Denial-of-Sleep Vulnerabilities and Defenses in Wireless Sensor Network MAC Protocols

David Richard Raymond

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

Doctor of Philosophy
in
Computer Engineering

Dr. Scott F. Midkiff, Chair
Dr. Luiz A. DaSilva
Dr. Y. Thomas Hou
Dr. C. Patrick Koelling
Dr. Joseph G. Tront

March 25, 2008
Blacksburg, Virginia

Keywords: Wireless Sensor Networks, Security, MAC Protocols

Denial-of-Sleep Vulnerabilities and Defenses in Wireless Sensor Network MAC Protocols

David R. Raymond

(ABSTRACT)

As wireless sensor platforms become less expensive and more powerful, the promise of their wide-spread use for everything from health monitoring to military sensing continues to increase. Like other networks, sensor networks are vulnerable to malicious attack; however, the hardware simplicity of these devices makes defense mechanisms designed for traditional networks infeasible. This work explores the *denial-of-sleep* attack, in which a sensor node's power supply is targeted. Attacks of this type can reduce sensor lifetime from years to days and can have a devastating impact on a sensor network. This work identifies vulnerabilities in state-of-the-art sensor network medium access control (MAC) protocols that leave them susceptible to denial-of-sleep attack. It then classifies these attacks in terms of an attacker's knowledge of the MAC layer protocol and ability to bypass authentication and encryption protocols. Attacks from each category in the classification are modeled to show the impacts on four current sensor network MAC protocols: S-MAC, T-MAC, B-MAC and G-MAC. To validate the effectiveness and analyze the efficiency of the attacks, implementations of selected attacks on S-MAC and T-MAC are described and analyzed in detail.

This research goes on to introduce a suite of mechanisms designed to detect and mitigate the effects of denial-of-sleep attacks on sensor networks. The **C**lustered **A**nt**i** **S**leep-Deprivation for **S**ens**o**r **N**etworks, or *Caisson*, suite includes a lightweight, platform-independent anti-replay mechanism, an adaptive rate-limiter and a jamming detection and mitigation mechanism. These tools are designed to be applied selectively or in concert to defend against denial-of-sleep attacks depending on the specific vulnerabilities in the MAC protocol used in a particular sensor network deployment.

This work makes two major contributions to state-of-the-art wireless sensor network research. First, it fully explores the denial-of-sleep attack, to include the implementation of a subset of these attacks on actual sensor devices and an analysis of the efficiency of these attacks. Sec-

ond, it provides a set of tools by which these attacks are detected and defeated in a lightweight, platform-independent, and protocol-independent way. If sensor networks are to live up to current expectations, they must be robust in the face of newly emerging network attacks, to include denial-of-sleep.

Acknowledgments

This work would not have been possible without the assistance, support, and collaboration of a great many people. First, I must thank my adviser, Dr. Scott Midkiff for his mentorship and support over the past two years. Throughout this research, Dr. Midkiff has always provided the perfect combination of encouragement and guidance and his enthusiastic involvement in this work has been invaluable.

I also must thank my advisory committee, Dr. Joseph Tront, Dr. Patrick Koelling, Dr. Luiz DaSilva, and Dr. Thomas Hou, for their insightful comments and recommendations during my research. Special thanks goes to LTC Michael Brownfield for his for his generous collaboration during the early stages of this research and for his guidance throughout the Ph.D. process. The value of Mike's contributions cannot be overstated.

I must also thank the many ECE students and faculty that I have collaborated with during my time at Virginia Tech. Foremost among those are Randy Marchany, Triiip Bowen, Ingrid Burbey, Youping Zhao, and Timothy Buennemeyer.

I could not have been successful without the loving support of my wife, Erin, and our two children, Lucas and Ainsley. Their encouragement and patience have been a godsend.

Finally, I must thank the United States Army and the United States Military Academy Department of Electrical Engineering and Computer Science for affording me the opportunity to pursue this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

You don't know what you can get away with until you try.

