table 2.1.** 802.11 a/b/g/n Features

| Protocol | Throughput | Operational Frequency |
|---|---|---|
| 802.11a | 54 Mbps | 5 GHz |
| 802.11b | 11 Mbps | 2.4 GHz |
| 802.11g | 54 Mbps | 2.4 GHz |
| 802.11n | 600 Mbps | 5 GHz and/or 2.4 GHz |

as shown.

### 2.1.2 802.11 Protocols

Table 2.1 shows the characteristics of various important protocols in the 802.11 WLAN family. There are several enhancements to the original 802.11 specification. Some of the important ones are discussed below:

- 802.11a - Legacy 802.11 Protocol supporting 54 Mbps at 5 GHz. [1]
- 802.11b - Enhancements to support 11Mbps at 2.4 GHz. [19]
- 802.11e - Enhancements for Quality of Service (QoS). The most significant feature was the addition of separate access categories for video, voice, best effort and background data. Other features include block acknowledgement, direct link protocol and automatic power save delivery. [20]
- 802.11i - Security enhancements including significant feature additions like CCMP (WPA), TKIP (WPA2) and Key Management to provide encryption, non-repudiation and data integrity. [8]
- 802.11n - Enhancements for higher throughputs targeted at 600 Mbps at 5 GHz and/or 2.4 GHz. [26]
- 802.11s - WLAN support for city-wide wireless mesh networks. [3]

- 802.11w - WLAN support for Protected Management Frames. [29]

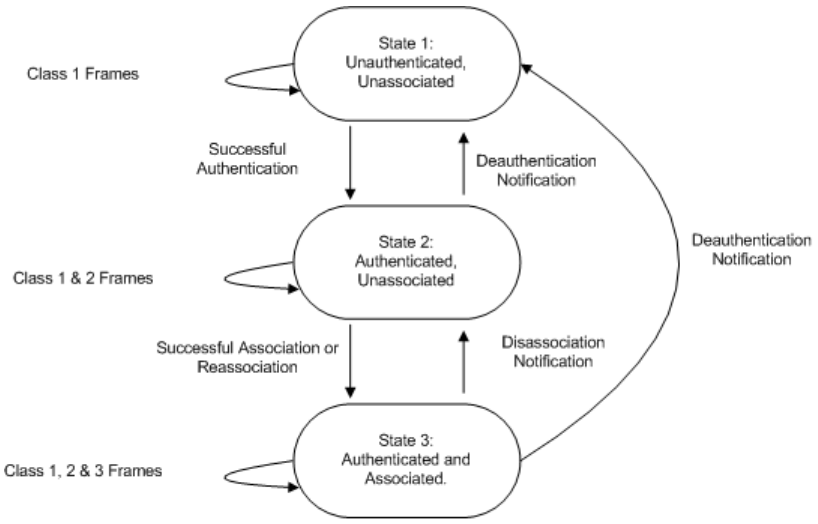


**Figure 2.2.** 802.11 Station State Machine

### 2.1.3 802.11 Station State Machine

The procedure for a Station to associate with an Access Point [1] is discussed next. The station state machine diagram is shown in Figure 2.2. Initially all stations are in state 1 i.e. unauthenticated and unassociated. At this point the stations can only receive Class 1 frames. Class 1 control frames include RTS, CTS, ACK and CF-END. Class 1 Management Frames include Probe Request/Response, Beacon, Authentication, Deauthentication and ATIM frames. In order to transition from State 1 to State 2, initially a station has to select which Access Point it wants to authenticate with. There are two techniques for doing so. The first technique is called *Passive Scanning* where a Station waits for a Beacon Frame broadcasted by the Access Point at regular intervals. The second technique is called *Active Scanning* where a Station explicitly sends a Probe Request frame to a known

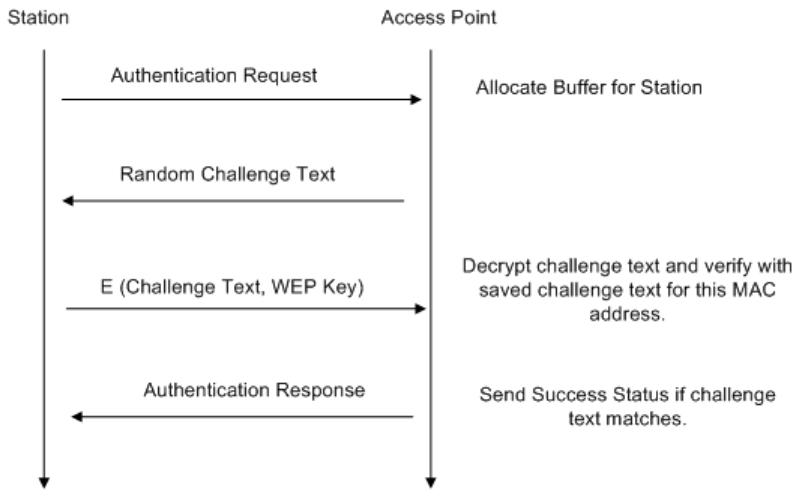


**Figure 2.3.** Shared Key Authentication

Access Point and waits for the probe response. Once the station receives a probe response or a beacon, the next step is to authenticate with the access point.

### 2.1.4 802.11 Authentication & Association

There are various techniques for authentication. The first technique is known as *Open System Authentication* [1]. In this rather superficial authentication method, an Authentication Request frame is sent by the Station and the Access Point must send an Authentication Response frame with a successful status if the limit for the maximum number of stations it can handle simultaneously is not reached. The second technique for authentication is known as *Shared Key Authentication* [1] and is shown in Figure 2.3. This is a more robust Authentication technique and involves a four message hand-shake between the Station and the Access Point. In the second message the Access Point sends the challenge text in the clear to the Station. The Station encrypts this challenge text and sends it back to the

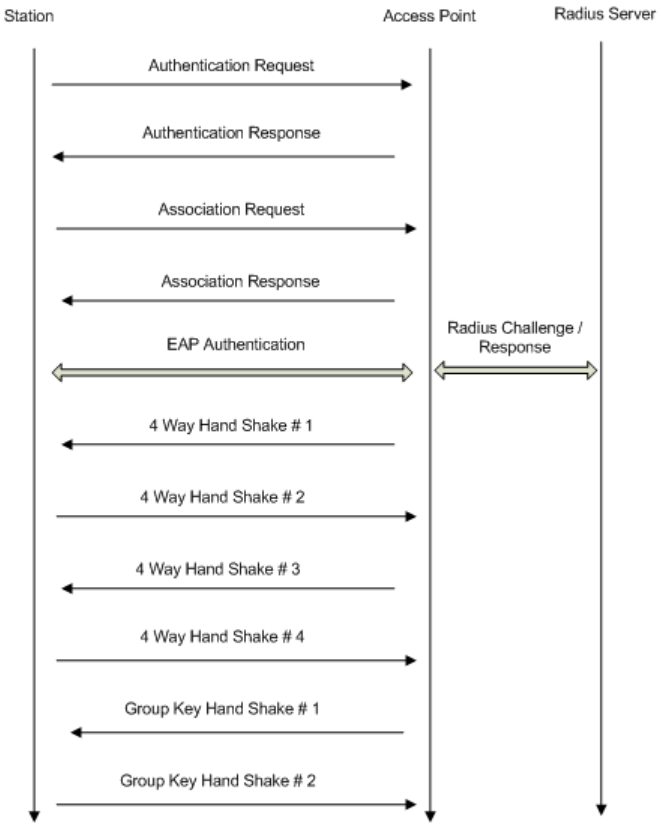


**Figure 2.4.** 802.1X Authentication

Access Point which verifies the challenge text after decryption and sends a success or failure status message in the final Authentication response message. The third technique used for Authentication in 802.11 WLAN's is 802.1X Authentication which results in transient pairwise and group keys. The message exchanges are shown in Figure 2.4. The initial message exchanges consist of Open System Authentication and Association Request and Responses. This is followed by a mutual authentication following the Extensible Authentication Protocol (EAP) [32]. Commonly used techniques for EAP in wireless networks include EAP-TLS [33], PEAP [35], LEAP [34] and EAP-TTLS [36]. This is followed by a four way handshake which results in a pairwise transient key being established at Station and Access Point. This may be followed by a Group Way Handshake resulting

in a Group Key for Broadcast/Multicast frames. At the end of this exchange the station and Access Point can exchange Data frames over a confidential and secure channel using either Advanced Encryption Standard (AES) [37] based algorithm called Counter Mode with Cipher Block Chaining Message Authentication Protocol (CCMP) [38] i.e. WPA or Temporal Key Integrity Protocol (TKIP) known as WPA2. WPA is more secure and less vulnerable to attacks compared to WPA2.

The final transition from State 2 to State 3 occurs when a Station receives a successful Association Response from the Access Point. Once a station is authenticated and associated with an Access Point it is eligible to transmit and receive Data frames. The Station periodically synchronizes its clock with the timestamp field in the beacon frames received from the Access Point. Correct functionality of 802.11 MAC is closely dependent on accurate time synchronization among all the devices in the BSS.

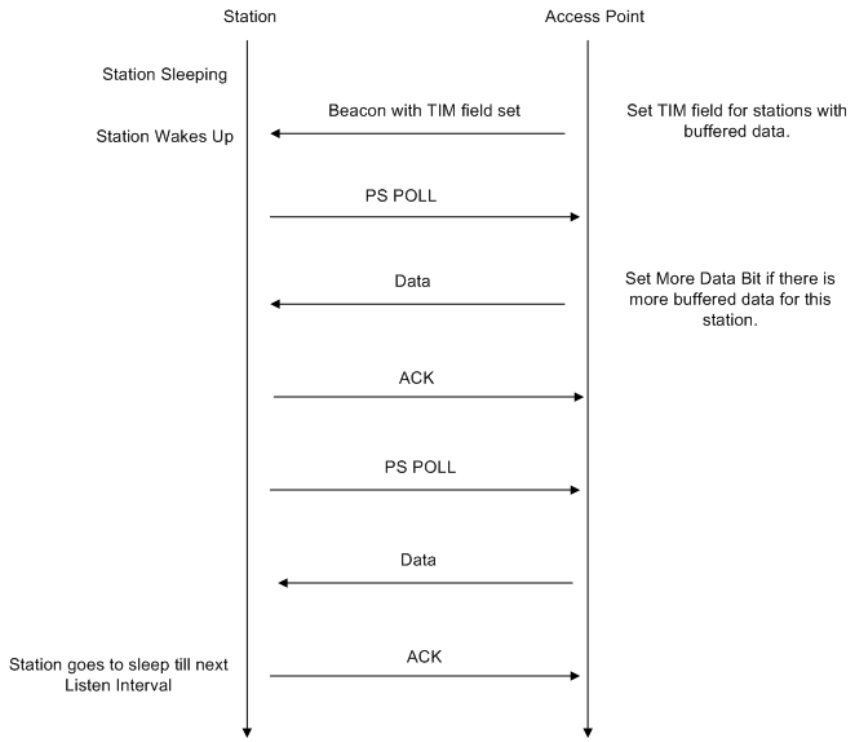


**Figure 2.5.** 802.11 Power Save

### 2.1.5 802.11 Power Save

The next important aspect of 802.11 WLAN's is the power save functionality [1] shown in Figure 2.5. Every station is configured with a specific Listen Interval during Association. The station switches off its radio and baseband clock during the listen interval period. It wakes up every listen interval number of beacons and checks the Traffic Indication MAP (TIM) field in the beacon to determine if there are packets buffered for it at the Access Point. If the TIM field is 0 then the station goes back to sleep till the next Listen Interval. If the TIM field is set then the station remains awake and sends a Power Save Poll (PS POLL) control frame to the Access Point. The Access Point on receipt of a PS POLL frame from the station transmits a single data frame buffered for the station and sets the More Data bit in the Frame Control field of the data frame if more packets are buffered for this station. On receipt of the data frame, the station responds with an ACK frame followed by another PS POLL frame depending on whether the More Data bit is set or not. This works fine with unicast data frames for individual stations. All stations observe the first element of the TIM field to check if there is broadcast/multicast data buffered at the Access Point. If there is broadcast/multicast data buffered then the stations do no go back to sleep and wait for the data to be transmitted by the Access Point. PS POLL frames are not sent for broadcast/multicast frames. Broadcast/Multicast frames are not acknowledged by the recipients and are not retried by the Access Point either.

### 2.1.6 802.11 Deauthentication & Disassociation

A station can transition from state 3 to state 2 by sending a disassociation frame [1] to the Access Point. A station can also transition directly from State 3 to State

1 or State 2 to State 1 by sending a Deauthentication [1] frame to the Access Point. These are just notifications and there is no explicit check by the Access Point to authenticate the Disassociation or Deauthentication frame. A station can also transition from State 2 to State 3 by sending a Reassociation frame to the Access Point. A mobile station can preauthenticate [41] itself with several Access Points so that it can transition from one Access Point to the other seamlessly. This is especially useful for Voice over WLAN applications [46].

## 2.2 802.11s Wireless Mesh Networks

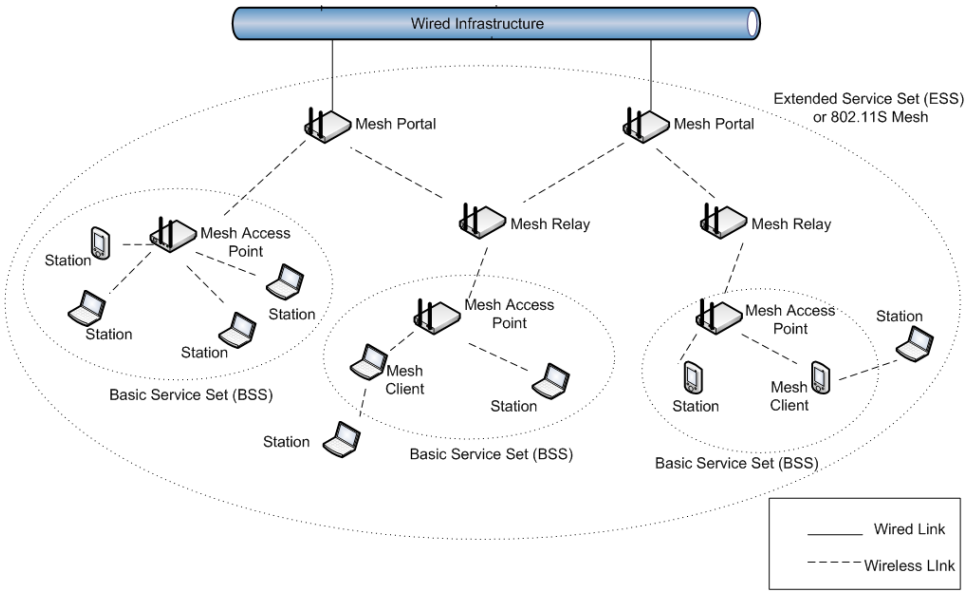


**Figure 2.6.** 802.11s Wireless Mesh Network

### 2.2.1 802.11s Extended Service Set

Figure 2.6 shows a 802.11s Wireless Mesh Network. As an example three BSS are shown but the scope of 802.11s is especially suited to city-wide wireless mesh networks. Such a collection of BSSs is known as an Extended Service Set (ESS).

The *Mesh Portals* are the devices which are connected to the wired infrastructure. A point at which MSDUs' exit and enter a WLAN Mesh to and from other parts of a distribution system or to and from a non-802.11 network. The *Mesh Relays* are responsible for forwarding data between the Mesh Portals and the *Mesh Access Points* and vice versa. The Mesh Access Points provide all the functionalities of an Access Point in 802.11 WLAN along with additional features like Optimum Route Discovery, Path Selection and Forwarding. The Hybrid Wireless Mesh Protocol (HWMP) [42, 43] is the default routing protocol. Mesh Portals [45], Mesh Relays and Mesh Access Points together participate in the organization and formation of a mesh network. HWMP combines the flexibility of on-demand routing with proactive topology tree formation. The combination of reactive and proactive elements of HWMP enables optimal and efficient path selection. It is also interesting to note that 802.11s supports multi-hop clients which are not within the coverage area of a Mesh Access Point in a BSS but might be in coverage of a Mesh Station. This feature greatly improves the last mile end-connectivity for stations in a Mesh Network.

### 2.2.2 802.11s MAC Functionalities

The responsibilities of a 802.11s MAC for successful deployment of a WMN in addition to traditional 802.11 WLAN functionalities, are as follows:

- Mesh Topology Learning is required to detect peers and find the optimal path to a destination.
- Optimal Routing [48] and Forwarding of packets needs to be performed. The topology is dynamically evolving and failures need to be detected early in order to traverse alternate routes.

- Quality of Service [47] guarantees and bandwidth management is important. It is feasible to provide transmission and reception on different channels for different access categories. There is also provision for different routes being traversed for different access categories.

- Security and key management [49, 50, 51] is important for successful deployment of a WMN. These requirements differ from standard 802.11 WLAN due to presence of multiple hop communication. Some of the earlier 802.11 WLAN security handshakes have been designed keeping single hop communication between the station and Access Point in perspective only.

- Congestion control [47] is required at the link layer. Legacy 802.11 WLAN along with 802.11e had been designed as a single hop wireless network. However in order to provide an end to end solution for congestion control in a WMN, a technique for local congestion monitoring is required.

### 2.2.3 802.11s Advantages & Challenges

Some of the key advantages of a 802.11s Wireless Mesh Network are as follows:

- Relatively inexpensive to deploy a large mesh network in a short turn around time. The infrastructure and man power required to deploy a large mesh network is much less when compared to wired network. It is also extremely difficult to install a wired network to an existing facility due to construction difficulties.

- Easy to provide connectivity in difficult terrains and hostile environments where deploying a wired network is not feasible. This is particularly useful in large areas which might require a wireless network for maybe a couple

of months. Examples of such scenarios include military missions in hostile environments and research camps.

- Self-Healing and robust network with Dynamic Route Discovery and Optimal Path Selection algorithms. Fault tolerance is a mandatory requirement for computer networks. A single point of failure should not be responsible for significant depletion in performance of the overall network.

- Last mile end connectivity for stations through multi-hop forwarding techniques. This is particularly useful for stations at the edge of a large mesh network.

- Conservation of battery life with efficient power save mechanisms [47] and lower power transmissions. It is predicted that most of the stations in a large mesh network will be mobile handsets with battery power as a critical resource.

Some of the key deployment scenarios for 802.11s Wireless Mesh Networks are large corporate offices, university campuses, residential layouts, surveillance systems, military communication and crises management teams. Although the advantages and applications of Wireless Mesh Networks are many, new challenges are brought forward as well. The biggest challenge is to secure a mesh network of such large magnitude with minimal overhead, possibly 32 hops from end to end. Introducing secure links with replay protection and prevention of Distributed Denial of Service attacks is a mandatory requirement. Other challenges include global clock synchronization, effective power save techniques, maintaining Quality of Service for voice/video applications and interoperability with other protocols like 802.3 [39] and 802.16 [40].

Chapter 3

# Related Work

We are not aware of existing research focusing on DDoS attacks in 802.11s WMN. However the inherent nature of 802.11s WMN is a combination of IEEE 802.11 Infrastructure WLAN and wireless ad hoc links. The following sections discuss the related work in these areas.

## 3.1 802.11 Infrastructure DoS Attacks

The central problem with IEEE 802.11 WLAN's is the lack of priority given to security requirements during the design phase of the initial specification [1]. IEEE 802.11i [8] was designed as a retroactive amendment to the initial specification. Arbaugh et al. [52] described vulnerabilities in WEP and the Shared Key Authentication procedure in IEEE 802.11 WLAN. Bernaschi et al. [53] describe Probe Request Floods, Authentication Request Floods and Association Request Floods without specifying solutions. Borisov et al. [54] describe vulnerabilities in the WEP protocol used for data exchange and key management in legacy 802.11 WLAN. Kyasanur et al. [55] describe solutions for malicious stations which per-

form MAC layer misbehaviour using NAV attacks. NAV attacks have been seen to have only a local temporary impact. Bellardo et al. [9] have described the Deauthentication, Disassociation, Power Save and NAV DoS attacks. The solution described by them requires the Access Point to wait for five seconds before removing a station from the legitimate station list. They claim that a legitimate station will not be inactive for five seconds. The method described in this thesis has a more direct approach by using IEEE 802.11 NULL frames as described later. This thesis also provides an uniform framework which can be utilized across all control, management and action frames. The HMAC [56] solution described later will ensure integrity and non-repudiation of these frames which are the final security goals. Detection of MAC layer spoofing by analysing the received signal strength pattern has been described by Sheng et al. [57]. Martinovic et al. [11] provide a solution for Sybil attacks based on wireless characteristics like signal propagation and is not a deterministic solution compared to the SYN cookie approach discussed in this thesis. Newsome et al. [12] survey Sybil attacks and solutions in specific detail with respect to wireless sensor networks only. Link and physical layer jamming attacks have been discussed by Wenyuan et al. [58], Gunmadi et al. [59] and Xu et al. [60]. This thesis does not focus on jamming attacks in detail.

## 3.2 Wireless Adhoc DoS Attacks

Zapata et al. [61] explore the challenges with securing adhoc networks. The security objectives described here include import authorization, source authentication and integrity. Hash chains and digital signatures are used as the cryptographic techniques. Black Hole attacks are described by Ramaswamy et al. [14]. They

introduce Data Routing Information Table and Cross Checking to develop a modified AODV protocol to identify multiple black holes colluding together. Rushing attacks are described by Hu et al. [62]. A new route discovery algorithm is designed by them which has a significantly high overhead. Defence against random packet injections in an adhoc network is described by Gu et al. [63]. This work focuses on resource exhaustion DoS attacks through injection of junk packets. A hop-by-hop source authentication protocol has been developed. Gu et al. also analyse area congested DDoS attacks in adhoc networks [64]. They categorize the DDoS attacks as remote attacks and local attacks. Jelly Fish and Black Hole attacks are described by Aad et al. [65]. The Jelly Fish attack targets end-to-end TCP congestion control mechanisms. The impact of DoS attacks on adhoc networks is discussed by Aad et al. [66]. Route misbehaviour which is a major emphasis in this thesis has been described by Marti et al. [67]. They have provided a protocol for a watchdog that detects misbehaving nodes and a pathrater that allows routing protocols to avoid such nodes. However since their approach is not scalable, it is difficult to implement their ideas in a large WMN setting. Zhu et al. describe a cross layer dropping attack in video streaming over adhoc networks [68]. This is a specific attack for video applications and exploits the IP protocol. A secure on demand routing protocol based on symmetric key cryptography named Ariadne as been discussed in by Perrig et al. [69]. This work is closely related to the HMAC solution provided in this thesis for routing attacks since it is also based on symmetric key cryptography and uses per hop hash chains. This fits the scheme of 802.11s WMN which utilizes 802.11i [8] to provide per link transient keys derived from the 802.1X [31] four way hand shake. Public key cryptography techniques discussed by Pirzada et al. [70] are not suitable for WMN's due to the massive overhead involved in key distribution.

Chapter 4

# Distributed Denial of Service Attacks and Solutions

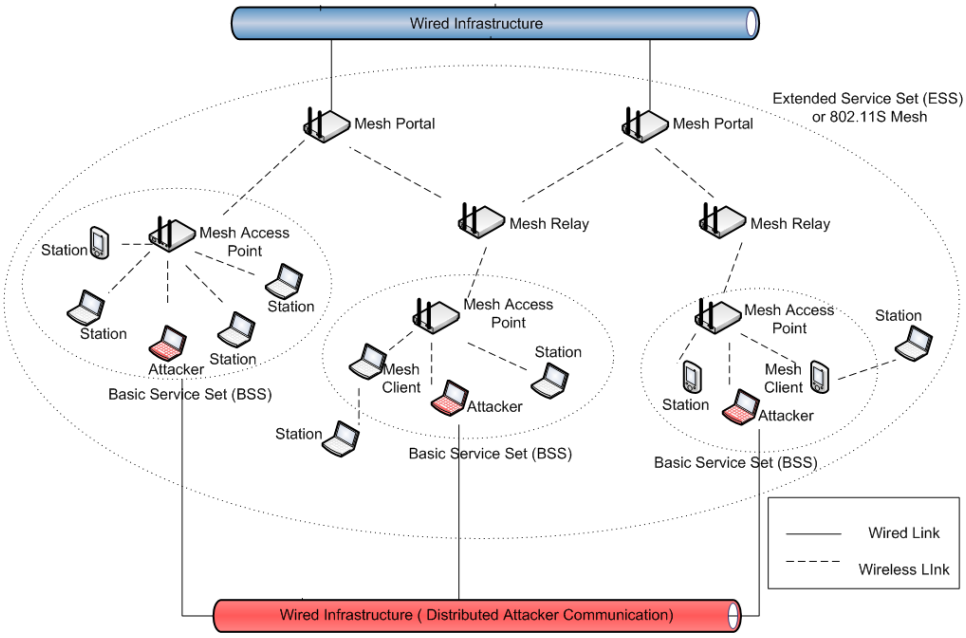


**Figure 4.1.** 802.11s Attack Scenario

## 4.1 Threat Model

Figure 4.1 shows a generic example of a coordinated Distributed Denial of Service(DDoS) attack in 802.11s Wireless Mesh Networks. A single malicious station in each BSS of the mesh is enough to bring down the entire network. These malicious devices could even communicate with each other using a separate wired infrastructure as shown in Figure 4.1. Communication between stations is not a mandatory requirement for successful DDoS attacks. Firmware manipulation is not an expensive or a technically challenging task. If a single station in an 802.11s Wireless Mesh Network can be manipulated then it is fairly simple to deploy similar malicious stations in different BSS of the mesh.

The mandatory features of the threat model are described below:

- The malicious devices should be capable of injecting 802.11s Control, Management and Data frames [71, 72]. Control and management frames are focused upon here since these are transmitted in the clear without encryption in an 802.11 WLAN or 802.11s WMN. Injection of management frames is feasible via firmware manipulation. It is difficult to inject Control Frames like RTS, CTS and ACK frames since these need to be transmitted from the hardware due to strict time constraints like the Short Inter-Frame Space (SIFS) [1] which is in the order of tens of microseconds. A strong threat model will allow injection of Control Frames though this is not practically feasible. A weak threat model will not allow injection of time critical Control Frames. The solutions presented in this thesis take both threat models into consideration.

- The malicious devices should be capable of spoofing its Medium Access Control (MAC) Address [74]. Most 802.11 WLAN and 802.11s WMN allow mod-

ification of the MAC address by issuing a software command to the driver e.g. IOCTL in Linux [73]. Successful attacks require injection of frames with spoofed MAC addresses at the highest transmission rate possible.

- The malicious device should be able to sniff 802.11 WLAN and 802.11 WMN frames by operating in *promiscuous* mode [75]. Management and Control Frames are transmitted in the clear and important information like the MAC address of legitimate clients can be detected easily. Most WLAN stations are capable of operating in promiscuous mode. In addition, WLAN sniffer software packages like Airopeek [77] and Wire Shark [76] are easily available. In order to successfully carry out the masquerading attacks like the Deauthentication or Power Save attacks, it is extremely important for the attacker to detect the MAC addresses of legitimate stations. The fact that a Station is utilizing the power save features can also be detected by sniffing for PS POLL frames [1].

Certain additional capabilities of the attacker can result in a strong threat model. The optional features of the threat model are described below:

- The malicious devices should be capable of transmitting an ACK frame within Short Inter Frame Space (SIFS) of receipt of a data frame. The data frame has the destination as a legitimate station whose MAC address has been spoofed by the malicious device. The ability of transmitting an ACK frame requires modification in the 802.11 firmware and most likely the hardware part of the firmware. This additional capability is specifically useful for the DDoS attack against stations in power save described later in this chapter.

- The malicious devices should be able to communicate with other devices in different BSS's of the Mesh Network via a separate communication channel. This channel could be a wired infrastructure network as shown in Figure 4.1. The details of the communication channel are not within the scope of this thesis. Communication between the various malicious devices could effectively allow simultaneous coordinated DDoS attacks across the different BSS's in the 802.11s WMN resulting in a successful DDoS attack which could provide complete unavailability of a city-wide network.

- The malicious devices should be capable of transmitting on multiple channels in order to exploit Multiple Input Multiple Output (MIMO) [78] techniques. This is a smart antenna technology by which multiple antennas are utilized at the transmitter and the receiver to improve overall throughput. Frames are transmitted and received over multiple channels at the same time. Simultaneous injection of frames in different channels can provide a much greater impact and result in a more successful attack. When an Access Point detects congestion in one channel probably due to a DDoS attack then it can change the default channel. However the attacker's capability to transmit on different channels at the same time is a potent threat.

## 4.2 Types of Attacks

The sheer size of a mesh network along with the different types of devices and links makes security a fairly difficult task. Various attacks can be exploited at different layers in the network. The following is a list of the attacks that will be concentrated upon in this thesis. The attacks are focused on the MAC layer.

- Intra-BSS DDoS Attacks.
- Routing DDoS Attacks.
- DDoS Attacks on Mesh Portal Points.

## 4.3 Intra-BSS DDoS Attacks

The first type of attack we will explore are Intra-BSS attacks where each BSS has a malicious station carrying out DoS attacks. The DoS attacks are termed as "distributed" because the attackers co-ordinate with each other to carry out the attacks at the exact same time in different BSS in the network thereby bringing down the entire WMN. The various attacks and solutions are discussed next.

### 4.3.1 Shared Key Authentication Sybil Attack

Every Access Point implementation has a limit on the maximum number of stations it can support simultaneously. Each time a new station associates or authenticates with an Access Point a single station buffer is reserved. The Shared Key Authentication attack is an example of a *Sybil attack* [10]. In a Sybil attack a malicious station creates a large number of bogus client entities which appear valid to the central server thereby depleting the ability of the server to accommodate legitimate clients once the maximum limit is reached.

The Shared Key message hand shake can be depicted in Figure 2.3. As soon as an Access Point receives an Authentication Request it reserves a buffer for this station. Since the station is bogus it will not be able to complete the Authentication procedure as it does not have the correct WEP keys. However a station buffer is used up at the Access Point. The malicious station can continuously create

bogus Authentication stations by spoofing the MAC address. Once all the buffers are exhausted at the Access Point, legitimate stations cannot authenticate and associate with the Access Point resulting in complete unavailability of the network.

### 4.3.2 Shared Key Authentication Solution

The availability of a random challenge text provides an elegant solution. State can be offloaded to the client via a mechanism similar to TCP SYN Cookies [79]. SYN cookies provide a similar solution during TCP Connection Establishment. During a resource exhaustion TCP SYN flood, a client floods a server with SYN packets, forcing the server to allocate state information for the malicious connection. As the client never completes the connection, the available buffers at the server gets exhausted and it cannot accept new legitimate connections.

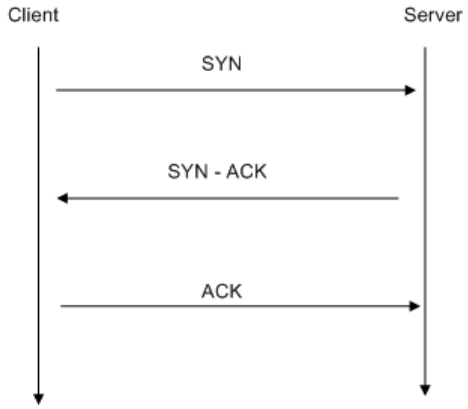


**Figure 4.2.** TCP Connection Establishment

A cookie is a mechanism to offload state information to a client machine. The three way handshake for TCP [82] connection establishment is shown in Figure

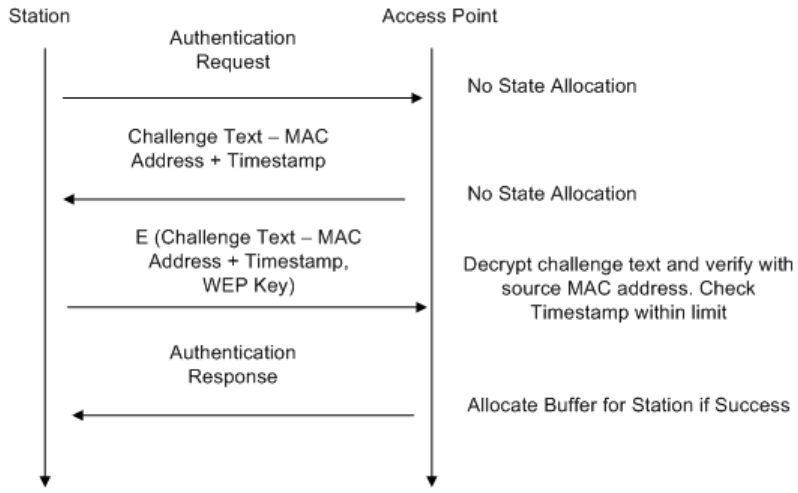


**Figure 4.3.** Shared Key Authentication Solution

4.2. Normally, the server allocates state for the client when the SYN is received allowing the possibility of SYN flood attacks. If the server postpones the allocation of state to when the ACK is received then only valid handshakes will result in a buffer being used up. The key idea behind SYN cookies is for the state to be offloaded to the client. An 802.11 WLAN Access Point maintains some state for every station it sends an authentication response with a success status. In order to avoid resource depletion at the Access Point a technique similar to TCP SYN cookies is used to maintain information about a station which has sent an Authentication Request in a *stateless* manner at the Access Point. The solution is shown in Figure 4.3. When the Access Point receives an Authentication Request during Shared Key Authentication it sends an arbitrary challenge text in clear text to the station in the current implementations. This provides an opportunity to offload the state to the station by sending the concatenation of the source MAC address and the third octet of the timestamp as the challenge text. The timestamp

is used to prevent replay attacks. At this point the Access Point need not maintain any information about the station which sent the Authentication Request. When the legitimate station receives the challenge text it will encrypt it using its WEP key and send the encrypted challenge text back to the Access Point. On receipt of this encrypted challenge text the Access Point will decrypt this packet and compare it with the MAC address of the source. The timestamp is checked and ensured to be within a particular time frame since the challenge text was sent. It is important that the time frame within which the encrypted challenge needs to be transmitted be kept very short to ensure that an attacker cannot exploit the delay and replay previously sent Authentication responses from legitimate stations. If the Access Point successfully verifies that the station has provided the correct encrypted challenge text then an Authentication Response with a successful status is sent to the station. At this stage the state for the legitimately authenticated station is created at the Access Point. A rogue station will not have the capability to encrypt the challenge text correctly due to unavailability of the correct WEP key.

### 4.3.3 Deauthentication/Disassociation Attack

This is an example of a masquerading [13] DDoS attack. When a client wants to disconnect from the current access point, it sends a Deauthentication (or Disassociation) frame to the access point. On receipt of the Deauthentication (or Disassociation) frame, it is removed from the list of authenticated (or associated) stations at the Access Point. The station can no longer send Data frames to the Access Point since it is in State 1 (or State 2) of the IEEE 802.11 management state machine. It is of importance to note that Deauthentication (or Disassociation)

frames, similar to Open System authentication request frames, have no authentication. It acts merely as a notification and can be sent by either the station or the Access Point to terminate the current connection [1].

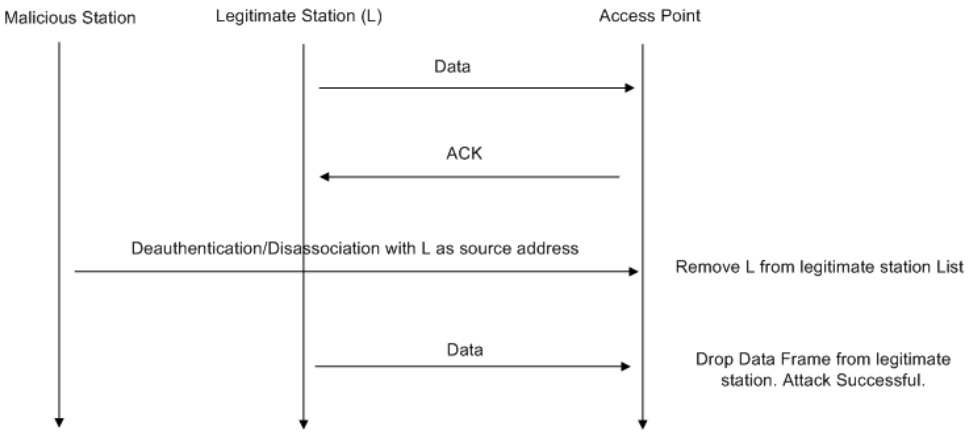


**Figure 4.4.** Deauthentication Attack

This leads to a serious denial-of-service threat as shown in Figure 4.4. A rogue station can monitor the MAC addresses of legitimate stations in a 802.11 WLAN by sniffing packets. Then by spoofing the MAC address of a victim, an attacker sends a Deauthentication (or Disassociation) Frame to the Access Point. The victim station is disconnected from the Access Point and has no way to know that it has been removed from the legitimate authenticated (or associated) stations list at the Access Point. It only senses this once the Access Point starts dropping its Data frames. At this point the station has to undergo the authentication (and/or association) process again. To completely deny the client service, as soon as the victim authenticates (or reassociates), the attacker can send a Deauthentication (or Disassociation) frame again to the Access Point. The impact of a Disassociation attack is slightly lesser since the entire process of authentication does not have to be carried out again.

### 4.3.4 Deauthentication/Disassociation Solution

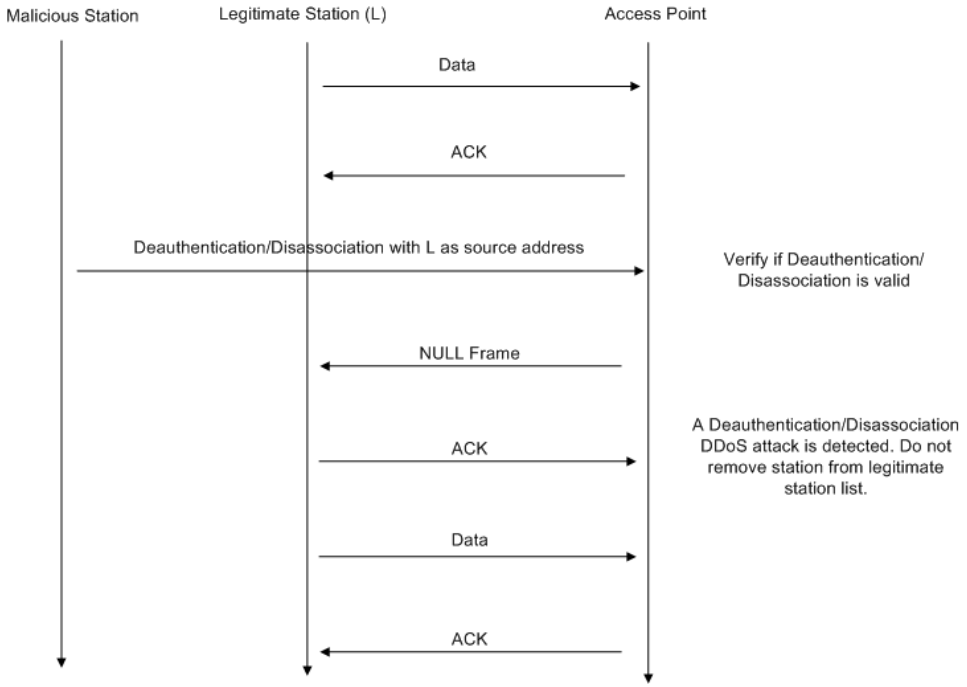


**Figure 4.5.** Deauthentication Solution

The solution shown in Figure 4.5 involves sending an IEEE 802.11 NULL Data frame [1] to the station which has supposedly sent the Deauthentication (or Disassociation) Request. If the station returns an 802.11 ACK then the Access Point is immediately notified that the Deauthentication (or Disassociation) frame it had received is illegitimate or a fake. This is a simple technique to authenticate whether the Deauthentication (or Disassociation) Frame received is legitimate or not. If a station has actually sent a Deauthentication (or Disassociation) Frame then it will not respond to the NULL Data Frame with an ACK. It is of importance to note here that we only send out a NULL Data frame to the station if it is in the authenticated (or associated) station list of the access point. Therefore, it is not possible to create a network flood of NULL Data Frames from the Access Point simply by spoofing Deauthentication (or Disassociation) requests with different

MAC addresses.

This solution works well for the weak threat model where the attacker cannot send ACK Control frames within SIFS time interval as this will require manipulation of the Hardware. However even if we consider a strong threat model where the attacker is capable of sending ACK frames within SIFS interval then this does not serve the real purpose of the attacker. The Access Point will only be made to believe that the Deauthentication (or Disassociation) frame received is malicious. The legitimate station will still remain authenticated (and/or associated) with the Access Point.

### 4.3.5 Power Save Attack

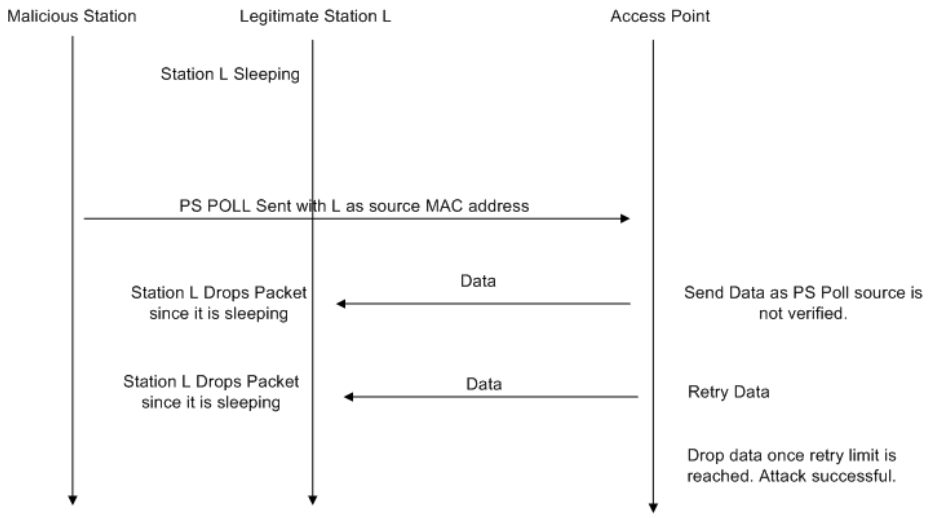


**Figure 4.6.** Power Save Attack

The details of the working of Power Save in 802.11 WLAN [1] has been explained in Chapter 2. This is another example of a masquerading DDoS attack. Initially the legitimate station is sleeping. It informs the Access Point that it is sleeping by setting the Power Management bit in the frame control field of the

last frame it sends to the Access Point before sleeping. The Access Point in turn buffers all unicast data frames for this station while it is sleeping. The station sleeps for a fixed number of beacon intervals known as the Listen Interval. The Access Point sets the Association ID (AID) [1] of this station in the TIM field of the Beacon frame at the listen interval. This is the manner in which the station knows that the Access Point has unicast data buffered for it.

The DDoS attack has been shown in Figure 4.6. The malicious station sniffs the Beacons and PS POLL frames sent by legitimate stations and can thus decipher the mobile stations taking advantage of power save mechanisms. This initial discovery of power save mobile stations is an important step in carrying out the power save DDoS attack. The malicious station can now send PS POLL frames to the Access Point when the legitimate station is sleeping in between its listen interval. The Access Point has no way to verify the PS POLL station and transmits unicast data buffered for the legitimate station. Since the legitimate station is sleeping it does not send an ACK for these data frames. These data frames are retried by the Access Point and eventually dropped once the retry limit is reached. A successful power save DDoS attack is thus been carried out. Data frames buffered at the Access Point which could have already consumed significant network resources by transmission over multiple hops are dropped by the Access Point.