- GEN Colin Powell

As new applications for networked devices emerge, protocols must evolve to support new computing platforms and environments. This evolution has been observed at all levels of the protocol stack. Network routing protocols, for example, evolved through the 1970s and 1980s to be efficient as the Internet grew to include millions of interconnected devices [1]. The increased popularity of ad hoc networks has led to new routing protocols designed to correctly forward traffic through these infrastructure-less networks. A similar evolution has taken place at the link layer. In wired networks, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) has proved to be effective and this mechanism, codified in the IEEE 802.3 standard [2], has emerged as the leading link-layer protocol for these networks. As wireless networking increased in popularity, new protocols were developed to ensure efficiency in this unique environment. The IEEE 802.11 [3] standard provided solutions to problems not seen in wired networks, such as the Distributed Coordination Function (DCF), which was introduced to control access to the wireless medium.

In recent years, research into wireless sensor networks (WSN) has increased rapidly and a new design criterion has become dominant in the development of link-layer protocols for these networks: energy efficiency. By using sophisticated algorithms, new medium access control (MAC) protocols designed specifically for WSN take advantage of the generally low throughput demands

of these networks to keep devices in a sleep state and, thereby, decrease energy consumption by orders of magnitude. By waking only periodically to exchange traffic, these protocols trade increased network latency for energy savings.

A traditional problem in network protocol development is that security requirements are often ignored as protocols are developed to meet design criteria such as low overhead and maximum efficiency. This shortcoming has plagued the development of WSN MAC protocols as well. A thorough analysis of wireless sensor network MAC protocols has identified security vulnerabilities not previously explored in the research community. As discussed in Chapter 4, the very mechanisms that ensure energy efficiency can be exploited by malicious intruders to keep these devices in a state that maximizes energy consumption, thereby reducing the life of the devices by rapidly draining batteries.

This research has investigated vulnerabilities in WSN MAC protocols that leave them open to resource-consumption attacks aimed at preventing the radio from entering sleep mode in order to rapidly drain energy reserves on the attacked devices, referred to herein as *denial-of-sleep* attacks. Furthermore, it has investigated techniques for defeating or mitigating the effects of these potentially devastating network attacks.

This chapter introduces the denial-of-sleep problem, along with research objectives and questions, and provides an overview of the remainder of this document. Section 1.1 presents the research problem under investigation. A brief discussion of denial-of-sleep is given in Section 1.2. Section 1.3 states the objectives of this research and Section 1.4 enumerates the questions that are answered in this work. Section 1.5 outlines the methodology that was used during this research. Finally, Section 1.6 provides the organization for the remainder of this document.

1.1 Problem Statement

The purpose of this research is twofold. First, it explores a class of vulnerabilities in wireless sensor networks in which sensor devices' energy resources can be rapidly drained by a malicious host by keeping the devices' radios active when they would otherwise be in sleep mode to conserve

energy. Several contention-based MAC protocols have been proposed and implemented for WSN that strive to minimize energy consumption by keeping the radios in sleep mode as much as possible. Unfortunately, the techniques that these protocols use to minimize energy consumption can be exploited by malicious users who understand the characteristics of the MAC protocols, even in networks that use encryption and authentication to secure network traffic. This research analyzes the functionality of a representative group of WSN MAC protocols and, from that analysis, generalizes the vulnerabilities to describe a classification mechanism for MAC-layer resource-consumption attacks on WSN.

This work goes on to design, implement, and evaluate an innovative suite of low-overhead, platform-independent, cross-layer mechanisms that will mitigate the impact of these resource consumption attacks on sensor networks that can be used with a wide range of existing and future WSN MAC protocols. The **C**lustered **A**nt**i** **S**leep-Deprivation for **S**ens**o**r **N**etworks, or *Caisson*, suite incorporates a low overhead anti-replay mechanism, an adaptive rate-limiting mechanism, and jamming detection and mitigation to identify and reduce the affects of network traffic that can be used to maliciously keep sensor node radios awake. These mechanisms have the added benefit of mitigating and, in some cases, preventing other types of network attacks such as replay attacks and some denial-of-service attacks in wireless sensor networks.