### 4.3.6 Power Save Solution

The solution for the power save DDoS attack is shown in Figure 4.8. The central problem is the inability of the Access Point to detect if a PS POLL frame has been sent by a legitimate station or not. Thus the solution involves the Access Point sending an IEEE 802.11 NULL data frame to the source station sending a

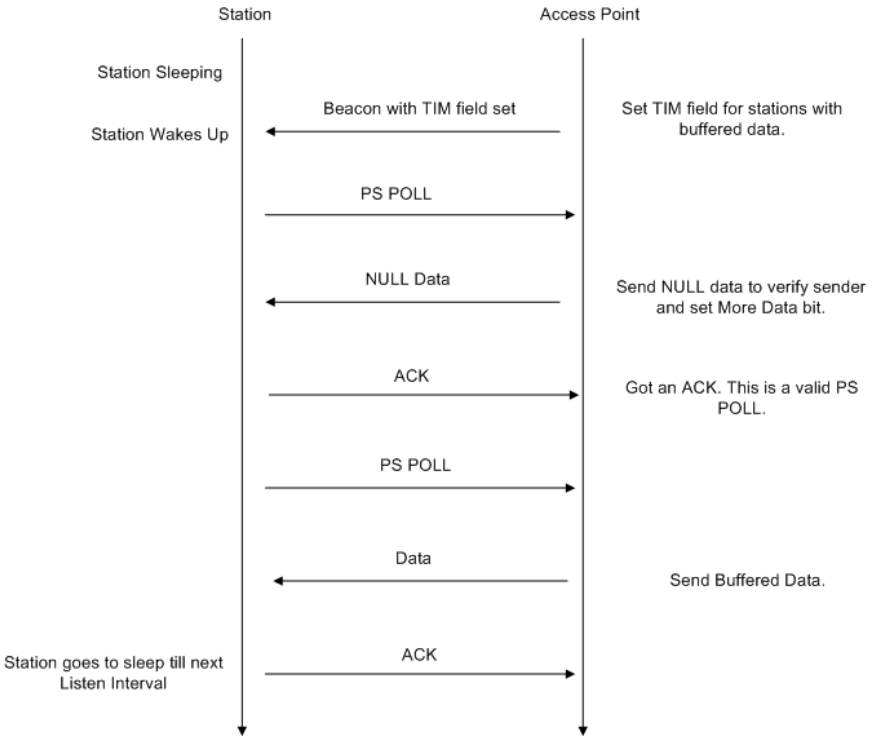


**Figure 4.7.** Power Save Solution - Legitimate PS POLL

PS POLL frame. A legitimate station which has sent a PS POLL frame will not go back to sleep immediately. Hence the legitimate station sends an ACK frame on receipt of a NULL data frame from the Access Point. The *more data* bit is set in the NULL data frame. The Access Point in this manner informs the legitimate station that there is more unicast data buffered at the Access Point in addition to verifying that the sender of the PS POLL frame is legitimate. The station can now pull the remaining buffered data at the Access Point by sending PS POLL frames. The overhead of sending the NULL frame occurs only in the first exchange of the PS POLL frame in a Listen Interval. Since the listen interval is fairly large for mobile power save stations the overhead incurred is minimal.

Figure 4.8 shows the behaviour of the Access Point if a malicious station sends a PS POLL frame. On sending of the NULL data frame an ACK is not received by

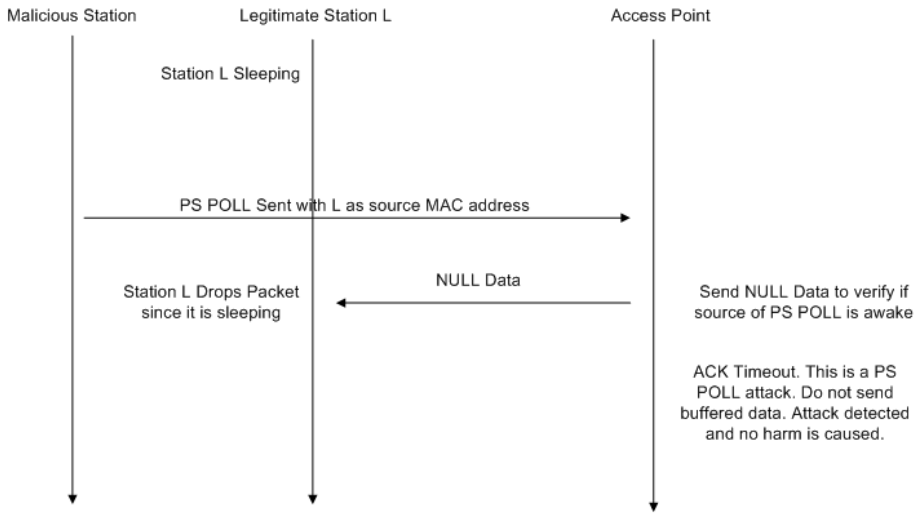


**Figure 4.8.** Power Save Solution - Malicious PS POLL

the Access Point since the legitimate station is sleeping. This ACK timeout serves as a trigger to the Access Point signifying detection of a power save attack by a malicious station. The Access Point does not respond to the malicious PS POLL and does not transmit any unicast buffered data. The attack is thus detected and countered successfully. As an optimization, this technique to detect DoS attacks against stations in Power Save can be triggered as a counter measure the first time buffered data for a station in power save is dropped due to retries.

This solution works well for the weak threat model where the attacker cannot send ACK Control frames within SIFS time interval as this will require manipulation of the Hardware. However if we consider a strong threat model where the attacker can respond with an ACK within SIFS interval, a different approach needs to be explored where an HMAC [56] needs to be provided with the PS POLL frame. Since the Access Point and the station have pairwise keys established between themselves, it is logical to use these keys to provide an HMAC with the PS POLL frames as well. This is to provide non-repudiation and the Access Point will be able to ensure that the sender of the PS POLL Frame is authentic

and has not been transmitted by a malicious attacker who has spoofed the MAC address of a legitimate station since the attacker will not have the correct key. The HMAC needs to be computed by taking the timestamp into consideration as well to prevent replay attacks.

### 4.3.7 Subtle Attacks

This section describes other subtle attacks which do not have the devastating impacts of the earlier attacks discussed in detail. These attacks have not implemented in the simulation setup but seem feasible to carry out in a WMN with relatively low impact.

#### 4.3.7.1 Brute Force Power Save Attack

In this simple attack the malicious station detects the MAC address of legitimate stations utilizing power save mechanisms. It is fair to assume that only mobile stations with limited battery power resource will utilize power save mechanisms. Once detected the malicious station can continuously send NULL data frames or even junk data frames in a legitimate manner to the Access Point. This will result in two interesting scenarios:

- Firstly, the legitimate station will have to remain awake to receive the junk data sent by the malicious station from the Access Point. This will result in depletion of battery power.

- Secondly, a large number of frames sent by the malicious station could deplete the buffers allocated at the Access Point to buffer data frames for stations utilizing power save. This might lead in dropping of legitimate packets which need to be transmitted to the legitimate stations at the Access Point due to

unavailability of power save data frame buffers. The implementation at the Access Point should ensure separate buffers for individual stations else a single station under attack could result in depletion of power save buffers across all stations.

The solutions to this attack can be tackled at various stages:

- The initial policing needs to be performed while authentication and association of legitimate stations. The key distribution needs to be secure to ensure that malicious stations are not allowed to be part of the network. Thus they will not be allowed to send NULL or Junk data to other stations in the network.

- The initial policing might fail due to an insider attack who has valid authentication credentials but is still a malicious station capable of carrying out DDoS attacks. Another valid possibility is modification of the firmware of valid stations by a rootkit [80] controlled possibly by a botnet [81]. In such cases it is possible to detect modification of the firmware binary in the stations by storing a hash of the firmware in non-volatile storage at factory installation and upgrade times. This is an important security requirement for all embedded system firmware.

- NULL frames can be detected by the Access Point. This can act as a trigger for DDoS detection since a station in a BSS has no legitimate reason for sending NULL frames to other stations in the BSS. It is more difficult to detect junk frames at the MAC layer by insider attacks. Some feedback from the higher layers is required to detect such an attack.

#### 4.3.7.2 Forged Beacons

Beacons are transmitted by the Access Point at fixed intervals known as the Beacon Interval and convey crucial information to the different stations in the BSS like accurate Timestamp Synchronization, Security Information Element, Capability, QoS Capability, etc. [1]. Since beacons are transmitted in the clear a malicious station can sniff a beacon and replay it after modification of crucial beacon information with the source address spoofed as the MAC address of the Access Point. Specific firmware implementations maintain the time stamp information in the hardware which is updated upon receipt of a Beacon. At present there is no verification of authenticity of the beacon or the sender of the beacon. Thus all the stations will update their time stamps with an incorrect clock due to the malicious beacon.

The solution for forged beacons can include the following:

- Beacons should be accepted only at the correct beacon interval. Beacons received at incorrect intervals should be dropped by legitimate stations.

- A technique to provide non-repudiation and integrity of a beacon needs to be provided by including an HMAC [56]. This is to inform the legitimate stations that the beacon has indeed been sent by the legitimate Access Point. This can be achieved by providing an HMAC within the beacon which can be generated by the legitimate Access Point only. Groupwise keys are established between the Access Point and the various stations in the BSS and this can be used to provide the HMAC which needs to be protected against replays by computing the HMAC of the frame concatenated with the timestamp. This attack will have a temporary impact since a correct beacon from the legitimate Access Point will reset the clock for all stations in the BSS.

#### 4.3.7.3 RTS/CTS NAV Attack

The correct working of the 802.11 MAC protocol depends to a great extent on the correct update of the Network Allocation Vector (NAV) [1] at the various stations. Stations do not contend with each other once the NAV is set for a particular time period. The NAV duration is conveyed to the rest of the stations by Request To Send (RTS) and Clear To Send (CTS) frames as well as the Duration field in other frames. Forgery of RTS/CTS frames with incorrect NAV durations can result in wastage of precious air time during MAC contention.

The solution for RTS/CTS NAV attacks involves providing non-repudiation for RTS and CTS frames. This is to inform the legitimate stations that the RTS/CTS frames have been sent by a legitimate Access Point or Station. This can be achieved by providing an HMAC within the RTS/CTS frame which can be generated by the legitimate Access Point or a legitimate station only. Group wise keys are established between the Access Point and the various stations in the BSS and this can be used to provide the hash secret which needs to be protected against replays by including the timestamp as well. This attack will have a temporary impact since a correct RTS/CTS from the legitimate Access Point or Station will reset the NAV for all stations in the BSS to the correct value. It is important to mention here that there is significant overhead of adding an extra field for the hash in Control Frames. The size of an RTS frame at the MAC layer is 20 octets and the size of a CTS frame is 14 octets. Assuming that the hash generated by any robust cipher will be 16 bytes results in a significant overhead for Control Frames.

#### 4.3.7.4 Probe Request Flood

Probe Requests are used by stations to actively scan the 802.11 WLAN for Access Points [1]. On receipt of a Probe Request from a station the Access Point is required to respond immediately with a Probe Response frame. This can be used as an opportunity for a DDoS attack. The malicious station continuously injects Probe Request frames and the Access Point is obliged to respond with a Probe Response. If the malicious station is able to inject frames at a high rate then the Access Point will devote a good chunk of its valuable processing time in transmitting Probe Responses rather than utilizing it for legitimate traffic.

One of the solutions for Probe Request frames involves a heuristic approach. The Access Point needs to monitor the number of Probe Requests it receives on average in a particular time interval and maintain a threshold. If the number of Probe Requests received by an Access Point crosses the threshold then the Access Point can assume that this is a DDoS attack and deploy countermeasures. The countermeasure involves a halt in responding to Probe Requests by the Access Point for a particular pre-decided time interval.

This might result in Probe Requests from legitimate stations being ignored by the Access Point. However this is not a major problem because the legitimate stations can still utilize passive scan and wait for a beacon from the Access Point it wants to authenticate and associate with. A slight delay in associating with an Access Point will be observed by legitimate stations during deployment of countermeasures. Note that the HMAC solution will not work well in this specific scenario since the Access Point is obliged to respond to Probe Requests from stations which do not have established keys initially.

## 4.4 Routing DDoS attacks

The second type of attacks is specific to routing attacks [14] in 802.11s WMN. Since 802.11s is inherently a multi-hop wireless network [83]. DoS attacks inherent in adhoc multi-hop networks can be easily carried out here. A single malicious station can create only a local impact with these attacks. It is possible only for a large number of malicious stations to create a significant impact on the entire mesh if they have a detailed knowledge of the network topology. The impact will be felt also by stations at the edge of the network which rely primarily on other stations to provide them multi-hop connectivity. Routing and Forwarding techniques might get disrupted locally due to manipulation of HWMP [42].

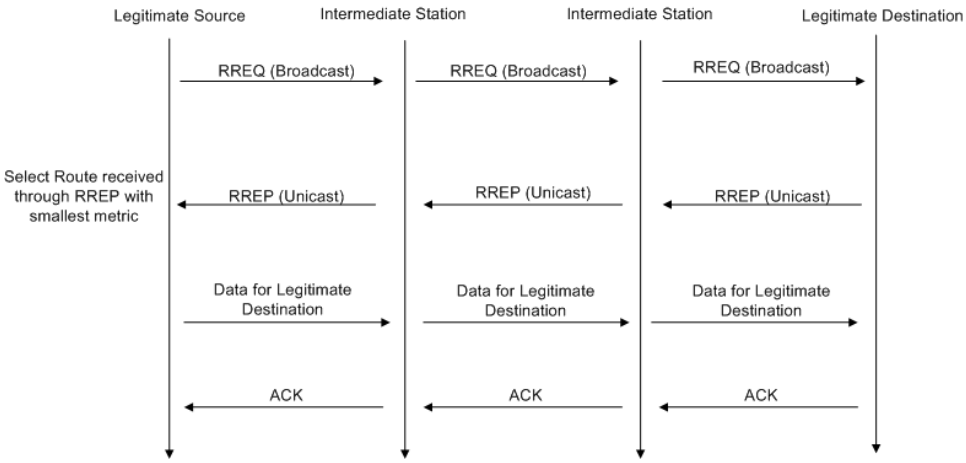


**Figure 4.9.** 802.11s WMN Routing

HWMP relies on RREQ, RREP and RERR messages to detect an optimal route from source to destination as shown in Figure 4.9. The source broadcasts a RREQ message which is received by neighbouring stations and forwarded to the destination. The metric field is updated each time a neighbouring station forwards or rebroadcasts the RREQ message to provide the cumulative metric from source to destination. The destination on receipt of the RREQ message with the best

metric sends an RREP message on the optimal route back to the source. This is a dynamic procedure and the optimal route can be recalculated at regular time intervals. The default routing metrics suggested in 802.11s is described in the formula [44] below where:

- Channel access overhead $O_{ca}$ (PHY dependent).
- Protocol overhead $O_p$ (PHY dependent).
- Number of bits $B_t$ in a test frame (PHY dependent).
- PHY bit rate r.
- Frame error rate $e_{pt}$ for the test frame.

$$c_a = [O_{ca} + O_p + B_t/r] * [1/[1 - e_{pt}]]$$

As an optimization the intermediate neighbouring nodes might send an RREP message back to the source so that the source can begin transmission of unicast frames even before receiving the final RREP message from the destination. This technique can be exploited because the RREQ, RREP and RERR message information elements are transmitted in the clear and are not encrypted since they are encapsulated in 802.11s Action frames.

It is important to note here that the attacks mentioned here are traditional DoS attacks with local impact. However multiple malicious nodes conspiring with each other can perform a DDoS attack with a sophisticated degree of coordination amongst each other along with knowledge of the network topology. Although complete unavailability of the mesh is difficult to achieve. Network partitioning can definitely be the result of such an attack.

The kinds of attacks discussed here include Black Hole attacks [14], Grey Hole attacks and Routing Metric Manipulation attacks.

### 4.4.1 Black Hole DoS Attack

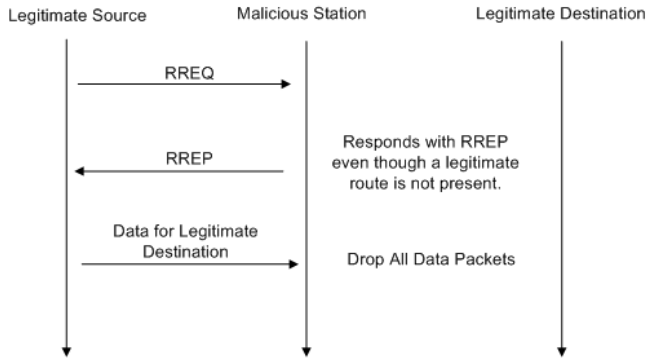


**Figure 4.10.** Black Hole DoS Attack

In Black Hole Attacks as seen in Figure 4.10 the malicious station always responds positively to any RREQ message it receives. Since the frames are not encrypted the source is made to believe that there is a legitimate route from the malicious station. Once the route is established the malicious station drops all packets sent towards it resulting in a DoS attack. As mentioned earlier the impact of a single station performing such an attack will be local.

### 4.4.2 Grey Hole DoS Attack

Grey Hole DoS attacks are similar to Black Hole DoS attacks as seen in Figure 4.11. The malicious station always responds positively to any RREQ message it receives. The only difference in execution is at the final step. The malicious station

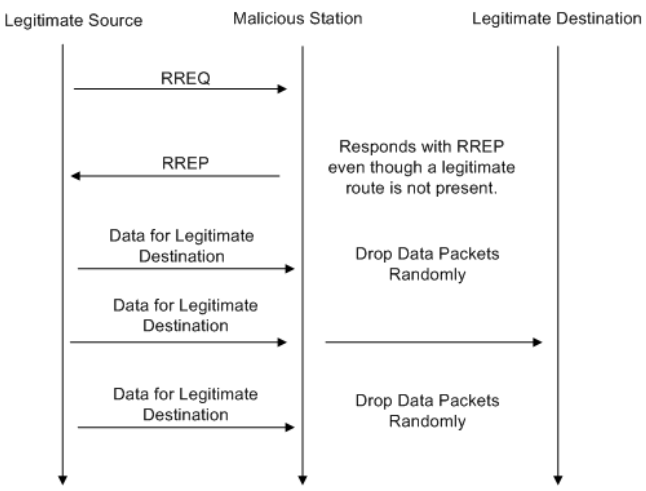


**Figure 4.11.** Grey Hole DoS Attack

randomly drops packets thereby making it difficult for the source to detect the DoS attack quickly. The dynamic nature of the wireless medium might encourage the source to continuously retry packets on this route attributing the intermittent packet losses to collisions or link errors.

### 4.4.3 Metric Manipulation DoS Attack

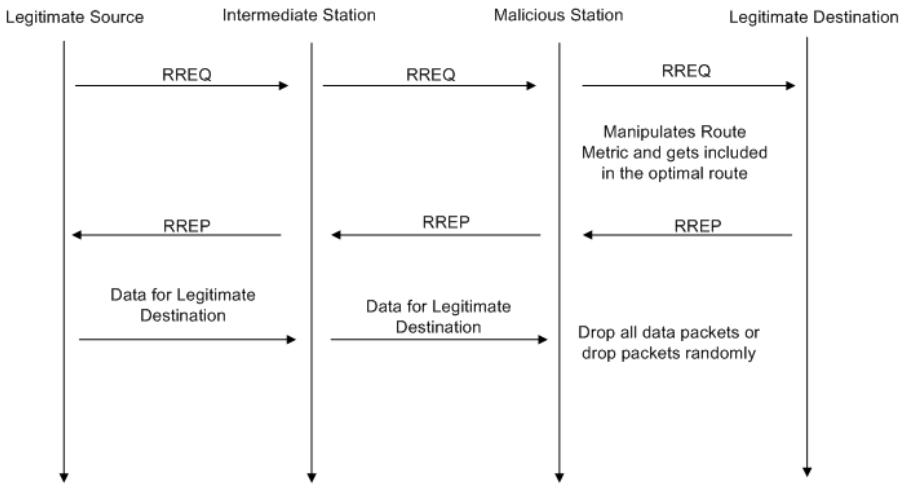


**Figure 4.12.** Metric Manipulation DoS Attack

The metric manipulation DoS attack is seen in Figure 4.12. The metric used to decide an optimal route in 802.11s is Bit Error Rate and Frame Error Rate. This can be updated by intermediate stations on the route dynamically. A malicious station can manipulate the metric to ensure that it is always selected as an intermediate route whenever it receives an RREQ message. Once the route is established the malicious station can either perform the Black Hole or Grey Hole DoS attacks.

### 4.4.4 Solution for Routing DDoS Attacks

Routing DDoS attacks can be prevented by ensuring the authenticity and integrity of RREQ, RREP and RERR messages. Cryptographic techniques to do so have been discussed by Perrig et al in [69]. The IEEE group for Protected Management Frames i.e 802.11w [29] is working towards the security of Management, Control and Action frames. This group is responsible for designing a specification which will ensure authenticity and integrity of management, control and action frames. 802.11w is expected to be released as a specification towards the end of 2009. Routing DDoS attacks will prove to be a threat to 802.11s unless this specification is deployed effectively in 802.11s devices.

Protection of management, control and action frames involves providing non-repudiation and integrity of these frames. Symmetric cryptographic keys are shared between each of the devices in a particular wireless link. It is feasible to provide a HMAC [56] on the concatenation of the following:

- Original Management, Control or Action Frame.
- Source MAC Address.
- Timestamp (Third octet).

The HMAC can be computed by using MD5 or SHA-1. This will result in an increase of packet length and affect the overall performance of the WMN. However it is required especially for the wireless links between the Mesh Relays and the Mesh Portal Points. The hash of the frame will ensure non-repudiation and integrity of the frame. Inclusion of the timestamp will ensure that this technique is not subject to replay attacks by malicious stations.

## 4.5 DDoS attacks on Mesh Portal Point

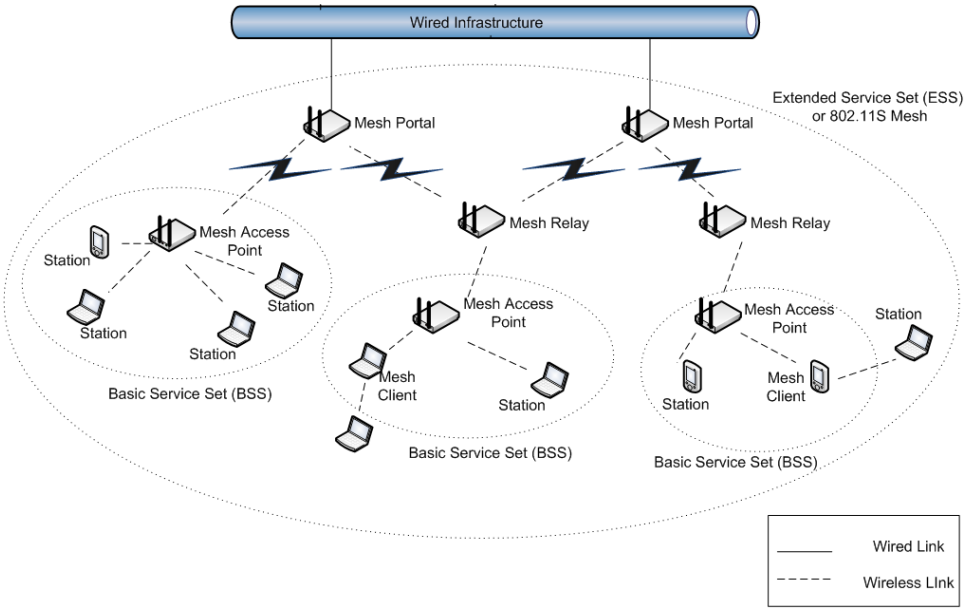


**Figure 4.13.** DDoS attack on Mesh Portal Point

The third type of attacks is specific to Mesh Portal Points [45] in 802.11s WMN's. 802.11s introduces new classes of devices including Mesh Relays which primarily forward packets over wireless links and Mesh Portal Points which are responsible for providing the back end wired connectivity to the entire WMN. The Mesh Portal Points, Mesh Relays and Mesh Access Points are in decreasing order of the level of criticality to the overall optimal performance of a WMN. It is of

importance to notice that disruption of services at Mesh Portal Points can result in degradation or complete unavailability of the entire WMN as seen in Figure 4.13. DoS attacks on Mesh Relays can result in partitioning of the network leading to a bottleneck at a single Mesh Portal Point. Features like load balancing [45] among Mesh Portal Points will be ineffective if the Mesh Relays can be rendered ineffective. The impact of an attack on Mesh Relays depends largely on the topology of the WMN.

Hence the focus will be on securing the Mesh Portal Points as they can be considered to be the back bone of the WMN. It is intuitive to see that the security of the Mesh Portal Points is of significant importance to the correct functioning of the entire mesh. The DDoS detection techniques at the Mesh Portals need to be more stringent than any other types of mesh device.

### 4.5.1 Detection of Mesh Portal Point

The first step by the attacker is to detect these Mesh Portal Points. It is fairly simple to detect the Mesh Portal Points if the attacker has complete knowledge of the network topology. In a real scenario this is not feasible. An attacker can detect the Mesh Portal Points without having knowledge of the network topology as discussed below.

**Table 4.1.** Fields in Portal Announcement Frame

| Element | Description |
|---|---|
| PANN Identifier | Unique Identifier for a PANN message. |
| Portal Address | Mesh Portal Point MAC Address. |
| Metric | Cumulative metric from originator to current receiver. |
| Hop Count | The number of hops from Originator to current receiver. |
| Time to Live | Maximum number of hops allowed for this Frame. |

802.11s provides a Mesh Portal Point Announcement Protocol to provide information to other devices in the Mesh about its presence. A Portal Announcement Information Element is used for broadcasting in the mesh about the presence of a Mesh Portal Point. This information element can be transmitted in a management frame along with other information elements e.g. Beacons or Probe Responses. A message containing a Portal Announcement message is known as a Portal Announcement (PANN) message [45].

Some of the information contained in the PANN message are shown in Table 4.1. The fields of importance to this discussion are the first three fields. It is important to remember that the mandatory attacker capabilities include the ability to operate in promiscuous mode and sniff 802.11 WLAN frames. The attacker can hence first detect that it has received a message from a Mesh Portal Point by seeing the unique PANN identifier. At this point the attacker also knows the MAC Address of the Mesh Portal. The attacker will know that it is in range of a Mesh Portal Point if it receives a beacon or can detect a route to the Mesh Portal Point by sniffing RREQ/RREP action frame messages. Once a Mesh Portal Point is identified the various DoS attacks discussed earlier can be used to disable services at the Mesh Portal Point. A few mobile malicious devices can possibly detect and attack all the Mesh Portal Points simultaneously resulting in complete unavailability of the mesh. This coordinated attack will have a higher impact with lesser number of resources compared to an Intra-BSS DoS attack which requires a malicious station in every BSS of the Mesh to co-ordinate with each other in order to result in a DDoS attack which can provide complete unavailability of services in the Mesh.

### 4.5.2 Attacks on Mesh Portal Point

#### 4.5.2.1 Metric Manipulation Routing DDoS Attack

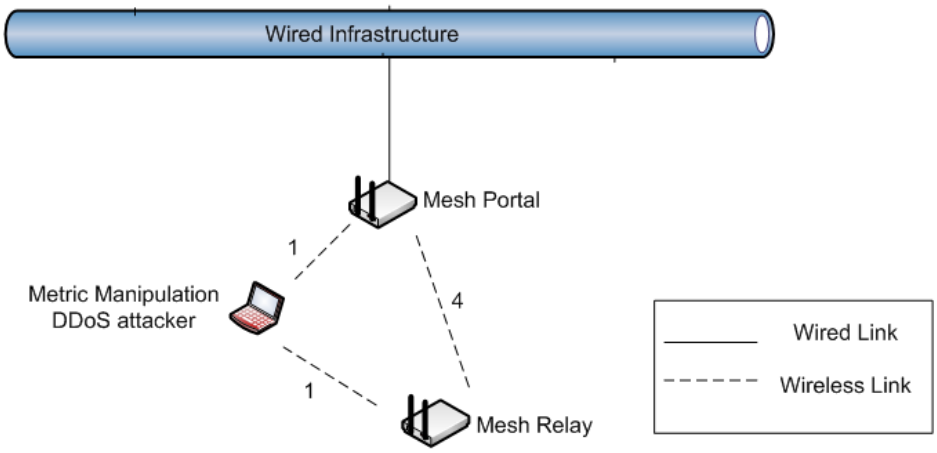


**Figure 4.14.** Metric Manipulation Routing Attack on Mesh Portal Point

The Mesh Portal Point is the best target for a Metric Manipulation Routing DDoS attack. This is shown in Figure 4.14. Assume that the metric from the Mesh Portal Point to the Mesh Relay is 4 units. Also assume that the DDoS attacker capable of metric manipulation modifies the RREQ messages to show that the metric from the Mesh Portal to the Mesh Relay via the malicious attacker is 2 units. The route chosen from the Mesh Portal to the Mesh Relay includes the malicious station. Once the route is established the malicious station can drop all packets resulting in an attack which will be detected quickly. However if it drops packets selected at random then the attack will be difficult to detect for a longer period of time.

The solution for the Metric Manipulation Routing Attacks along with other routing attacks have been discussed earlier in Section 3.4.4 . These attacks are more difficult to detect if they are insider attacks where the insider shares legitimate

keys with the Mesh Portal Point. If the attacker does not share keys with the Mesh Portal Point then a route will not be established through it since it does not belong to the legitimate stations list. The attacker will belong to State 1 of the 802.11 State Machine which does not allow transmission of data frames to such stations.

#### 4.5.2.2 Route Disruption DDoS Attack

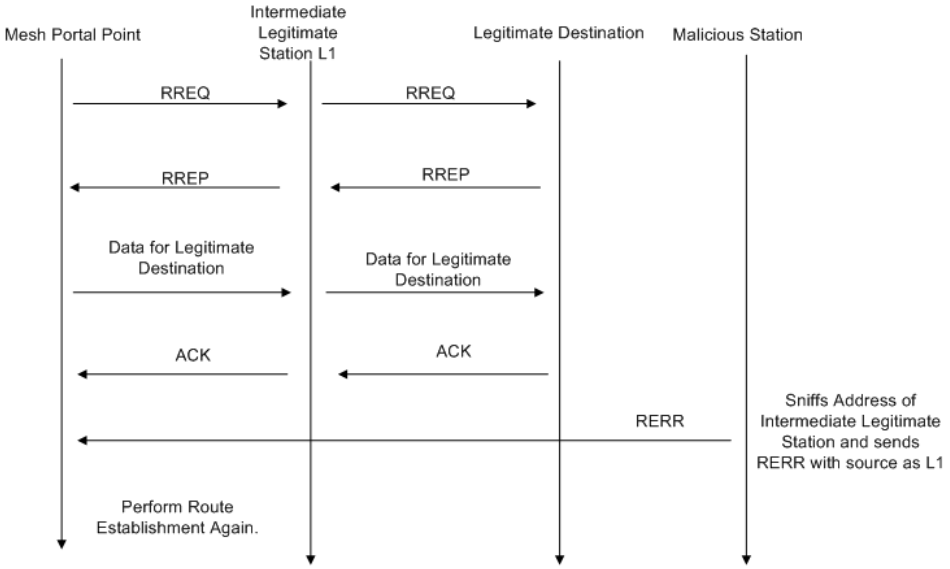


**Figure 4.15.** Route Disruption DDoS Attack

The RERR information element is used to inform all peers about a broken link which had been previously established by a RREQ/RREP message exchange. It is important to note that these information elements are not encrypted since they are included in Action frames. The attack is shown in Figure 4.15. It is assumed that the malicious station is within range of the Mesh Portal Point. Once a Mesh Portal Point is identified the attacker needs to promiscuously sniff all 802.11 frames to detect RREQ/RREP messages. Once a route is established for the Mesh Portal

the attacker can spoof the MAC address of the next device on the route which is one hop away from the Mesh Portal Point and transmit an action frame with an RERR information element notifying the Mesh Portal Point about a broken link. Even though the route is still active the Mesh Portal Point is made to believe that the route is broken and it has to perform the RREQ/RREP message exchanges again. This can be done repeatedly to provide complete unavailability at the Mesh Portal Point.

#### 4.5.2.3 Solutions for Route Disruption DDoS Attack

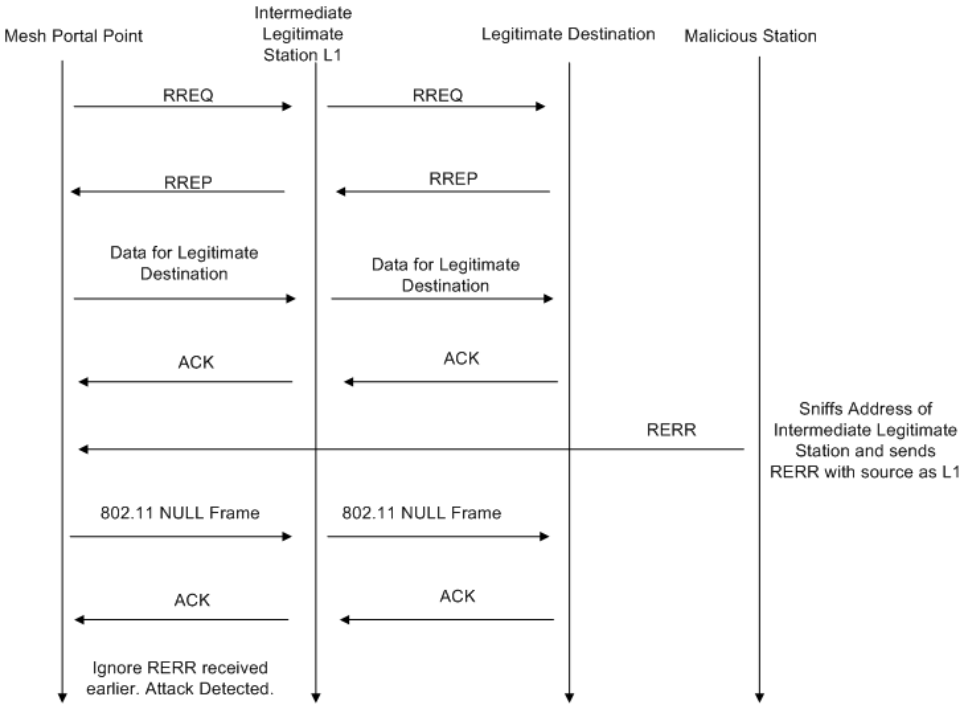


**Figure 4.16.** Solution for Route Disruption DDoS Attack

The Route Disruption attack can be prevented by deployment of 802.11w as discussed earlier. This attack can also be avoided as shown in Figure 4.16 by allowing the transmission of a single 802.11 NULL frame along the established route from which a RERR message had just been received. It is assumed that

the malicious station is within range of the Mesh Portal Point. If an ACK is received from the established route then the route establishment by RREQ/RREP messages is not required and the RERR message can be ignored. This will be the case because the legitimate station will still generate the ACK on the optimal route and the attacker cannot suppress this ACK due to the nature of the wireless medium. If an ACK is not received then the RERR message is genuine and the route establishment steps involving RREQ/RREP need to be performed.

#### 4.5.2.4 Jamming Attacks

Mesh Portal Points are good candidates for carrying out a Physical Layer Jamming Attack [17] due to the criticality of Mesh Portal Points to the overall functionality of the mesh. Physical layer jamming attacks are expensive to carry out for a significant amount of time due to the fairly sophisticated resources required. A high transmission power along with a high packet injection rate is required for such attacks to be successful over long periods of time. In most implementations the physical layer is not a software developed layer. It requires a significantly high number of resources to manipulate or tamper the hardware in order to carry out a physical layer attack. However it is possible to completely jam the wireless network if we assume a strong attack model. This is similar to The Physical Layer Jamming Attacks. The MAC hardware is manipulated so that 802.11 MAC headers are continuously transmitted at a high injection rate. Link Layer Jamming Attacks [18] are easier to implement compared to Physical Layer Jamming Attacks. The impact of a Link Layer Jamming Attack is similar to the Physical Layer Jamming Attacks although this is highly dependent on the injection rate of the headers. The solutions for Link Layer Jamming attacks are similar to the Physical Layer Jamming attacks discussed in the previous section.

Such attacks are difficult to prevent. However certain techniques to make it significantly difficult for attackers to succeed can be deployed. 802.11 utilizes a single physical channel at a time. The solution for Physical Layer Jamming Techniques involves the following:

- Mesh Portal Points should be able to detect physical layer activity on multiple channels simultaneously. Profiling of multiple channels simultaneously needs to be performed as a countermeasure whenever the overall throughput drops below a certain predefined threshold.

- Once the profiling is done, the Mesh Portal Point should be able to modify its channel by only informing its single hop peers through an encrypted channel. The RTX/CTX technique can be used by the Mesh Portal Points. The Mesh Portal Points sends an encrypted RTX frame informing its peers that it is modifying its channel and the peers respond by sending a CTX once they have performed the required bookkeeping and management to modify their current channel. The other members in the mesh more than a single hop away from the Mesh Portal Point can continue at their current operating channel.

- Jamming attacks can also be avoided if the administrators can detect the location of the source of the attacker in a WLAN as discussed in [84].

The attacker can however detect this change in activity from one channel to another and continue the attack in the new channel. It is needless to say that if the attacker is able to jam all channels continuously then it will automatically succeed in providing complete unavailability. Hopping channels continuously might result in some amount of useful throughput being transmitted in bursts.

Chapter 5

# Simulation Setup and Results

## 5.1 Simulation Setup

**Table 5.1.** GTNetS Features

| Features | Description |
|---|---|
| Layer 2 Protocols | IEEE 802.3 and IEEE 802.11 protocols. |
| Layer 3 Protocols | IPv4. |
| TCP | Tahoe, Reno, NewReno, and SACK. |
| Routing | Static, on-demand using the Nix Vector approach etc. |
| Applications | Examples include FTP models, UDP based models etc. |
| Links | Point-to-Point, Ethernet, and Wireless links. |
| Queuing | Drop-tail, Random Early Detection ( RED) etc. |
| Animation | Graphical viewing of the simulation topology. |
| Topologies | Star, Tree, Dumbbell, and Grid. |
| Node Mobility | Random way point and specific way point models. |
| Packet Tracing | Tracing can be enabled or disabled by node, protocols etc. |

The Georgia Tech Network Simulator (GTNetS) [16] is a network simulation environment with accurate implementation of every layer in the protocol stack. The important features of GTNetS are listed in Table 5.1. Packets in GTNetS consist of a list of Protocol Data Units (PDU's) that are appended and removed

from the packet as it moves through the protocol stack. Modifications to the simulation environment were done only in the IEEE 802.11 MAC layer. Additional features implemented in the IEEE 802.11 MAC layer are listed below:

- IEEE 802.11 Power Save functionality was implemented including the following
  - Queues for buffering data per station at the Access Point.
  - TIM field updates in the beacon frame based on buffered data.
  - PS POLL transmission at the station and reception at the Access Point.
  - Update of More Data bit by the Access Point and Power Management bit by the Station in the Frame Control field of the 802.11 header.
  - Every station is provided an unique Association Identifier (AID) at the Access Point.
  - A new device type BSS_POWER_SAVE is defined corresponding to 802.11 Stations which want to utilize 802.11 Power Save features.
  - Additional statistics specific to Power Save.
- Data Structure to hold information about each Station at the Access Point.
- Shared Key Authentication handshake between Station and Access Point using GNU's basic cryptography library called Libgcrypt [85].
- Transmission and Reception of Deauthentication and Disassociation frames.
- IEEE 802.11 State Machine implementation. Data frames are dropped if the Station is in State 1 i.e unauthenticated and unassociated as well as in State 2 i.e unassociated.

- Solutions to prevent Intra-BSS DDoS attacks discussed in Chapter 4.
- Injection of PS POLL, Deauthentication, Disassociation and Routing frames.
- Malicious activity by nodes for either dropping all frames or dropping frames randomly.

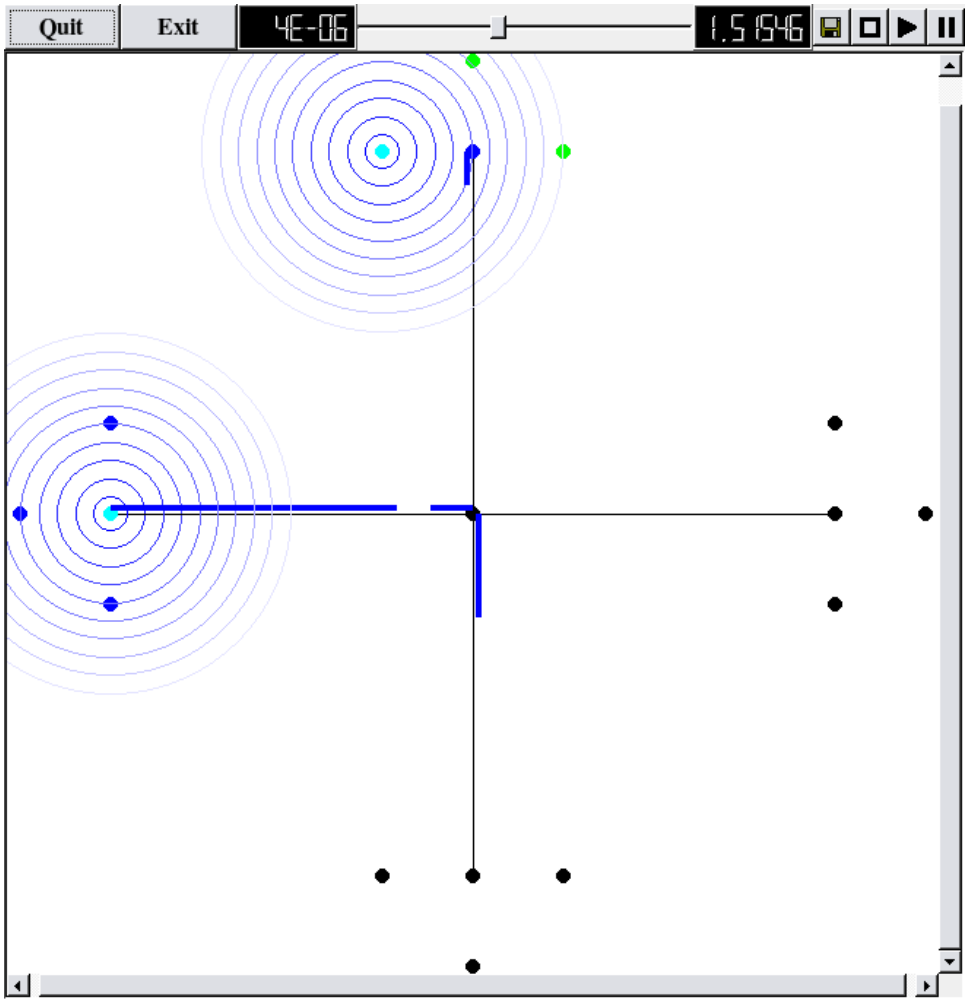


**Figure 5.1.** GTNetS Infrastructure Topology Example

The GTNets simulator has been developed using an object oriented design. Each network element is a C++ object. The network elements are instantiated in a C++ main test program. Figure 5.1 shows a sample network topology created

using GtNetS. The figure shows four 802.11 BSS with an Access Point and three stations each. The Access Points are connected to each other via a wired back end. It is fairly simple to define different topologies for the network along with specifying transmission range, location and mobility of wireless nodes.