1.2 Background and Motivation

Before this research, energy-efficient MAC protocols developed to improve lifetimes of sensor network deployments had not been analyzed in detail to identify vulnerabilities that might leave them open to resource-consumption attacks. An understanding of the vulnerabilities introduced by mechanisms designed to reduce energy consumption is essential for two reasons. First, only by understanding these vulnerabilities can we develop techniques for identifying attacks that attempt to take advantage of them and implement mechanisms to mitigate these attacks. Second, such an understanding is necessary as future protocols are developed so that they can be designed to be immune to these types of attacks.

The contention-based sensor network MAC protocols examined in this research are representative of the energy-efficient techniques used in current WSN MAC protocols and each of them has features that leave it vulnerable to denial-of-sleep. These vulnerabilities and some specific techniques that can be used to exploit them are detailed in Chapter 4. While some of the most efficient attacks can be mitigated by encrypting and authenticating WSN traffic, doing so does not protect against all denial-of-sleep attacks. In fact, even when traffic is authenticated, each of the protocols examined here is still susceptible to attacks that keep WSN platform radios in receive mode permanently, which causes batteries to drain at a rate that is orders of magnitude faster than under routine traffic.

Current security research in sensor networks primarily focuses on the development of encryption and authentication mechanisms tailored to the resource constraints of today's sensor platforms. Such work is important and it forms the foundation of denial-of-sleep prevention described in this research. Encryption and authentication alone, however, are not sufficient to mitigate the effects of these attacks. To prevent the potentially devastating effects of denial-of-sleep, anomalous network activity indicating that such an attack is ongoing must trigger mechanisms to prevent further battery exhaustion. Some of the vulnerabilities investigated in this research involve message replay. While anti-replay mechanisms for sensor networks have been researched, this problem had not previously been solved in a low-overhead, platform-independent way. Other indicators of denial-of-sleep attack are specific forms of jamming and high rates of unauthenticated traffic. Current research on intrusion detection systems (IDS) for sensor networks investigates techniques to characterize network traffic and then notify a system administrator when anomalous traffic is detected. The memory and processing power required to execute these anomaly-based IDS engines is prohibitive for most sensor platforms. Furthermore, attempting to alert an administrator of anomalous network behavior is useless in an unmonitored network and futile in a jammed one.

This research describes the design of a lightweight, fully distributed, rules-based intrusion detection mechanism that identifies and mitigates sensor network denial-of-sleep attacks, regardless of the platform or MAC protocol used in the network and without user intervention.

1.3 Research Objectives

The fundamental goals for this research were to identify MAC-layer resource-consumption vulnerabilities in WSN and develop and analyze low-overhead, platform-independent mechanisms for mitigating these attacks. The Caisson suite extends sensor lifetime in the face of resource consumption attacks aimed at keeping the radio awake when it should be sleeping to conserve energy. These mechanisms have low overhead in terms of energy consumption, memory and processing requirements, and added network traffic. They easily integrate into existing WSN protocol stack implementations, such as the University of Southern California's Information Sciences Institute (USC/ISI) sensor network protocol stack [4]. Finally, the suite is user-configurable so that specific protections can be enabled or disabled depending on the characteristics of the network and the level of protection desired.

Specific performance goals for the Caisson system were to limit network lifetime overhead to 5% or less when the network is not under attack and to preserve 90% or more of the expected network lifetime when the network is attacked. This research has shown that sensor network lifetime overhead goals are met by Caisson under routine network conditions. Network lifetime overheads range from 0% to 4.6%, depending on the WSN platform and MAC protocol used in the network. The techniques used to identify and react to denial-of-sleep attacks can be configured to favor network lifetime or throughput. If network lifetime is favored, the goal of preserving 90% of original network lifetime in the face of denial-of-sleep attack is easily met. When parameters are set to preserve both network lifetime and throughput, the Mica2 implementation of the Caisson suite is able to maintain 71% or more of the expected network lifetime while still allowing 64% to 96% of expected network throughput in the face of denial-of-sleep attack, depending on the MAC protocol used in the network and network traffic levels.