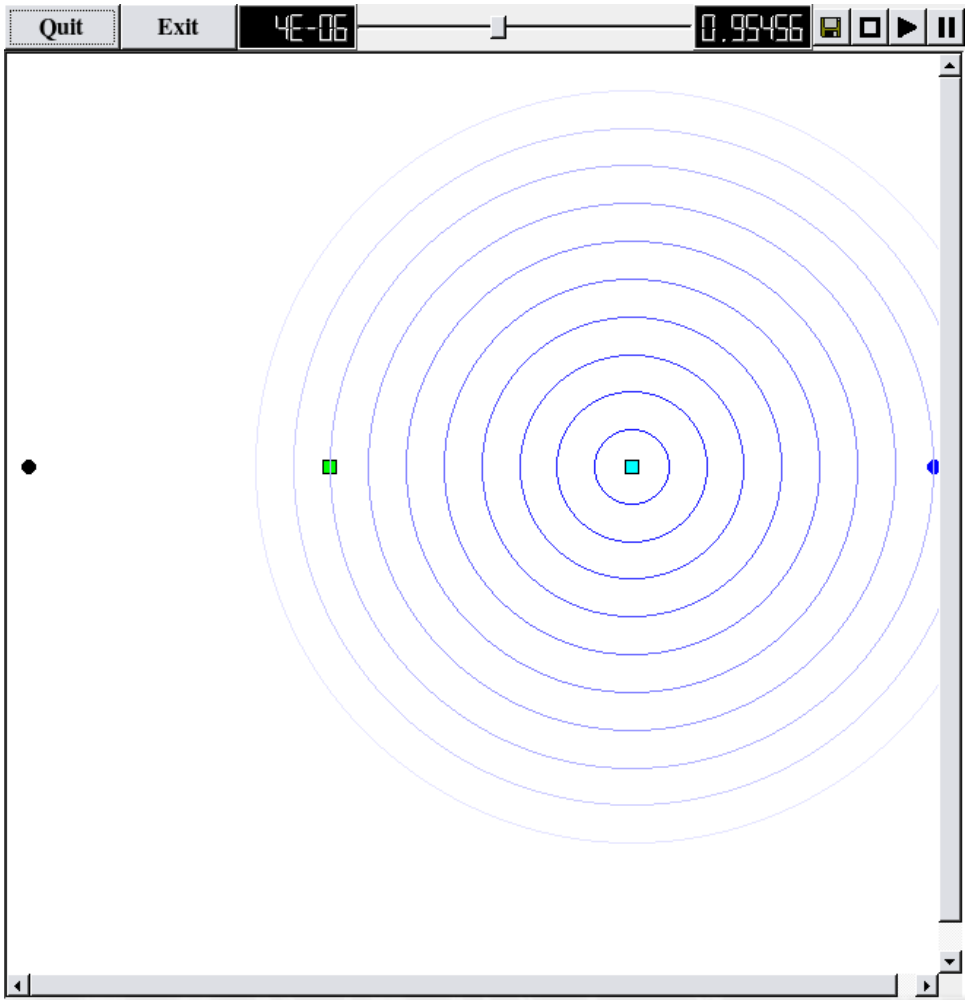


**Figure 5.2.** GTNetS Multi-Hop Adhoc Topology Example

Figure 5.2 shows an example of a GTNetS simulation setup in a multi-hop wireless setting. The routes are established using RREQ, RREP and RERR messages as in HWMP in 802.11s. Data is transmitted from the left most node in the figure to the right most node over multiple hops. Mobility, location, transmission

range, data source (i.e UDP, TCP etc) can be specified. Route computation is performed again if a route fails for a particular time period.

## 5.2 Simulation Results

The results of the simulation experiments are detailed in the following sections.

### 5.2.1 Intra-BSS DDoS Attacks

#### 5.2.1.1 Shared Key DDoS Attack

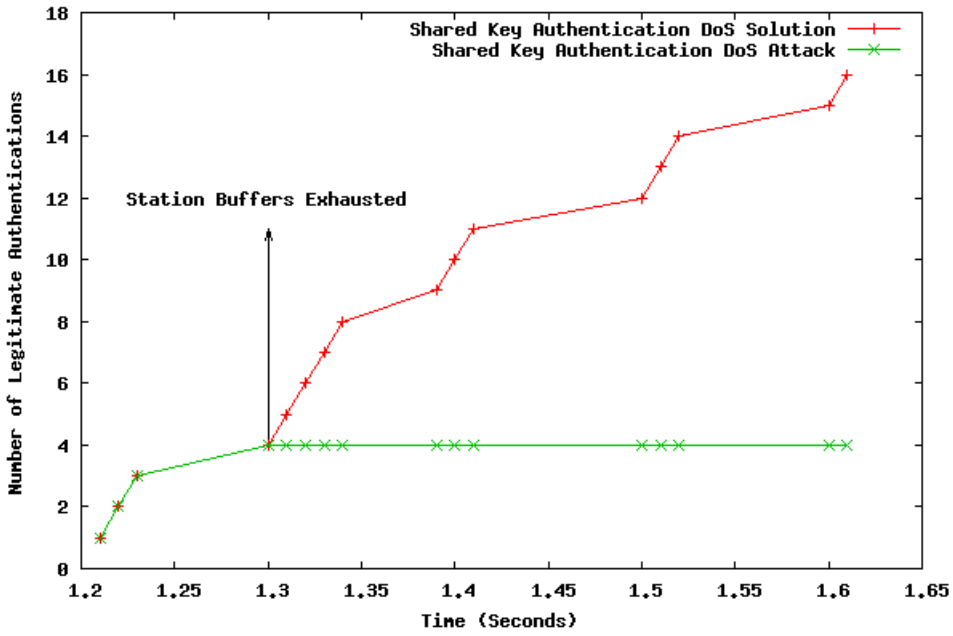


**Figure 5.3.** Shared Key Authentication DDoS Attack Results

The Shared Key Authentication DDoS Sybil attack has been carried out with and without the solution described in Chapter 4. The attack was carried out in four BSS each having a single Access Point and four stations. The results are displayed in Figure 5.3. The results show that the Shared Key DoS attack is successful

in each BSS after the addition of only a single legitimate station. The number of station buffers supported at the Access Point get exhausted at time t = 1.3 seconds in the entire network. Subsequently no new legitimate stations are able to join the BSS. However the solution of utilizing the Challenge Text to offload state to the station from the Access Point prevents this attack i.e., Legitimate stations are able to join the BSS without being affected by the malicious sybil activity.

#### 5.2.1.2 Deauthentication DDoS Attack

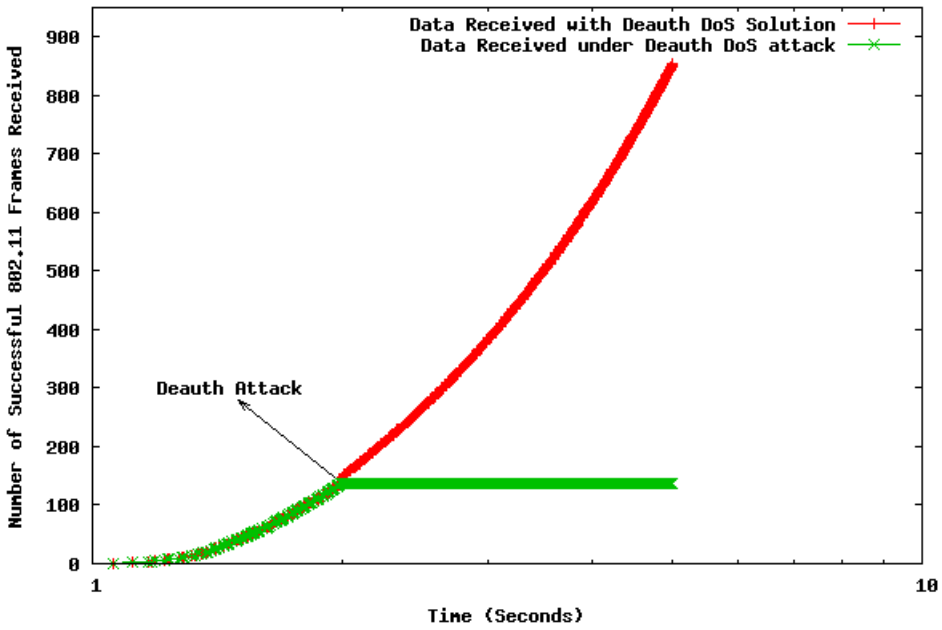


**Figure 5.4.** Deauthentication DDoS Attack Results

The Deauthentication DDoS attack has been carried out with and without the solution described in Chapter 4. The attack was carried out in four BSS each having a single Access Point and three stations. The results are displayed in Figure 5.4. UDP traffic was generated for this experiment. The DDoS attack was coordinated by the malicious stations to begin at time t= 2 seconds after the start of the simulation. The results show that without the solution the network

throughput is severely limited because the stations are not aware that they have been deauthenticated from the Access Point. However the approach of sending and validating the Deauthentication frame by sending an 802.11 NULL frame to the source station foils the DDoS attack. There is no impact on the overall network throughput if the solution is incorporated in the implementation. This works well for the weak attack model. The malicious stations have the capability to forge ACK frames as well in the strong attack model. However there is no reason for the malicious station to send an ACK since that will only result in the Access Point believing that the Deauthentication frame sent earlier was malicious.

#### 5.2.1.3 Disassociation DDoS Attack

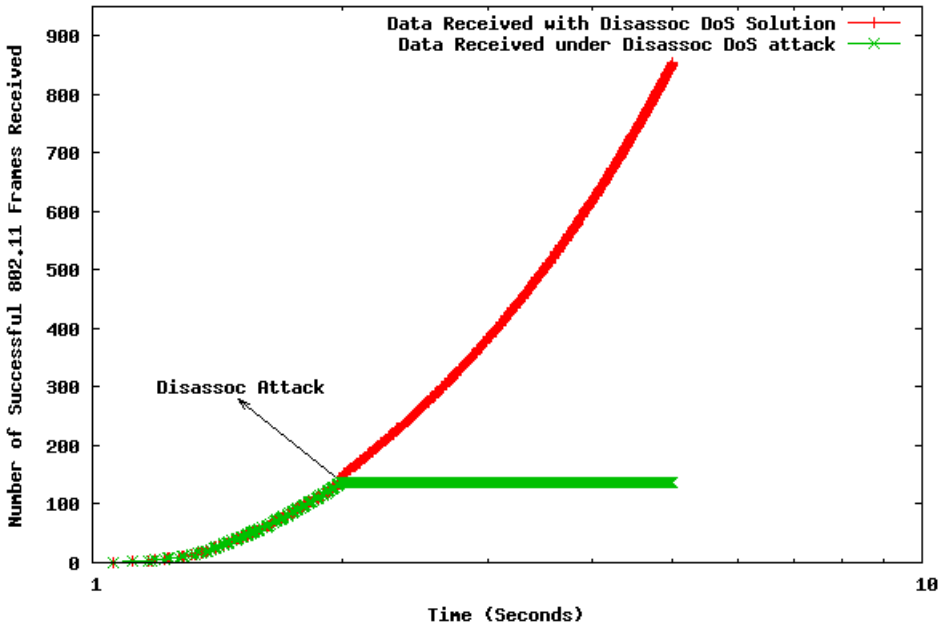


**Figure 5.5.** Disassociation DDoS Attack Results

The Disassociation attack has been carried out with and without the defense described in Chapter 4. The attack was carried out in four BSS each having a single Access Point and three stations. The results are displayed in Figure 5.5. UDP

traffic was generated for this experiment. The DDoS attack was coordinated by the malicious stations to begin at time t= 2 seconds after the start of the simulation. The results show that without the solution the network throughput comes to a complete stop because the stations are not aware that they have been disassociated from the Access Point. However if the solution of validating the Disassociation frame by sending an 802.11 NULL frame to the source station results in a failure of the DDoS attack. There is no impact on the overall network throughput if the defense is incorporated in the implementation. This works well for the weak attack model. The malicious stations have the capability to forge ACK frames as well in the strong attack model. However there is no reason for the malicious station to send an ACK since that will only result in the Access Point believing that the Disassociation frame sent earlier was malicious.

#### 5.2.1.4 Power Save DDoS Attack

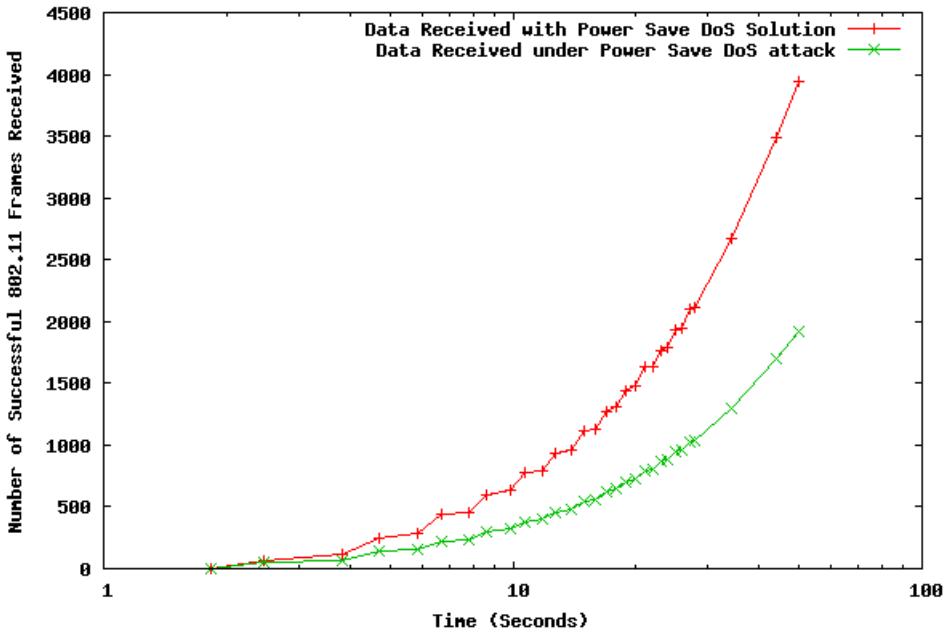


**Figure 5.6.** Power Save DDoS Attack Results

The DDoS attack on stations in Power Save has been carried out with and without the defense described in Chapter 4. The attack was carried out in four BSS each having a single Access Point and a single power-saving station each. The results are displayed in Figure 5.6. UDP background traffic was generated for this experiment. The DDoS attack was coordinated by the malicious stations to begin at time t= 1.35 seconds after the start of the simulation. The results show that without the solution the network throughput falls rapidly because the packets buffered at the Access Point for the station in power save get dropped. However the solution of validating the first PS POLL frame by sending an 802.11 NULL frame to the source station prevents the DDoS attack. There is a slight impact on the overall network throughput if the solution is incorporated in the implementation due to the overhead of transmitting an 802.11 NULL frame each listen interval.

This works well for the weak attack model. The malicious stations have the capability to forge ACK frames as well in the strong attack model. This is a problem for this solution since the Access Point on reception of the ACK frame will believe that the legitimate station is awake and transmit the buffered data for this station which will eventually get dropped since the station is actually sleeping. In order to successfully counter DDoS attacks against stations in power-save it will be required to cryptographically verify if the sender of the PS POLL frame is a legitimate part of the BSS. The HMAC solution discussed earlier is robust to the DoS attacks resulting from a strong attack model. Enhancements suggested in 802.11w or the Protected Management frames group will have to be incorporated.

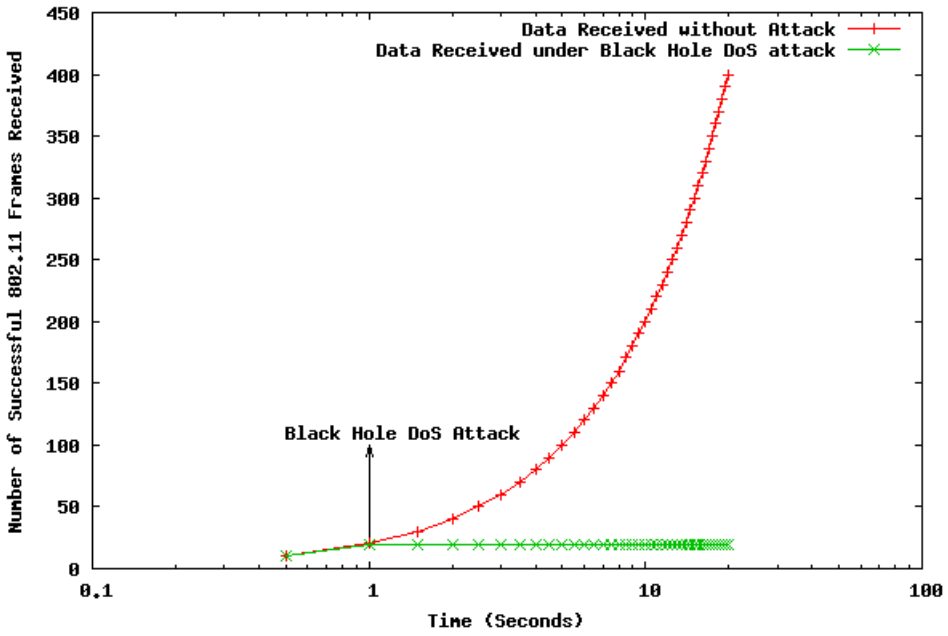


**Figure 5.7.** Black Hole DDoS Attack Results

## 5.2.2 Routing DDoS Attacks

### 5.2.2.1 Black Hole DDoS Attack

The routing DDoS attacks required a different approach compared to the earlier Intra-BSS DDoS attacks. The simulation setup comprised of four wireless devices within wireless range of only one other station. This provided the multi-hop wireless feature similar to Adhoc networks. 802.11s is inherently a combination of 802.11 Infrastructure WLAN and 802.11 ad hoc Wireless Links along with Mesh Portal Points being attached to a wired back end. The experimental results are displayed in Figure 5.7. The results confirm that without the attack the number of 802.11 frames received at the legitimate station increases at a fairly constant rate. However with the Black Hole DoS attack, it is seen that the throughput on this route drops completely since the malicious station in between drops all frames it receives for the legitimate destination.

These results confirm the requirement that RREP/RREQ/RERR messages in the HWMP protocol need to be protected. These action frames are currently sent out in the clear. Malicious stations can drop or modify these frames at will thereby affecting the performance of the entire mesh. This attack can be converted to a DDoS attack by co-ordination among multiple malicious stations in the mesh. The real key is to ensure that the RREP/RREQ messages are not tampered with. Each station in the Mesh should be able to verify that the source of the frame is a legitimate part of the mesh.

#### 5.2.2.2 Grey Hole DDoS Attack

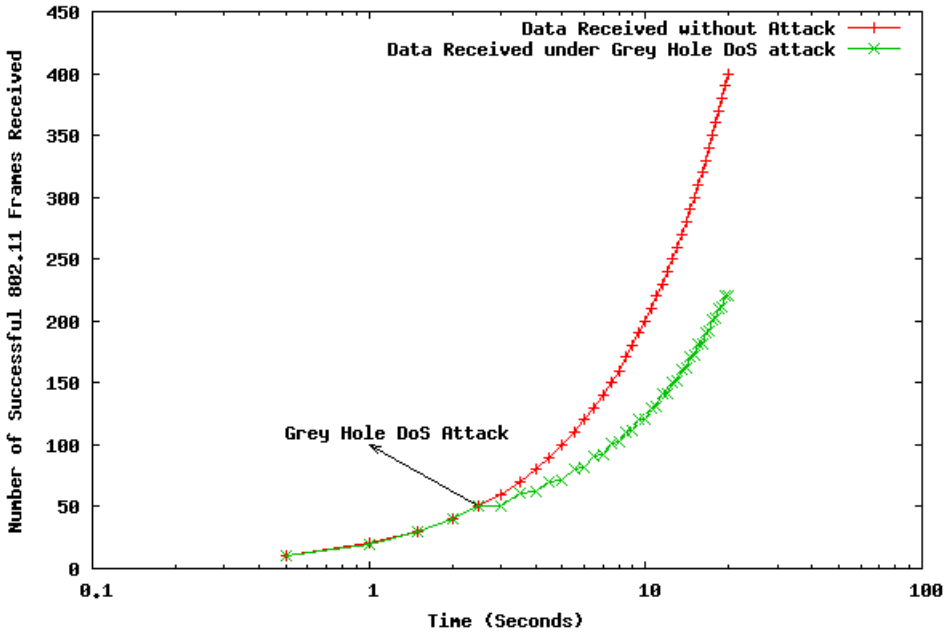


**Figure 5.8.** Grey Hole DDoS Attack Results

The experimental results displayed in Figure 5.8 confirm that without the attack the number of 802.11 frames received at the legitimate station increases at a fairly constant rate. However with the Grey Hole DoS attack it is seen that the throughput on this route drops at a constant rate since the malicious station in

between selectively drops frames it receives for the legitimate destination. This attack is much more difficult to detect compared to the Black Hole DDoS attack since the malicious station drops frames randomly. The source station is made to believe that the random losses are due to wireless link errors, collisions and congestion at devices which are multiple hops away from it.

These results confirm the requirement to protect the RREP/RREQ/RERR messages in the HWMP protocol. These action frames are currently sent out in the clear. Malicious stations can drop or modify these frames at will thereby affecting the performance of the entire mesh. This attack can be converted to a DDoS attack by co-ordination among multiple malicious stations in the mesh. The real key is to ensure that the RREP/RREQ messages are not tampered with. Each station in the mesh should be able to verify that the source of the frame is a legitimate part of the mesh.

### 5.2.3 DDoS Attacks against Mesh Portal Point

Insider attacks at the Mesh Portal Points can result in Black Hole and Grey Hole DDoS attacks as shown in Figure 5.7 and Figure 5.8 respectively. However the impact of such an attack will result in complete unavailability of Internet connectivity or network partitioning resulting in significant load increase at Mesh Portal Points which are already bottlenecks for the entire WMN. Possible upgrades to IEEE 802.11n [26] for higher throughputs in the range of a theoretically possible 600 Mbps at the Mesh Portal Points will result in a devastating impact on the performance of a WMN.

**Table 5.2.** Implementation Overhead - Frame Size

| Frame Type | Size(octets) | Time Critical | Overhead | Modify HW |
|---|---|---|---|---|
| Deauth | 30 | No | 53% | No |
| Disassoc | 30 | No | 53% | No |
| RTS | 20 | Yes | 80% | Yes |
| CTS | 14 | Yes | 114% | Yes |
| PS POLL | 20 | No | 80% | No |
| RREQ | 100+ | No | 16% | No |
| RREP | 100+ | No | 16% | No |
| RERR | 100+ | No | 16% | No |
| Beacon | 76+ | No | 21% | No |
| PANN | 93 | No | 17% | No |

## 5.3 Implementation Overhead

The implementation overhead of using a 16 octet HMAC is shown in Table 5.2. The HMAC is computed with the original frame concatenated with the source MAC address and the third octet of the current timestamp. It is fairly expensive to provide protection for small sized Control frames like RTS and CTS. The overhead for Deauthentication, Disassociation and PS POLL can also be avoided by using the IEEE 802.11 NULL frame approach. It is however advisable to protect the other frames. This approach will ensure non-repudiation and integrity of the frames which are critical in order to provide robustness against DDoS attacks. These changes are easy to implement in all cases except control frames where modification of the 802.11 hardware components in a WLAN solution might be required.

Chapter 6

# Security Requirements of WMN's

The experimental results emphasize the importance of security in WMN's. The impact of DDoS attacks on WMN's can result in complete unavailability or significant degradation of the overall performance. Based on the results in this thesis the security requirements for WMN's are listed below in the following categories.

- Intra-BSS DDoS Attacks.
- Routing DDoS Attacks.
- DDoS Attacks on Mesh Portal Points.

## 6.1 Intra-BSS DDoS Attacks

The following rules need to be followed while designing and implementing 802.11 WMN solutions in order to prevent Intra-BSS DDoS attacks:

- It is fairly simple to modify the MAC address of an 802.11 device. For example in Linux [73] this can be done by using the commands below:

```
sudo ifconfig wlan0 down
```

```
sudo ifconfig wlan0 hw ether 00:A0:B0:C0:D0:E0
sudo ifconfig wlan0 up
```

The ability to change the unique MAC address should be disabled in all firmware for 802.11 WLAN stations and access points. The MAC address is a unique world wide identifier for a device. This ability to modify the MAC address results in malicious devices being able to carry out Sybil and Masquerading DoS attacks by first sniffing the MAC address of legitimate stations and then spoofing their MAC address.

- The central problem faced by IEEE 802.11 WLAN's and WMN's is the lack of security for management, control and action frames. These frames either need to be encrypted or a cryptographic technique to verify whether the source is legitimate should be provided. The frames of particular importance are listed below:
  - Management Frames
    - Deauthentication - Masquerading Attacks.
    - Disassociation - Masquerading Attacks.
    - Beacon - Frame Modification Attacks.
    - ATIM - Masquerading Attacks.
  - Control Frames
    - PS POLL - Masquerading Attacks.
    - Trigger Frames. (APSD) - Masquerading Attacks.
    - RTS - NAV Attacks.
    - CTS - NAV Attacks.

- The 802.11 State Machine should be strictly implemented and adhered to. It is important for Access Points not to accept Data Frames from stations in State 1 (i.e. unauthenticated and unassociated) and State 2 (i.e. unassociated).

- The ability to detect that frames like Deauthentication, Disassociation and PS POLL have been transmitted by a legitimate source is necessary. This can be performed by either of two techniques. Either the solution of sending 802.11 NULL frames should be used or cryptographic techniques of including an HMAC of the frame concatenated with source MAC Address and a timestamp should be used.

- The defense of sending an 802.11 NULL frame before removing a station from the list at the Access Point should be incorporated if the cryptographic techniques are not implemented.

- The defense of sending an 802.11 NULL frame before actually transmitting buffered data at the Access Point should be incorporated if the cryptographic techniques are not implemented..

- Access Points should use the 802.1X hand shake for authentication of station. Open System and Shared Key Authentication mechanisms should be deprecated over the next three to five years. The solution for preventing Sybil Attacks using the cookie mechanism for offloading state in the Shared Key Authentication handshake should be incorporated till such time.

## 6.2 Ad hoc Routing Attacks

The following rules need to be followed while designing and implementing 802.11 WMN solutions in order to prevent Ad hoc Routing attacks:

- The central problem faced by IEEE 802.11 WLAN's and WMN's is the lack of security for management, control and action frames. These frames either need to be encrypted or a cryptographic technique to allow verification about whether the source is legitimate or not, should be provided. The cryptographic technique involves including an HMAC of the frame concatenated with source MAC Address and a timestamp. The frames of particular importance are listed below:
  - Action Frames
    - RREQ (HWMP) - Routing Attacks.
    - RREP (HWMP) - Routing Attacks.
    - RERR (HWMP)- Routing Attacks.
- The timeout on route disruption should be small to ensure quick recovery to an alternate route in case of Black Hole Attacks.
- The integrity of frames like RREQ, RREP and RERR need to be ensured so that a malicious station cannot modify the contents of these frames leading to a Metric Manipulation Routing Attack. Grey Hole and Black Hole attacks can be completely avoided in this manner as well.
- Insider Grey Hole Attacks will still be difficult to detect and is an open research problem.

## 6.3 Attacks on the Mesh Portal Point

The following rules need to be followed while designing and implementing 802.11 WMN solutions in order to prevent DDoS attacks on the Mesh Portal Point:

- The central problem faced by IEEE 802.11 WLAN's and WMN's is the lack of security for management, control and action frames. These frames either need to be encrypted or a cryptographic technique to allow verification about whether the source is legitimate or not, should be provided. The cryptographic technique involves including an HMAC of the frame concatenated with source MAC Address and a timestamp. The frames of particular importance are listed below:
  - Management Frames
    - Mesh Portal Announcement (PANN) - Mesh Portal Attacks.
    - Mesh Root Announcement (RANN) - Mesh Portal Attacks.
  - Action Frames
    - RREQ (HWMP) - Routing Attacks.
    - RREP (HWMP) - Routing Attacks.
    - RERR (HWMP)- Routing Attacks.
- The ability to detect the presence of a Mesh Portal Point needs to be obscured to stations which are not a legitimate part of the network.
- Implementation of additional features is required at the Mesh Portal Point considering its importance to the overall performance of the mesh. The Mesh Portal Point must be able to detect congestion, Link Layer and Physical

Layer jamming attacks on a single channel. Once detected the Mesh Portal Point needs to modify its current channel using the RTX/CTX mechanism.

- The approach of detecting a Route Disruption DDoS attack involves sending a 802.11 NULL frame on a route from which an RERR frame has been received. Lazy action on receiving RERR frames might help avoid the overhead of establishing a route again using RREQ/RREP on an already established legitimate route.

- All single hop wireless links between the Mesh Portal Point and its peers need to be secure encrypted channels.

- All devices in the mesh should be able to decipher that a frame transmitted by a Mesh Portal Point is legitimate using cryptographic techniques.

Chapter 7

# Conclusion

The next generation of wireless networks may include city-wide multi-hop IEEE 802.11s WMN's. The advantages of WMN's include ease of deployment in hostile terrains, last-mile connectivity through multiple hops, self-healing robust routing algorithms and effective power-save mechanisms. There are several new challenges in terms of security, route management, Quality of Service (QoS) and interoperability with other protocols. The focus in this thesis was on security and specifically robustness towards DDoS attacks in 802.11s WMN's. These attacks have been seen to result in complete unavailability of the wireless network along with depletion of battery power which is an important resource in mobile environments. These attacks are particularly devastating in 802.11s WMN's because a single packet could have traversed multiple hops and consumed significant amount of network and power resources before it gets dropped due to a DDoS attack.

The attacks considered the following:

- Sybil Attacks.
- Masquerading Attacks.

- Routing Attacks.

The simulation experiments were carried out using the GTNetS simulation environment. The experimental results show that malicious attackers can perform DDoS attacks resulting in either complete unavailability or significant degradation of overall performance of the mesh. The solutions are shown to have minimal overhead and are effective in successfully countering the DDoS attacks. A uniform framework for providing an HMAC with critical frames as identified in this thesis. The experiments also highlight the importance of securing the Mesh Portal Points. The Mesh Portal Points can be deemed as the back bone of the WMN.

However there are various remaining research problems. Retroactive security amendments to the original protocol results in vulnerable implementations. The specification designed by 802.11w [29], or the Protected Management Frames Group needs to be incorporated in all 802.11 solutions to ensure adequate security against DDoS attacks. It is also difficult to design a complete security architecture for 802.11 WMN's without loopholes due to the presence of a large number of different types of devices. Attacks can be carried out in each layer of the network protocol stack. The focus in this thesis has been on DDoS attacks in the MAC layer only. A centralized authentication server for a WMN is not feasible as it will severely affect the scalability of WMN's. Though 802.11s has been designed to be scalable for a city-wide multi-hop WMN there is significant additional research to feasibly secure a large WMN.

Appendix A

# GTNetS MAC Layer Modifications

The following sections contain the modifications made to the 802.11 MAC Layer in GTNetS simulator. Only the important modifications have been included. The details of the features added have been described in Chapter 5. Approximately 2200 lines of source code have been added to the original GTNetS framework.

## A.1 Patch for 802.11 MAC Header File

```
diff -crB ../orig/GTNetS-Oct-10-08/SRC/l2proto802.11.h
          ../GTNetS-Oct-10-08/SRC/l2proto802.11.h
*** ../orig/GTNetS-Oct-10-08/SRC/l2proto802.11.h
          2008-10-03 18:00:00.000000000 -0400
--- ../GTNetS-Oct-10-08/SRC/l2proto802.11.h
          2009-01-26 23:16:56.000000000 -0500
***************
*** 160,165 ****
--- 160,172 ----

+ /* Crypto Includes */
+ #include <gcrypt.h>
+ #include <termios.h>
+ #include <unistd.h>
```

```
+ #include <stdlib.h>
+ #include <sys/stat.h>
+
--- 174,190 ----
  #define MAC_Type_Data           0x02
  #define MAC_Type_Reserved       0x03

+ #define MAC_Subtype_PS_POLL     0x0A
+ #define MAC_Subtype_NULL_DATA   0x04
+ #define MAX_STATIONS            16
  using namespace std;

+ class Queue;

***************
*** 232,237 ****
--- 242,258 ----

+ class Info_element {
+  public:
+       u_char el_id;
+       u_char len;
+ };
+
+ class chall_text_element : public Info_element {
+  public:
+       char text[256];
+ };
+
  class SeqCacheEntry {
***************
*** 242,247 ****
--- 263,284 ----
    Word_t  sn;  // sequence number
  };

+ struct sta_info
+ {
+ public:
+   MACAddr stamac;   // Station MAC address
+   Word_t  aid;
+   bool    authenticated;
```

```
+   bool    associated;
+   bool    authenticationPending;
+   bool    deauthPending;
+   bool    resetDeauthPending;
+   bool    powerSave;
+   bool    setMoreDataBit;
+   bool    sentNullPS_POLL;
+   Queue   *powerSaveQueue;
+ };
+

***************
*** 352,357 ****
--- 389,405 ----
+   bool SendAuthReq(Time_t time, MACAddr dst,
+                    Word_t algo,Word_t trans);
+   bool SendAuthResp(Time_t time, MACAddr dst,Word_t result);
    bool SendAssocReq(Time_t time, MACAddr dst);

***************
*** 370,381 ****
--- 418,447 ----
+   bool SendDisassocFrame(MACAddr addr_, MACAddr srcMAC,
+                          IPAddr_t srcIP,Time_t time);
    bool SendDisassocFrame(MACAddr addr_, Time_t time);
+   bool SendDeauthFrame(Time_t time);
+   bool Send_PS_POLL(MACAddr , MACAddr , Word_t dur,
+                     Time_t time,bool attack);
+
+   bool SendDeauthFrame(MACAddr addr_, MACAddr srcMAC,
+                        IPAddr_t srcIP,Time_t time);
+   void UpdateStaInfo(MACAddr addr_,bool authenticated,
+                      bool associated,bool remove,
+                      bool deauthPending,bool resetDeauthPending,
+                      bool powerSave,bool sentNullPS_POLL);
+   Word_t CheckStaInfoList(MACAddr addr,
+                      bool deauthPending,bool sentNullPS_POLL);
+
    bool SendReassocReq(Time_t time, MACAddr dst);
***************
*** 488,494 ****
+   struct sta_info stainfo[MAX_STATIONS];
```

```
+   chall_text_element   g_cte;
+   gcry_cipher_hd_t     cipherHandle;
+   Word_t          nonce;
+   char            *wepDefaultKey;
+   char            *wepIV;
+   Word_t          aid;
+   bool            stationSentPS_POLL;
    Word_t          cw;
***************
*** 522,527 ****
--- 595,602 ----
    LastHeardMap_t lastH;

    /* DS for BSS mode */
+   bool            authenticated;
+   bool            authenticationPending;

***************
--- 611,617 ----
+   MACAddr         MACAP;
    static Word_t   defaultSeqCacheSize;

    /* RTS - CTS */
***************
*** 564,569 ****
--- 639,645 ----
  public:
+   Count_t  freeIndex;
***************
*** 572,577 ****
--- 648,656 ----
    static Count_t  dataRxCount;
+   static Count_t  dataRxCount3;
+   static Count_t  dataRxCount4;
+   static Count_t  dataRxCount2;
    static Count_t  ackRxCount;
    static Count_t  rtsDelCount;
***************
*** 581,595 ****
--- 660,679 ----
+   static Count_t  authReqCount;
+   static Count_t  authRepCount;
```

```
+   static Count_t  deauthCount;
+   static Count_t  PS_POLL_Count;
  };

+
  struct Frame_control
  {
  public:
***************
--- 710,718 ----
   BEACON,
   DISASSOC,
   AUTH,
+  AUTHRES,
+  DEAUTH,
+  PS_POLL
  } Frametype_t;

***************
*** 666,671 ****
--- 752,793 ----

+ class PS_POLL_frame : public L2Header802_11
+ {
+ public:
+   PS_POLL_frame (): L2Header802_11()
+   { type = PS_POLL; L2Proto802_11::PS_POLL_Count++;}
+   PS_POLL_frame (Word_t aid, MACAddr rcv, MACAddr xmt,
+                  Long_t fcs_);
+   // Copy constructor
+   PS_POLL_frame(const PS_POLL_frame&);
+
+   bool PS_POLL_attack;
+   Size_t  Size() const;
+   PDU*    Copy() const
+   { return new PS_POLL_frame(*this);}
+   void Trace(Tfstream&, Bitmap_t, Packet*, const char*);
+ };
+

***************
*** 783,789 ****
```

```
+   DATA_frame(Word_t dur, MACAddr rcv, MACAddr xmt,
+              Word_t sqn, Long_t fcs_);
+   DATA_frame(Word_t dur, MACAddr rcv, MACAddr xmt,
+              Word_t sqn, Long_t fcs_,
+              bool Null_Frame,bool SentNullPS_POLL);
***************
*** 819,824 ****
--- 941,947 ----
+   bool     PS_POLL_Sent;
  };

***************
*** 832,842 ****
    Information elements
   */

***************
*** 874,880 ****
  u_char dtim_count;
  u_char dtim_period;
  u_char bitmap_control;
+ u_char pv_bitmap[256];
  };

  class ibss_paramset_element : public Info_element {

***************
*** 1071,1085 ****
--- 1245,1308 ----
  };

+ #define OPEN_SYSTEM_AUTH 01
+ #define SHARED_KEY_AUTH  02
+
  class Auth_frame: public BSSBaseFrame {
   public:
    Auth_frame() : BSSBaseFrame() {
      type = AUTH;
    }
+   Auth_frame(MACAddr src, IPAddr_t srcIP,
+              MACAddr dst,Word_t algo_num,
+              Word_t algo_trans_num)
```

```
+              : BSSBaseFrame(src, dst) {
+     type = AUTH;
+     fc.fc_more_frag  = 0;
+     fc.fc_retry      = 0;
+     fc.fc_pwr_mgt    = 0;
+     fc.fc_more_data  = 0;
+     fc.fc_wep        = 0;
+     fc.fc_order      = 0;
+     sip = srcIP;
+     L2Proto802_11::authReqCount++;
+     algo_nr = algo_num;
+     algo_trans_nr = algo_trans_num;
+   }
+
+
    Size_t  Size() const;
    PDU*    Copy() const { return new Auth_frame(*this); }
    void    Trace(Tfstream&, Bitmap_t,
    Packet* = nil, const char* = nil);
   public:
+       IPAddr_t sip;
+ Word_t algo_nr;
+ Word_t algo_trans_nr;
+ Word_t status;
+ chall_text_element cte;
+ };
+
+
+ class Authresp_frame: public BSSBaseFrame {
+  public:
+   Authresp_frame() : BSSBaseFrame() { type = AUTHRES;}
+   Authresp_frame(MACAddr src, MACAddr dst,Word_t result)
+       : BSSBaseFrame(src, dst) {
+     type = AUTHRES;
+     fc.fc_more_frag  = 0;
+     fc.fc_retry      = 0;
+     fc.fc_pwr_mgt    = 0;
+     fc.fc_more_data  = 0;
+     fc.fc_wep        = 0;
+     fc.fc_order      = 0;
+     L2Proto802_11::authRepCount++;
+     status = result;
```

```
+   }
+
+   Size_t  Size() const;
+   PDU*    Copy() const
+   { return new Authresp_frame(*this); }
+   void    Trace(Tfstream&, Bitmap_t,
+           Packet* = nil, const char* = nil);
+  public:
  Word_t status;
***************
--- 1313,1340 ----
   public:
      Deauth_frame() : BSSBaseFrame () {
      type = DEAUTH;
+     reason_code = 0;
+     L2Proto802_11::deauthCount++;
+   }
+   Deauth_frame(MACAddr src, IPAddr_t srcIP, MACAddr dst)
+       : BSSBaseFrame(src, dst) {
+     type = DEAUTH;
+     fc.fc_more_frag  = 0;
+     fc.fc_retry      = 0;
+     fc.fc_pwr_mgt    = 0;
+     fc.fc_more_data  = 0;
+     fc.fc_wep        = 0;
+     fc.fc_order      = 0;
+     sip = srcIP;
+     reason_code=0;
+     L2Proto802_11::disassocCount++;
    }
    Size_t  Size() const;
```