1.4 Research Questions

This research addresses the following questions.

1. What denial-of-sleep vulnerabilities exist in WSN MAC protocols that can lead to rapid energy consumption?
   (a) What are the impacts of denial-of-sleep attacks on sensor network lifetime?
   (b) What are the impacts of denial-of-sleep attacks on network throughput?
   (c) How efficient are these attacks in terms of attacker energy consumption?
   (d) Can MAC protocols be distinguished via network traffic analysis so that attacks can be easily tailored to the MAC protocol being used? What other information can be gleaned from analysis of network traffic that can help an intruder tailor a network attack?
2. Can these attacks be mitigated in a low-overhead, platform-independent, MAC protocol-independent manner?
3. What are the benefits of the Caisson suite of denial-of-sleep mitigation mechanisms in the face of denial-of-sleep attacks?
   (a) How well does Caisson extend network lifetime versus that of an unprotected WSN?
   (b) How well does Caisson preserve network throughput during an attack?
4. What are the costs of Caisson in a network that is not under attack?
   (a) How does network lifetime compare to a WSN without Caisson?
   (b) What is the sensor node memory overhead when compared to a WSN without Caisson?

1.5 Methodology Overview

This research was conducted in three phases. Phase I involved an analysis of WSN MAC protocols for denial-of-sleep vulnerabilities, analytical modeling of the impacts of these attacks on state-of-the-art WSN MAC protocols, and the implementation of a subset of these attacks to show effectiveness and efficiency. Phase II focused on the development of the Caisson mechanisms, using Jain's ten-step performance evaluation methodology [5] which is described in Chapter 3. The

third phase of this work included testing and validation of the denial-of-sleep mitigation mechanisms. The mechanisms were developed in simulation using the OPNET simulation environment. The sensor MAC protocol models contain statistics-collecting variables to monitor sleep and awake time and energy consumption, throughput, and delay so that accurate statistics could be taken. The mechanisms were then tested in simulation across a representative set of MAC protocols and sensor platforms and against a variety of denial-of-sleep attacks. Finally, the Caisson system was validated through implementation on the Crossbow Mica2 wireless sensor platform running the Sensor MAC (S-MAC) [6], Timeout MAC (T-MAC) [7], and Berkeley MAC (B-MAC) [8] wireless sensor network protocols and tested against attacks on these protocols.

1.6 Organization of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides background in the area of wireless networking and WSN MAC protocols and surveys related work in the area of wireless and sensor network security. Chapter 3 describes the methodology that was used to design and test mechanisms for mitigating these attacks. Chapter 4 provides an analysis of denial-of-sleep vulnerabilities in current WSN MAC protocols and describes the effects of denial-of-sleep attacks on wireless sensor networks. Chapter 5 reviews the classes of denial-of-sleep prevention categories and provides an overview of the Caisson suite of tools for denial-of-sleep attack defense and mitigation. Chapter 6 describes clustered anti-replay protection, a technique for providing reliable, low-overhead anti-replay protection in sensor networks. Chapter 7 introduces adaptive rate-limiting as a technique for mitigating unauthenticated broadcast or replay attacks in WSN. Chapter 8 discusses implementation details and results for a reliable jam detection technique for current WSN platforms. It goes on to discuss a simple mitigation scheme called *random hibernation*, which conserves sensor energy in the face of jamming attacks. Finally, Chapter 9 summarizes the results provided herein, discusses potential directions for future research in the area of WSN denial-of-sleep defense, and offers concluding remarks.

Chapter 2

Overview of Sensor Networks, MAC Protocols, and Security

Prepare for the unknown by studying how others in the past
have coped with the unforeseeable and the unpredictable.
- GEN George S. Patton

This chapter provides necessary background on sensor networks and the characteristics of WSN MAC protocols, along with an introduction to sensor network security and intrusion detection systems (IDS). Section 2.1 gives an overview of WSNs, to include sensor platforms and the protocol stack used for wireless communication. Section 2.2 provides a synopsis of wireless networking standards, describing some of the techniques that are shared by WSN MAC protocols for medium access. Section 2.3 focuses on the sources of energy loss in sensor network communications and describes the sensor network MAC protocols used in this research. Section 2.4 discusses clustering in WSN as a technique for routing traffic and energy conservation and Section 2.5 provides a description of current research on sensor network security, to include an introduction to intrusion detection in WSN. Section 2.6 explores related research on resource consumption attacks on wireless handheld devices and Section 2.7 provides a chapter summary.

2.1 Wireless Sensor Networks

WSN are made up of tens to potentially thousands of small, low-power sensor devices designed to sense information about their environment and then transmit that information to other network nodes or to a base station. Research involving these devices has proposed a wide range of applications, to include atmospheric monitoring, wildlife tracking, physical perimeter intrusion detection, medical monitoring, homeland security, nuclear, biological, and chemical (NBC) monitoring, and a wide range of military applications [9, 10, 11].

Sensor network protocols must span the range of potential deployment scenarios, from sparse networks in which maintaining inter-node communication is a challenge, to extremely dense ones where access to the wireless channel must be carefully arbitrated to avoid collisions and overhearing. While these networks share many of the same characteristics as wireless networks based on the IEEE 802.11 standard, commonly known as Wi-Fi networks, they are unique in many ways. The following sections describe the characteristics of sensor node platforms that make them different from other wireless devices and explore the implications of these differences in terms of the network protocols stack and, particularly, MAC protocol behavior.

2.1.1 Sensor Network Platforms

Sensor platforms tend to be simple and relatively inexpensive so that they can be deployed in large numbers. An extremely small form factor is also preferred for many applications to prevent node destruction and tampering. In fact, the *Smart Dust* project at the University of California, Berkeley, seeks to develop wireless sensor nodes, to include sensor, processor, radio, and battery, that are one cubic millimeter in size [12]. Many deployment scenarios, particularly military ones, have the devices being abandoned once their power supplies are exhausted. Three important goals, therefore, are to keep the devices small, inexpensive, and as energy-efficient as possible. One cannot expect Moore's Law to lead to enhanced performance. Rather, technological improvements are likely to drive down size and cost while maintaining current capabilities.

Several sensor platforms have been developed in the last few years. Appendix A gives

a listing of currently available platforms. These devices are characterized by 8-bit processors, around 100 kilobytes of program memory, and 2 to 10 kilobytes of random access memory (RAM). Some of the more robust devices have more memory, however, this added functionality comes at the expense of a larger form factor and higher cost. The amount of RAM also affects energy consumption. RAM requires constant current to maintain stored data, and more RAM consumes more energy. Since the data stored in RAM must be maintained while the devices are in the sleep state, increased RAM results in an increase in sleep-state energy consumption and, therefore, reduces network lifetime. Of the 19 devices known to be on the market, five of them use the IEEE 802.15.4 compliant CC2420 radio [13], which is, at the time of this writing, the only *ZigBee*-compliant radio currently used on sensor platforms. Three others use IEEE 802.15.1 *Bluetooth* [14] radios, and the remainder use lower bit-rate radios operating in the 868 - 914 MHz frequency band, such as the ChipCon CC1000 [15].

There are a handful of sensor platform operating systems (OS), the most popular being TinyOS [16], which runs on 14 of the 19 platforms listed in Appendix A. Other examples include SOS [17], MANTIS [18], and Contiki [19]. TinyOS, the most widely-used sensor network OS and the one used in this research, is a component-based operating system written in the nesC programming language [20]. Its modular architecture is ideal for the resource constraints inherent to WSN platforms. Only required components are wired into an application using a specification file. Components provide functionality via *interfaces* and take advantage of other components' functionality via *calls* to their interfaces. Functions are either *events*, which are triggered by an occurrence such as a preamble being sensed by the physical layer or a send call returning, or *tasks*, which are similar to function calls in the C programming language. When an application is compiled, only the specific OS components necessary for that application are included so as to keep code footprint as small as possible.

Table 2.1 gives energy consumption, memory, and other data for the Crossbow Mica2™ [21] and Tmote™ Sky [22] sensor nodes, both of which use the TinyOS operating system. These are the primary platforms used in this research as they are representative of the wide range of available wireless sensor platforms.