## A.2 Patch for 802.11 MAC C++ File

```
diff -crB ../orig/GTNetS-Oct-10-08/SRC/l2proto802.11.cc
          ../GTNetS-Oct-10-08/SRC/l2proto802.11.cc
***       ../orig/GTNetS-Oct-10-08/SRC/l2proto802.11.cc
          2008-10-10 16:24:22.000000000 -0400
---       ../GTNetS-Oct-10-08/SRC/l2proto802.11.cc
          2009-02-01 01:22:14.000000000 -0500
***************
```

```
*** 162,173 ****
+ //#define AUTH_TYPE OPEN_SYSTEM_AUTH
+ #define ADHOC_TESTING 1
+ #define BLACK_HOLE    0
+ #define GREY_HOLE     1
+
+ #define AUTH_TYPE SHARED_KEY_AUTH
+ #define DEAUTH_SOLN 1
+ #define POWER_SOLN  0
+ chall_text_element g_cte;
+

***************
*** 220,232 ****
--- 229,249 ----
+ Count_t L2Proto802_11::authReqCount = 0;
+ Count_t L2Proto802_11::authRepCount = 0;
+ Count_t L2Proto802_11::deauthCount = 0;
+ Count_t L2Proto802_11::PS_POLL_Count = 0;
+ Count_t L2Proto802_11::dataRxCount = 0;
+ Count_t L2Proto802_11::dataRxCount3 = 0;
+ Count_t L2Proto802_11::dataRxCount4 = 0;
+ Count_t L2Proto802_11::dataRxCount2 = 0;

***************
*** 264,269 ****
--- 281,288 ----

+   authenticated = false;
+   authenticationPending = false;

***************
*** 275,280 ****
--- 294,300 ----

+   stationSentPS_POLL = false;
***************
*** 287,292 ****
--- 307,318 ----

+   if(cipherHandle)
+   {gcry_cipher_close(cipherHandle);}
```

```
+   free(wepDefaultKey);
+   free(wepIV);

--- 360,403 ----
+ case InterfaceWireless::BSS_POWERSAVE:
        case InterfaceWireless::BSS:
+       drop = !associated;
+            break;
          case InterfaceWireless::HOSTAP:
+         {
+           Word_t aid = CheckStaInfoList(df->ra,
+                                         false,false);
+           if(!aid || btv.empty())
+           {
+               drop = true; break;
+           }
+           //Take some queueing action here for power save
+           if(stainfo[aid].powerSaveQueue)
+           {
+               if(!df->PS_POLL_Sent)
+               {
+                   stainfo[aid].powerSaveQueue->Enque(p);
+                   pendingPkt = iface->GetQueue()->Deque();
+                   return;
+               }
+               else
+               {
+                   if(stainfo[aid].powerSaveQueue->LengthPkts()
+                                                          > 0)
+                   {
+                       df->fc.fc_more_data=1;
+                       stainfo[aid].setMoreDataBit = true;
+                   }
+                   else
+                   {
+                       stainfo[aid].setMoreDataBit = false;
+                       stainfo[aid].sentNullPS_POLL = false;
+                   }
+               }
+            }
+         }
```

```
***************
+   // Update Power Save Status for stations at Access Point.
+   if ( ((InterfaceWireless*)iface)->GetOpMode()
+                               ==  InterfaceWireless::HOSTAP)
+   {
+       if(l2pdu->fc.fc_pwr_mgt == 1)
+       {
+           UpdateStaInfo(l2pdu->ta,false,false,false,
+                         false,false,true,false);
+       }
+   }
+
***************
*** 585,591 ****
--- 647,705 ----
+               if ( ((InterfaceWireless*)iface)->GetOpMode()
+                     ==  InterfaceWireless::HOSTAP)
+               {
+                   if(!CheckStaInfoList(dataFrame->addr2,
+                                                false,false))
+                   {
+                       delete p;
+                       Stats::pktsDropped++; break;
+                   }
+               }
+               if ( ((InterfaceWireless*)iface)->GetOpMode()
+                          == InterfaceWireless::BSS_POWERSAVE)
+               {
+                   if(!stationSentPS_POLL)
+                   {
+                       delete p; break;
+                   }
+                   else
+                   {
+                       stationSentPS_POLL = false;
+                   }
+               }
+ #if ADHOC_TESTING
+ #if BLACK_HOLE
+ if(dataFrame->ra.macAddr == 3 && Simulator::Now() > 2)
+ {
+     cout << "DROPPING - Maliciously\n";
```

```
+     delete p;
+     Stats::pktsDropped++; // Count dropped packets
+     break;
+ }
+ #endif
+ #if GREY_HOLE
+ if(dataFrame->ra.macAddr == 6&&
+   (Simulator::Now() - int(Simulator::Now()))  >= 0.50
+    && Simulator::Now() > 2)
+ {
+     cout << "DROPPING - Maliciously\n";
+     delete p;
+     Stats::pktsDropped++; // Count dropped packets
+     break;
+ }
+ #endif
+ #endif
***************
*** 605,610 ****
--- 719,744 ----
  SendACK(dataFrame->addr2); // src-dest pair
+ dataRxCount++;
+ if(dataFrame->fc.fc_more_data == 1)
+ {
+     Send_PS_POLL(iface->GetMACAddr(),dataFrame->ta,
+                  aid,sifs,false);
+ }

***************
*** 617,623 ****
+ if(dataFrame->fc.fc_subtype
+             != MAC_Subtype_NULL_DATA)
+ {
+     iface->HandleLLCSNAP(p, fbrx);
+ }
***************
***************
*** 660,665 ****
--- 802,819 ----

+
+ #if DEAUTH_SOLN
```

```
+             if ( ((InterfaceWireless*)iface)->GetOpMode()
+                             ==  InterfaceWireless::HOSTAP)
+             {
+                 DATA_frame* df =
+                 (DATA_frame*)pendingPkt->PeekPDU();
+                 if(CheckStaInfoList(df->ra,true,false))
+                 {
+                     UpdateStaInfo(df->ra,false,false,false,
+                       false,true,false,false);
+                 }
+             }
+ #endif
***************
+                   if (beaconRxCount % 5 == 0)
+                   {
+                        if((beaconFrame->tim.pv_bitmap[aid/8]> 0)
+                          &&((beaconFrame->tim.pv_bitmap[aid/8]
+                          & (1<<(aid%8)))>0))
+                        {
+                            Send_PS_POLL(iface->GetMACAddr(),
+                                 beaconFrame->ta,aid,sifs,false);
+                        }
+                   }
+                   if(authenticated)
+                   {
+                        SendAssocReq(sifs, beaconFrame->ta);
+                   }
+                   else
+                   {
+                       if(!authenticationPending)
+                       {
+                         SendAuthReq(sifs, beaconFrame->ta,
+                                     AUTH_TYPE,1);
+                       }
                   }

+     case AUTH:
+       {
+           Auth_frame* asreqFrame = (Auth_frame*) l2pdu;
+           if (asreqFrame->ra == iface->GetMACAddr())
+           { // Got association request addressed to me
+             if ( ((InterfaceWireless*)iface)->GetOpMode()
```

```
+                   == InterfaceWireless::HOSTAP)
+                {
+                    if(asreqFrame->algo_nr == OPEN_SYSTEM_AUTH
+                    && asreqFrame->algo_trans_nr == 1)
+                    {
+                       SendAuthResp(sifs, asreqFrame->ta,0);
+                                   MACAP = asreqFrame->ra;
+                       UpdateStaInfo(asreqFrame->ta,true,
+                       false,false,false,false,false,false);
+                    }
+                    if(asreqFrame->algo_nr == SHARED_KEY_AUTH &&
+                        asreqFrame->algo_trans_nr == 1)
+                    {
+                        /* Build Challenge Text Here */
+                        Time_t now = Simulator::Now();
+                        memset(g_cte.text,0x0,256);
+                        strncpy((char*)(g_cte.text),
+                        (const char*)(&asreqFrame->ta),
+                        sizeof(MACAddr));
+                        strncpy((char*)(g_cte.text +
+                        sizeof(MACAddr)),
+                        (const char*)(&now),sizeof(now));
+                        strncpy((char*)(g_cte.text +
+                              sizeof(MACAddr)+ sizeof(now)),
+                        (const char*)(&nonce),sizeof(nonce));
+                        MACAP = asreqFrame->ra;
+                        SendAuthReq(sifs, asreqFrame->ta,
+                        SHARED_KEY_AUTH,2);
+                    }
+
+                    if(asreqFrame->algo_nr == SHARED_KEY_AUTH &&
+                                asreqFrame->algo_trans_nr == 3)
+                    {
+                        int result = 0;
+                        /* Set the IV to be used for encryption*/
+                        if(gcry_cipher_setiv(cipherHandle,
+                                           wepIV,16) != 0)
+                        {
+                            printf("setting the IV failed\n");
+                            assert(0);
+                        }
+
```

```
+                       if(gcry_cipher_decrypt(cipherHandle,
+                               (char*)asreqFrame->cte.text,
+                       {
+                           printf("Decryption failed\n");
+                           assert(0);
+                       }
+
+                       if((MACAddr)asreqFrame->cte.text[0] !=
+                                   asreqFrame->ta)
+                       {
+                           if(Simulator::Now() -
+                           (*(Time_t*)(asreqFrame->cte.text+4))
+                           > 0.002)
+                           {
+                               result = 1;
+                           }
+                       }
+
+                       SendAuthResp(sifs, asreqFrame->ta,result);
+
+                       if(result == 1)
+                       {
+                           assert(0);
+                       }
+                       else
+                       {
+                          UpdateStaInfo(asreqFrame->ta,true,
+                            false,false,false,false,false,false);
+                       }
+                   }
+                   WirelessLink* wlink =
+                               (WirelessLink*)iface->GetLink();
+                   Interface* tif =
+                                wlink->GetIfByMac(asreqFrame->ta);
+                   if(!tif)
+                   {
+                       //assert(0);
+                   }
+                 }
+               if ( ((InterfaceWireless*)iface)->GetOpMode()
+                    == InterfaceWireless::BSS ||
+                  ((InterfaceWireless*)iface)->GetOpMode()
```

```
+                  == InterfaceWireless::BSS_POWERSAVE)
+           {
+               if(asreqFrame->algo_nr == SHARED_KEY_AUTH
+                   && asreqFrame->algo_trans_nr == 2
+                   && authenticationPending)
+               {
+                   /* Set the IV to be used for encryption */
+                   if(gcry_cipher_setiv(cipherHandle,
+                                        wepIV,16) != 0)
+                   {
+                       printf("setting the IV failed\n");
+                       assert(0);
+                   }
+
+                   if(gcry_cipher_encrypt(cipherHandle,
+                      &asreqFrame->cte.text,256,NULL,0) != 0)
+                  {
+                      printf("Encryption failed\n");
+                      assert(0);
+                  }
+                  memcpy(g_cte.text,asreqFrame->cte.text,256);
+                  SendAuthReq(sifs,asreqFrame->ta,
+                              SHARED_KEY_AUTH,3);
+               }
+           }
+         }
+        delete p;
+        break;
+      }
+    case AUTHRES:
+    {
+        Authresp_frame* asrespFrame = (Authresp_frame*) l2pdu;
+        if (asrespFrame->ra == iface->GetMACAddr())
+          { // Addressed to me
+            if ( ((InterfaceWireless*)iface)->GetOpMode()
+                 == InterfaceWireless::BSS ||
+               ((InterfaceWireless*)iface)->GetOpMode()==
+                InterfaceWireless::BSS_POWERSAVE)
+              {
+                  if(asrespFrame->status == 0)
+                  {
+                     authenticated = true;
```

```
+                   }
+                   else
+                   {
+                      authenticated = false;
+                   }
+                   authenticationPending = false;
+                   WirelessLink* wlink =
+                    (WirelessLink*)iface->GetLink();
+                   Interface* tif =
+                     wlink->GetIfByMac(asrespFrame->ta);
+                   if(!tif)
+                   {
+                       //assert(0)
+                   }
+               }
+           }
+           delete p;
+           break;
+       }
+

+ #if DEAUTH_SOLN
+    if(CheckStaInfoList(dframe->ta,false,false))
+    {
+        Word_t dur = usec(sifs +
+                txtime(ETHER_ACK_LEN, basicRate) +
+                DSSS_MaxPropagationDelay);
+        DATA_frame* pdu = new DATA_frame(dur,
+        dframe->ta,dframe->ra, ++seqNo, 0,true,false);
+        UpdateStaInfo(dframe->ta,false,false,
+                false,true,false,false,false);
+        Packet* p = new Packet();
+        p->PushPDU(pdu);
+        if(Busy())
+        {
+            Queue *q = iface->GetQueue();
+            q->Enque(p);
+        }
+        else
+        {
+            DataRequest(p);
+        }
```

```
+        break;
+    }
+ #else
+    RemoveFromBTV(dframe->sip);
+    UpdateStaInfo(dframe->ta,false,false,true,
+              false,false,false,false);
+ #endif
+
+ delete p;
+ break;
+ }
+     case DEAUTH:
+     {
+         Deauth_frame* dframe = (Deauth_frame*) l2pdu;
+         if (dframe->ra == iface->GetMACAddr())
+           { // Got deauth frame addressed to me
+             if ( ((InterfaceWireless*)iface)->GetOpMode()
+                  == InterfaceWireless::HOSTAP) {
+ #if DEAUTH_SOLN
+             if(CheckStaInfoList(dframe->ta,false,false))
+             {
+                 Word_t dur = usec(sifs +
+                         txtime(ETHER_ACK_LEN, basicRate)
+                         + DSSS_MaxPropagationDelay);
+                 DATA_frame* pdu = new DATA_frame(dur,dframe->ta,
+                      dframe->ra, ++seqNo, 0,true,false);
+                 UpdateStaInfo(dframe->ta,false,false,
+                      false,true,false,false,false);
+                 Packet* p = new Packet();
+                 p->PushPDU(pdu);
+                 if(Busy())
+                 {
+                     Queue *q = iface->GetQueue();
+                     q->Enque(p);
+                 }
+                 else
+                 {
+                     DataRequest(p);
+                 }
+                 break;
+             }
+ #else
```

```
+             RemoveFromBTV(dframe->sip);
+             UpdateStaInfo(dframe->ta,false,false,
+                    true,false,false,false,false);
+ #endif

***************
*** 902,911 ****
--- 1306,1394 ----
          delete p;
          break;
        }
+     case PS_POLL:
+       {
+           Packet *temp          = NULL;
+           DATA_frame *tempData  = NULL;
+           PS_POLL_frame* dframe = (PS_POLL_frame*) l2pdu;
+           Word_t staid = dframe->duration;
+           if (dframe->ra == iface->GetMACAddr())
+           {
+               if ( ((InterfaceWireless*)iface)->GetOpMode()
+                               ==  InterfaceWireless::HOSTAP)
+               {
+ #if POWER_SOLN
+                   if(stainfo[staid].powerSaveQueue &&
+                   stainfo[staid].powerSaveQueue->LengthPkts()> 0
+                   && !stainfo[staid].setMoreDataBit &&
+                   !stainfo[staid].sentNullPS_POLL)
+                   {
+                       Word_t dur = usec(sifs +
+                            txtime(ETHER_ACK_LEN, basicRate)
+                            + DSSS_MaxPropagationDelay);
+                       DATA_frame* pdu = new DATA_frame(dur,
+                             dframe->ta,dframe->ra, ++seqNo, 0,
+                             true,true);
+                       UpdateStaInfo(dframe->ta,false,
+                            false,false,false,false,false,true);
+                       stainfo[staid].setMoreDataBit = true;
+                       pdu->PS_POLL_Sent = true;
+                       Packet* p = new Packet();
+                       p->PushPDU(pdu);
+                       if(Busy())
+                       {
```

```
+                           Queue *q = iface->GetQueue();
+                           q->Enque(p);
+                       }
+                       else
+                       {
+                           DataRequest(p);
+                       }
+                       break;
+                   }
+ #endif
+   if(stainfo[staid].powerSaveQueue &&
+     stainfo[staid].powerSaveQueue->LengthPkts() > 0)
+   {
+       temp =
+          stainfo[staid].powerSaveQueue->Deque();
+   }
+   else
+   {
+       break;
+       assert(0);
+   }
+
+   if(temp)
+   {
+       tempData = (DATA_frame*)temp->PeekPDU();
+   }
+   else
+   {
+      assert(0);
+   }
+
+   if(tempData)
+   {
+      tempData->PS_POLL_Sent = true;
+   }
+   else
+   {
+      assert(0);
+   }
+
+   if(Busy())
+   {
```

```
+       Queue *q = iface->GetQueue();
+       q->Enque(temp);
+   }
+   else
+   {
+       DataRequest(temp);
+   }
+ }
+ }
+ delete p;
+ break;
+ }

***************
*** 940,945 ****
--- 1423,1444 ----
  }

  // Sending the packets
+ bool L2Proto802_11::Send_PS_POLL(MACAddr ta, MACAddr ra,
+      Word_t aid,Time_t time,bool attack)
+ {
+     PS_POLL_frame* rf = new PS_POLL_frame(aid, ra, ta, 0);
+     rf->PS_POLL_attack = attack;
+     Packet* p = new Packet();
+     p->PushPDU(rf);
+     ScheduleSendAfterIFS(p, time);
+     return true;
+ }

***************
*** 1451,1458 ****
--- 2092,2146 ----
+ case PS_POLL:
+ {
+     if (MediumBusy())
+     {
+         if((navTimer - Simulator::Now()) > 1e-4)
+             ScheduleSendAfterIFS(mev->p,
+            (navTimer - Simulator::Now())+0.000005);
+         else
+             ScheduleSendAfterIFS(mev->p, 0.0000075);
```

```
+     }
+     else
+     {
+         PS_POLL_frame* pspoll = (PS_POLL_frame*)l2pdu;
+                 // Trace this send
+         iface->GetNode()->TracePDU(this,
+                     l2pdu, mev->p, "-");
+         successfulTx =
+         iface->GetLink()->Transmit(mev->p,
+                iface, iface->GetNode(),
+                txtime(mev->p->Size(), basicRate));
+         if(!successfulTx)
+         {
+             ScheduleSendAfterIFS(mev->p, 0.0000075);
+         }
+         else
+         {
+             if(!pspoll->PS_POLL_attack)
+             {
+                 stationSentPS_POLL = true;
+             }
+             else
+             {
+                 //cout<<"INJECTING PS POLL ATTACK\n";
+             }
+             delete mev->p; /* SDR */
+         }
+     }
+     break;
+ }
+
***************
--- 2267,2304 ----
      case MACTimerEvent::ACK_TIMEOUT:
      {
          ackTOCount++;
          DATA_frame* df = (DATA_frame*)pendingPkt->PeekPDU();
+ #if DEAUTH_SOLN
+         if ( ((InterfaceWireless*)iface)->GetOpMode()
+              ==  InterfaceWireless::HOSTAP)
+         {
+             if(CheckStaInfoList(df->ra,true,false))
```

```
+             {
+                 WirelessLink* wlink =
+                    (WirelessLink*)iface->GetLink();
+                 Interface* tif =
+                    wlink->GetIfByMac(df->ra);
+                 RemoveFromBTV(tif->GetNode()->GetIPAddr());
+                 UpdateStaInfo(df->ra,false,false,
+                     true,false,false,false,false);
+                 break;
+             }
+ #if POWER_SOLN
+         if(CheckStaInfoList(df->ra,false,true))
+         {
+             mac_state = IDLE;
+             backingOff = false;
+             backoffTimerState = NONE;
+             CancelTimer(backoffTimeout);
+             DropPacket("ACK-TO");
+             stainfo[CheckStaInfoList(df->ra,
+                     false,true)].setMoreDataBit  = false;
+             stainfo[CheckStaInfoList(df->ra,
+                     false,true)].sentNullPS_POLL = false;
+             break;
+         }
+ #endif
+ #endif
+   int i;
+   aid = 0;
+   if(gcry_cipher_open(&cipherHandle,
+      GCRY_CIPHER_AES128,GCRY_CIPHER_MODE_CBC,0) != 0)
+   {
+       printf("Opening Handle for AES failed\n");
+       assert(0);
+   }
+
+   if((wepDefaultKey = (char*)malloc(sizeof(char)*16)) == NULL)
+   {
+       printf("Malloc failed\n");
+       assert(0);
+   }
+
+   if((wepIV = (char*)malloc(sizeof(char)*16)) == NULL)
```

```
+   {
+       printf("Malloc failed\n");
+       assert(0);
+   }
+
+   for(i=0;i<16;i++)
+   {
+       wepDefaultKey[i]=i;
+       wepIV[i] = i+20;
+   }
+
+   if(gcry_cipher_setkey(cipherHandle,wepDefaultKey,16) !=0)
+   {
+       printf("setting the cipher key failed\n");
+       assert(0);
+   }
+
+
    if (ifw->GetOpMode() == InterfaceWireless::HOSTAP)
+   {
+       freeIndex=1;
+       nonce = 16;
+       for(i=0;i<MAX_STATIONS;i++)
+       {
+           stainfo[i].stamac                = 0;
+           stainfo[i].aid                   = 0;
+           stainfo[i].authenticated         = 0;
+           stainfo[i].authenticationPending = 0;
+           stainfo[i].associated            = 0;
+           stainfo[i].deauthPending         = 0;
+           stainfo[i].resetDeauthPending    = 0;
+           stainfo[i].powerSave             = 0;
+           stainfo[i].powerSaveQueue        = NULL;
+           stainfo[i].setMoreDataBit        = NULL;
+           stainfo[i].sentNullPS_POLL       = NULL;
+       }
        ScheduleBeaconTimer(time);
+   }
  }

***************
*** 2041,2046 ****
```

```
--- 2814,2826 ----
  {
    Assocreq_frame* AReqf = new Assocreq_frame(
        iface->GetMACAddr(), iface->GetIPAddr(), da);
+
+   InterfaceWireless* ifw = (InterfaceWireless*)iface;
+   if (ifw->GetOpMode() == InterfaceWireless::BSS_POWERSAVE)
+   {
+       AReqf->fc.fc_pwr_mgt    = 1;
+   }
    Packet* p = new Packet();
***************
*** 2051,2061 ****
    return true;
  }

+ bool L2Proto802_11::SendAssocResp(Time_t time, MACAddr da)
  {

+   Assocresp_frame* ARespf = new Assocresp_frame(
+                     iface->GetMACAddr(),da);
    Packet* p = new Packet();
    p->PushPDU(ARespf);

--- 2831,2885 ----
    return true;
  }

+ bool L2Proto802_11::SendAuthReq(Time_t time, MACAddr da,
+                                 Word_t algo,Word_t trans)
+ {
+   int k=0;
+   Auth_frame* AReqf = new Auth_frame(
+       iface->GetMACAddr(), iface->GetIPAddr(), da,algo,trans);
+   if(algo == SHARED_KEY_AUTH)
+   {
+       if(trans == 2)
+       {
+           for(k=0;k<256;k++)
+           AReqf->cte.text[k] = g_cte.text[k];
+       }
+       if(trans == 3)
```

```
+       {
+           for(k=0;k<256;k++)
+           AReqf->cte.text[k] = g_cte.text[k];
+       }
+   }
+   Packet* p = new Packet();
+   p->PushPDU(AReqf);
+   ScheduleSendAfterIFS(p, time);
+   authenticationPending = true;
+   return true;
+ }
+
+ bool L2Proto802_11::SendAuthResp(Time_t time, MACAddr da,
+                                  Word_t result)
  {

+   Authresp_frame* ARespf =
+            new Authresp_frame(iface->GetMACAddr(),
+                               da,result);
+   Packet* p = new Packet();
+   p->PushPDU(ARespf);
+   ScheduleSendAfterIFS(p, time);
+   return true;
+ }
+
+ bool L2Proto802_11::SendAssocResp(Time_t time, MACAddr da)
+ {
+   Assocresp_frame* ARespf =
+           new Assocresp_frame(iface->GetMACAddr(),da);
+   if(CheckStaInfoList(da,false,false))
+   {
+       ARespf->aid = CheckStaInfoList(da,false,false);
+   }

***************
*** 2089,2100 ****
--- 2913,2968 ----
      ScheduleSendAfterIFS(p, time);
      associated = false;
+     authenticated = false;
      return true;
    }
```

```
    else
      return false;
  }

+ bool L2Proto802_11::SendDisassocFrame(
+      MACAddr addr_, MACAddr srcMAC,
+      IPAddr_t srcIP,Time_t time)
+ {
+     Disassoc_frame* Disassf = new Disassoc_frame(
+                                srcMAC, srcIP,addr_);
+     Packet* p = new Packet();
+     p->PushPDU(Disassf);
+     ScheduleSendAfterIFS(p, time);
+     associated = false;
+     return true;
+ }
+
+ bool L2Proto802_11::SendDeauthFrame(MACAddr addr_,
+      MACAddr srcMAC,IPAddr_t srcIP,Time_t time)
+ {
+     Deauth_frame* Disassf = new Deauth_frame(
+                            srcMAC,srcIP,addr_);
+     Packet* p = new Packet();
+     p->PushPDU(Disassf);
+     ScheduleSendAfterIFS(p, time);
+     associated = false;
+     authenticated = false;
+     return true;
+ }
+
+ bool L2Proto802_11::SendDeauthFrame(Time_t time)
+ {
+     Deauth_frame* Disassf = new Deauth_frame(
+         iface->GetMACAddr(),iface->GetIPAddr(),
+         assocBssid);
+     Packet* p = new Packet();
+     p->PushPDU(Disassf);
+     ScheduleSendAfterIFS(p, time);
+     associated = false;
+     authenticated = false;
+     return true;
+ }
```

```
  bool L2Proto802_11::SendReassocReq(Time_t time, MACAddr da)
  {
***************
*** 2156,2162 ****
--- 3024,3048 ----

    bf->tim.el_id = 5;
    bf->tim.len = 0;
+
+   int i;
+   for(i = 0;i< 252; i++)
+   {
+       bf->tim.pv_bitmap[i] = 0;
+   }

+   for(i = 0;i < MAX_STATIONS; i++)
+   {
+       if(stainfo[i].powerSaveQueue)
+       {
+           if(stainfo[i].powerSaveQueue->LengthPkts() > 0)
+           {
+               int dummy = i/8;
+               bf->tim.pv_bitmap[dummy] |= (1<<(i%8));
+           }
+       }
+
+   }
    Packet* p = new Packet();
***************
*** 2211,2216 ****
--- 3098,3224 ----
  }
+ Word_t L2Proto802_11::CheckStaInfoList(MACAddr addr,
+             bool deauthPending,bool sentNullPS_POLL)
+ {
+     int find=0;
+     for(find=0;find<MAX_STATIONS;find++)
+     {
+         if(deauthPending)
+         {
+             if(stainfo[find].stamac == addr &&
```

```
+                stainfo[find].deauthPending)
+                 return find;
+             else
+                 continue;
+         }
+         if(sentNullPS_POLL)
+         {
+             if(stainfo[find].stamac == addr &&
+                 stainfo[find].sentNullPS_POLL)
+                 return find;
+             else
+                 continue;
+         }
+         if(stainfo[find].stamac == addr)
+         {
+             return find;
+         }
+     }
+     return 0;
+ }
+
+ void L2Proto802_11::UpdateStaInfo(MACAddr addr,
+    bool authenticated,
+    bool associated,
+    bool remove,
+    bool deauthPending,
+    bool resetDeauthPending,
+    bool powerSave,
+    bool sentNullPS_POLL)
+ {
+     int i=0,find=0,count=0;
+     bool found;
+     if(stainfo[freeIndex].stamac != 0)
+     {
+         goto printret;
+     }
+     for(find=0;find<MAX_STATIONS;find++)
+     {
+         if(stainfo[find].stamac == addr)
+             {found = true;break;}
+     }
+
```

```
+    if(remove && found)
+    {
+        stainfo[find].stamac = 0;
+        stainfo[find].authenticated = false;
+        stainfo[find].associated = false;
+        stainfo[find].deauthPending = false;
+        goto printret;
+    }
+
+    if(sentNullPS_POLL && found)
+    {
+        stainfo[find].sentNullPS_POLL = true;
+        goto printret;
+    }
+
+    if(deauthPending && found)
+    {
+        stainfo[find].deauthPending = true;
+        goto printret;
+    }
+
+    if(resetDeauthPending && found)
+    {
+        stainfo[find].deauthPending = false;
+        goto printret;
+    }
+
+    if(!found && authenticated)
+    {
+        stainfo[freeIndex].stamac = addr;
+        stainfo[freeIndex].authenticated = authenticated;
+        stainfo[freeIndex].associated = false;
+        stainfo[find].aid = freeIndex;
+        do
+        {
+            freeIndex = (freeIndex + 1) & (MAX_STATIONS-1);
+            count++;
+        }while(stainfo[freeIndex].stamac != 0
+               && count <MAX_STATIONS);
+        goto printret;
+    }
+
```

```
+    if(found && powerSave)
+    {
+        stainfo[find].powerSave = true;
+        if(stainfo[find].powerSaveQueue == NULL)
+        {
+            stainfo[find].powerSaveQueue =
+            Queue::Default().Copy();
+        }
+        goto printret;
+    }
+
+    if(found && associated)
+    {
+        stainfo[find].associated = associated;
+    }
+
+ printret:
+    for(i=0;i<MAX_STATIONS;i++)
+    {
+        if(stainfo[i].stamac != 0)
+        {
+            // Display info
+        }
+    }
+    return;
+ }

***************
*** 2255,2260 ****
--- 3263,3305 ----
  }

+ // PS POLL_frame
+ PS_POLL_frame::PS_POLL_frame (Word_t aid, MACAddr rcv,
+                               MACAddr xmt, Long_t fcs_)
+   : L2Header802_11(aid, fcs_, PS_POLL)
+ {
+   fc.fc_proto_ver  = MAC_ProtocolVersion;
+   fc.fc_type       = MAC_Type_Control;
+   fc.fc_subtype    = MAC_Subtype_PS_POLL;
+   fc.fc_to_ds      = 0;
+   fc.fc_from_ds    = 0;
```

```
+   fc.fc_more_frag  = 0;
+   fc.fc_retry      = 0;
+   fc.fc_pwr_mgt    = 0;
+   fc.fc_more_data  = 0;
+   fc.fc_wep        = 0;
+   fc.fc_order      = 0;
+
+   ra   = rcv;
+   ta   = xmt;
+   L2Proto802_11::PS_POLL_Count++;
+ }
+
+ PS_POLL_frame::PS_POLL_frame(const PS_POLL_frame& f)
+   : L2Header802_11(f)
+ {
+   L2Proto802_11::PS_POLL_Count++;
+ }
+
+ Size_t PS_POLL_frame::Size() const
+ {
+   return ETHER_RTS_LEN;
+ }
+

***************
*** 2369,2391 ****
  }

  // DATA_frame
+ DATA_frame::DATA_frame(Word_t dur, MACAddr rcv,
+             MACAddr xmt, Word_t sqn, Long_t fcs_,
+             bool Null_Frame,bool SentNullPS_POLL)
    : L2Header802_11(dur, fcs_, DATA)
  {
    fc.fc_proto_ver  = MAC_ProtocolVersion;
    fc.fc_type       = MAC_Type_Data;
+   if(Null_Frame)
+       fc.fc_subtype    = MAC_Subtype_NULL_DATA;
+   else
+       fc.fc_subtype    = MAC_Subtype_DATA;
+
+
```

```
+   if(SentNullPS_POLL && Null_Frame)
+       fc.fc_more_data  = 1;
+   else
+       fc.fc_more_data  = 0;
+
```

# Bibliography


[1] IEEE Standard 802.11-1999. Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: *Wireless LAN Medium Access Control and Physical Layer Specifications.*, 1999.

[2] WiFi Gaining Traction. `http://www.businessweek.com/technology/tech_stats/wifi051003.htm`, 2005.

[3] IEEE P802.11 - TASK GROUP S. `http://www.ieee802.org/11/Reports/tgs_update.htm`, 2009.

[4] Camp, J.; Knightly, E., *The IEEE 802.11s Extended Service Set Mesh Networking Standard*, Communications Magazine, IEEE , vol.46, no.8, pp.120-126, August 2008.

[5] Hiertz, G.R.; Max, S.; Rui Zhao; Denteneer, D.; Berlemann, L., *Principles of IEEE 802.11s*, Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on , vol., no., pp.1002-1007, 13-16 Aug. 2007.

[6] Hiertz, G.R.; Max, S.; Yunpeng Zang; Junge, T.; Denteneer, D., *IEEE 802.11s MAC Fundamentals*, Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on , vol., no., pp.1-8, 8-11 Oct. 2007.

[7] Yang Xiang, Zhongwen Li, *An Analytical Model for DDoS Attacks and Defense* iccgi,pp.66, International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06), 2006.

[8] IEEE P802.11i. *Medium Access Control (MAC) Security Enhancements*, Amendment 6 to IEEE Standard for Information technology -Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11:Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[9] Bellardo, J. and Savage, S. 2003. *802.11 denial-of-service attacks: real vulnerabilities and practical solutions.* In Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12 (Washington, DC, August 04 - 08, 2003). USENIX Security Symposium. USENIX Association, Berkeley, CA, 2-2.

[10] J. R. Douceur. *The Sybil attack.* In First International Workshop on Peer-to-Peer Systems (IPTPS - 02), Mar. 2002.

[11] Martinovic, I., Zdarsky, F. A., and Schmitt, J. B. 2007. *Regional-based authentication against dos attacks in wireless networks.* In Proceedings of the 3rd ACM Workshop on QoS and Security For Wireless and Mobile Networks, October, 2007.

[12] Newsome, J.; Shi, E.; Song, D.; Perrig, A., *The Sybil attack in sensor networks: analysis & defenses*, Information Processing in Sensor Networks, 2004.

IPSN 2004. Third International Symposium on , vol., no., pp. 259-268, 26-27 April 2004.

[13] Bhuse, V.; Gupta, A.; Al-Fuqaha, A., *Detection of Masquerade Attacks on Wireless Sensor Networks Communications*, 2007. ICC '07. IEEE International Conference on , vol., no., pp.1142-1147, 24-28 June 2007

[14] Sanjay Ramaswamy, Huirong Fu, Manohar Sreekantaradhya, John Dixon,and Kendall E. Nygard. *Prevention of cooperative black hole attacks in wireless ad hoc networks*. International Conference on Wireless Networks,pp. 570-575, June 2003.

[15] Zhang, W.; Rao, R.; Cao, G.; Kesidis, G., *Secure routing in ad hoc networks and a related intrusion detection problem*, Military Communications Conference, 2003. MILCOM 2003. IEEE , vol.2, no., pp. 735-740 Vol.2, 13-16 Oct. 2003.

[16] The Georgia Tech Network Simulator
`http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/`

[17] W. Xu, W. Trappe, Y. Zhang, and T. Wood. *The feasibility of launching and detecting jamming attacks in wireless networks.* In Proceedings of ACM MOBIHOC, 2005.

[18] Y. Law, L. Hoesel, J. Doumen, P. Hartel, and P. Havinga. *Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols*. In Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005).

[19] IEEE Std 802.11b-1999, Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: *Higher-speed physical layer extension in the 2.4 GHz Band*, The Institute of Electrical and Electronics Engineers, 1999.

[20] MAC Enhancements for Quality of Service `http://grouper.ieee.org/groups/802/11/Reports/tge_update.htm`

[21] IEEE Std 802.11g-2003 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)specifications AMENDMENT 4: *Further Higher Data Rate Extension in the 2.4 GHz Band*

[22] IEEE Std 802.11f, *IEEE Trial-use Recommended Practice for Multi-vendor Access Point Interoperability via an Inter-access Point Protocol Across Distribution Systems Supporting 802.11 Operation* , 2003.

[23] IEEE 802.11h SMa - Spectrum Managed 802.11a `http://grouper.ieee.org/groups/802/11/Reports/tgh_update.htm`

[24] IEEE 802.11j 4.9 GHz - 5 GHz Operation in Japan `http://grouper.ieee.org/groups/802/11/Reports/tgj_update.htm`

[25] IEEE 802.11k Radio Resource Measurement Enhancements `http://grouper.ieee.org/groups/802/11/Reports/tgk_update.htm`

[26] IEEE 802.11n Standard for Enhancements for Higher Throughput `http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm`

[27] IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) http://grouper.ieee.org/groups/802/11/Reports/tgp_update.htm

[28] IEEE 802.11r Fast Roaming/Fast BSS Transition http://grouper.ieee.org/groups/802/11/Reports/tgr_update.htm

[29] IEEE 802.11w Protected Management Frames http://grouper.ieee.org/groups/802/11/Reports/tgw_update.htm

[30] IEEE 802.11z Direct Link Setup http://grouper.ieee.org/groups/802/11/Reports/tgz_update.htm

[31] IEEE Std 802.1X-2004, *802.1X: Port-Based Network Access Control*, IEEE, LAN/MAN Standard, 2004.

[32] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, Eds.,*Extensible Authentication Protocol (EAP), RFC 3748,* June 2004.

[33] B. Aboba and D. Simon, *PPP EAP TLS Authentication Protocol, RFC 2716,* 1999.

[34] S. Zhu, S. Setia, and S. Jajodia, *LEAP: efficient security mechanisms for large-scale distributed sensor networks*, in CCS -03: Proceedings of the 10th ACM Conference on Computer and Communications Security, NewYork, ACM Press, 2003, pp. 62-72.

[35] Microsoft's PEAP version 0 http://www.watersprings.org/pub/id/draft-kamath-pppext-peapv0-00.txt

[36] EAP-TTLS http://www.juniper.net/techpubs/software/aaa_802/sbrc/sbrc70/sw-sbrc-admin/html/EAP-024.html

[37] J. Daemen and V. Rijmen, *AES proposal: Rijndael,* in Proceedings of 1st AES Conference, August 1998.

[38] D. Whiting, R. Housley, and N. Ferguson, *Counter with CBC-MAC (CCM), RFC 3610,* September 2003.

[39] IEEE Std 802.3-2002, *802.3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE, LAN/MAN Standard, 2002.

[40] IEEE 802.16 Standard Group Website, `http://www.ieee802.org/16/`

[41] Bernard Aboba , *IEEE 802.1X Pre-Authentication*, 2002.

[42] Bahr, M. *Update on the Hybrid Wireless Mesh Protocol of IEEE 802.11s Mobile Adhoc and Sensor Systems*, 2007. MASS 2007. IEEE Internatonal Conference on Volume , Issue , 8-11 Oct. 2007 Page(s):1-6

[43] Won-Ju Yoon,; Sang-Hwa Chung,; Seong-Joon Lee,; Yun-Sung Lee, *An efficient cooperation of on-demand and proactive modes in Hybrid Wireless Mesh Protocol*, Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on , vol., no., pp.52-57, 14-17 Oct. 2008.

[44] Hiertz, G. R., Max, S., Wei, E., Berlemann, L., Denteneer, D., and Mangold, S. 2006. *Mesh technology enabling ubiquitous wireless networks: invited paper.* In Proceedings of the 2nd Annual international Workshop on Wireless internet (Boston, Massachusetts, August 02 - 05, 2006).

[45] Isabwe, Ghislain Maurice N. Kim, Kyeong Soo. *A novel approach to WLAN mesh interworking with multiple mesh portals.* Mobile Ad Hoc and Sensor

Systems, 2008. MASS 2008. 5th IEEE International Conference on Publication Date: Sept. 29 2008-Oct. 2 2008.

[46] Shiao-Li Tsao, *Research challenges and perspectives of voice over wireless LAN*, Emerging Information Technology Conference, 2005.

[47] Faccin, S.M.; Wijting, C.; Kenckt, J.; Damle, A., *Mesh WLAN networks: concept and system design*," Wireless Communications, IEEE ,April 2006.

[48] Qiang Shen; Xuming Fang, *A Multi-metric AODV Routing in IEEE 802.11 s*, Communication Technology, 2006. ICCT '06. International Conference on , Nov. 2006.

[49] Siddiqui, M.S.; Choong Seon Hong ;KyungHee Univ., Yongin; Xinghua Li; Weidong Yang; SangJae Moon; Jianfeng Ma, *Security Issues in Wireless Mesh Networks* ,Multimedia and Ubiquitous Engineering, 2007. MUE 2007.

[50] Redwan, H.; Ki-Hyung Kim, *Survey of Security Requirements, Attacks and Network Integration in Wireless Mesh Networks*, New Technologies, Mobility and Security, 2008. NTMS '08. , vol., no., pp.1-5, 5-7 Nov. 2008.

[51] Yanchao Zhang; Yuguang Fang, *ARSA: An Attack-Resilient Security Architecture for Multihop Wireless Mesh Networks*, Selected Areas in Communications, IEEE Journal on , vol.24, no.10, pp.1916-1928, Oct. 2006.

[52] Arbaugh, W.A.; Shankar, N.; Wan, Y.C.J.; Kan Zhang, *Your 80211 wireless network has no clothes*, Wireless Communications, IEEE , vol.9, no.6, pp. 44-51, Dec. 2002.

[53] Ferreri, F.; Bernaschi, M.; Valcamonici, L., *Access points vulnerabilities to DoS attacks in 802.11 networks*, Wireless Communications and Networking

Conference, 2004. WCNC. 2004 IEEE , vol.1, no., pp. 634-638 Vol.1, 21-25 March 2004.

[54] Borisov, N., Goldberg, I., and Wagner, D. 2001. *Intercepting mobile communications: The insecurity of 802.11.* In Proceedings of the 7th Annual international Conference on Mobile Computing and Networking (Rome, Italy). MobiCom 2001.

[55] Kyasanur, P.; Vaidya, N.H., *Detection and handling of MAC layer misbehavior in wireless networks*, Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on , vol., no., pp. 173-182, 22-25 June 2003.

[56] *The keyed-Hash Message authentication code (HMAC).* NIST. (2002, March.) `http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf`.

[57] Yong Sheng; Tan, K.; Guanling Chen; Kotz, D.; Campbell, A., *Detecting 802.11 MAC Layer Spoofing Using Received Signal Strength*, INFOCOM 2008. The 27th Conference on Computer Communications. IEEE , vol., no., pp.1768-1776, 13-18 April 2008.

[58] Xu, W., Trappe, W., and Zhang, Y. 2008. *Anti-jamming timing channels for wireless networks.* In Proceedings of the First ACM Conference on Wireless Network Security. April ,2008.

[59] Gummadi, R., Wetherall, D., Greenstein, B., and Seshan, S. 2007. *Understanding and mitigating the impact of RF interference on 802.11 networks.* SIGCOMM 2007.

[60] Xu, W., Trappe, W., Zhang, Y., and Wood, T. 2005. *The feasibility of launching and detecting jamming attacks in wireless networks.* In Proceedings of the

6th ACM international Symposium on Mobile Ad Hoc Networking and Computing (Urbana-Champaign, IL, USA, May, 2005).

[61] Zapata, M. G. and Asokan, N. 2002. *Securing ad hoc routing protocols*. In Proceedings of the 1st ACM Workshop on Wireless Security. September, 2002.

[62] Hu, Y., Perrig, A., and Johnson, D. B. 2003. *Rushing attacks and defense in wireless ad hoc network routing protocols*. In Proceedings of the 2nd ACM Workshop on Wireless Security. September, 2003. 1

[63] Gu, Q., Liu, P., Chu, C., and Zhu, S. 2007. *Defence against packet injection in ad hoc networks*. Int. J. Secur. Netw. 2, 1/2 (Mar. 2007).

[64] Gu, Q., Liu, P., and Chu, C. 2007. *Analysis of area-congestion-based DDoS attacks in ad hoc networks*. Ad Hoc Netw. July, 2007.

[65] Aad, I., Hubaux, J., and Knightly, E. W. 2004. *Denial of service resilience in ad hoc networks*. In Proceedings of the 10th Annual international Conference on Mobile Computing and Networking. October, 2004.

[66] Aad, I.; Hubaux, J.-P.; Knightly, E.W., *Impact of Denial of Service Attacks on Ad Hoc Networks*, Networking, IEEE/ACM Transactions on , vol.16, no.4, pp.791-802, Aug. 2008.

[67] Marti, S., Giuli, T. J., Lai, K., and Baker, M. 2000. *Mitigating routing misbehavior in mobile ad hoc networks*. In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking. August ,2000.

[68] Shao, M., Zhu, S., Cao, G., La Porta, T., and Mohapatra, P. 2008. *A cross-layer dropping attack in video streaming over ad hoc networks*. In Proceedings

of the 4th international Conference on Security and Privacy in Communication Netowrks (Istanbul, Turkey, September, 2008).

[69] Hu, Y., Perrig, A., and Johnson, D. B. *Ariadne: a secure on-demand routing protocol for ad hoc networks*. Wireless Networks Jan, 2005.

[70] Pirzada, A.A.; McDonald, C., *Secure Routing with the AODV Protocol*, Communications, 2005 Asia-Pacific Conference on ,Oct. 2005.

[71] 802.11 Packet Injection for Windows `http://www.codeproject.com/KB/IP/PacketReplay_11.aspx`

[72] Wi-Fi frame injection patch for Wireshark `http://802.11ninja.net/lorcon/wiki/WiresharkWiFiInjection`

[73] The Linux Home Page at Linux Online `http://www.linux.org/`

[74] Joshua Wright. *Detecting Wireless LAN MAC Address Spoofing*, GCIH, CCNA - Wednesday, 22 January 2003.

[75] Chatterjee, N.; Jaya Lakshmi, Y.; Potluri, A.; Negi, A., *Effect of Adapter Promiscuous Mode Operation on DSR Performance in MANETs*, Signal Processing, Communications and Networking, 2007. ICSCN '07.

[76] Wireshark `http://www.wireshark.org/`

[77] Wildpackets Airopeek `http://www.wildpackets.com/products/legacy_products/airopeek`

[78] Goldsmith, A.; Jafar, S.A.; Jindal, N.; Vishwanath, S., *Capacity limits of MIMO channels*, Selected Areas in Communications, IEEE Journal on , vol.21, no.5, pp. 684-702, June 2003

[79] Smith, C.; Matrawy, A., *Comparison of operating system implementations of SYN flood defenses (Cookies)*, Communications, 2008 24th Biennial Symposium on , vol., no., pp.243-246, 24-26 June 2008

[80] Levine, J.F.; Grizzard, J.B.; Owen, H.L., *Detecting and categorizing kernel-level rootkits to aid future detection*, Security & Privacy, IEEE , Feb. 2006

[81] Nazario, Jose; Holz, Thorsten, *As the net churns: Fast-flux botnet observations*, Malicious and Unwanted Software, 2008. MALWARE 2008.

[82] Information Sciences Institute, University of Southern California, *TRANSMISSION CONTROL PROTOCOL, RFC 793* September 1981.

[83] Hong Man; Yang Li; Xinhua Zhuang, *Video Transport Over Multi Hop Directional Wireless Networks*, Multimedia and Expo, 2006 IEEE International Conference on , vol., no., pp.1525-1528, 9-12 July 2006.

[84] Golden, S.A.; Bateman, S.S., *Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging*, Mobile Computing, IEEE Transactions on , vol.6, no.10, pp.1185-1198, Oct. 2007.

[85] GNU's Basic Cryptography Library. `http://www.gnupg.org/download/`