Table 2.1: Sensor Platform Power Consumption and Sleep Transition Data

		Mica2™ [21]		Tmote™ Sky [22]	
Power draw	Receive	36.81 mW		64.68 mW	
	Transmit	87.90 mW		55.20 mW	
	Sleep	0.09 mW		0.114 mW	
RAM		4 KB		10 KB	
Program memory		128 KB		48 KB	
RF transceiver		CC1000		CC2420	
Data rate		76.8 kbps		250 kbps	
Sleep to RX transition		2.45 ms	0.095 mW	3.13 ms	0.018 mW
RX/TX transition		0.25 ms	0.016 mW	1.52 ms	0.009 mW
RX to sleep transition		0.10 ms	0.002 mW	2.16 ms	0.012 mW

2.1.2 Sensor Network Protocol Stack

As in other network devices, the seven-layer Open Systems Interconnect (OSI) layered model is used to generally describe sensor network protocol functionality. The traditional seven layers in the OSI model are often reduced to the following five layers: physical, link, network, transport, and application [23]. These are the same layers used by Ye, Heidemann, and Estrin to describe the USC/ICI layered model for WSN [4], which is an example of a layered abstraction for sensor protocols.

In the OSI model, the physical layer is responsible for moving bits between directly connected network nodes. Connection types might be wired, using twisted pair or coaxial cable, or wireless. Physical layer standards dictate encoding mechanisms and timings for this direct, inter-node communication. In traditional wireless networks, if the RF transceiver is not transmitting data, it is in receive mode, regardless of whether there is data being transmitted by another node or not. In a sensor network, the radio can be in one of three states: transmit, receive, or off.

This research primarily focuses on the the link layer, which is responsible for moving link-layer packets, called *frames*, from one networked device to the next hop along the path to the destination. At each node along the path, a frame is passed from the network layer down to the link layer for transmission to the next node. This layer controls access to the physical medium,

so in a wireless network it is in the best position to know when nodes should be transmitting or receiving, and when they can turn the radio off and enter sleep mode. Link layer protocols for sensor networks are more thoroughly explored in Section 2.3.

The network layer provides end-to-end routing of network layer packets, often called *datagrams*. In traditional networks, this layer provides two basic services: it dictates the datagram format and how the various datagram fields are interpreted by hosts and routers in the network, and it provides protocols for routing datagrams between hosts. In sensor networks, packet format is often dictated by the application to minimize packet size and maximize processing efficiency. Sensor networks do, however, use a variety of protocols to route data from sensor platforms to base stations, sometimes aggregating data along the way for communication efficiency.

The transport layer generally provides end-to-end connection-oriented or connectionless transmission of data packets between hosts. It handles such tasks as guaranteeing delivery of application-layer packets and provides services such as congestion control within the network. The most widely supported transport layer protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Sensor platforms do not generally support a separate transport layer, instead incorporating this layer into the deployed application for efficiency.

The application layer provides the functionality for which a computing device is deployed. In wired networks, application layer protocols include Hypetext Transfer Protocol (HTTP) for web documents, File Transfer Protocol (FTP) for file transfer, and Simple Mail Transfer Protocol (SMTP) for email communication. WSN application protocols are usually specific to the sensor network deployment and their packet headers are minimalistic in nature, providing the minimum required data for transmission through the network to another node or to a base station.

2.2 Wireless Networking Standards

Sensor network protocols borrow techniques for collision avoidance and medium access from current IEEE 802 networking standards. A basic understanding of some of these techniques is, therefore, necessary to understand the features of the WSN protocols described later in this section.

Knowledge of power conservation techniques used by these protocols is also important to understand why these protocols are insufficient for WSN deployments.

2.2.1 IEEE 802.11

The IEEE 802.11 standard specifies wireless local area network (LAN) MAC and physical layer specifications for wireless network devices. This protocol allows wireless devices to connect to IEEE 802 networks, accounting for peculiarities in wireless medium access and mobile nodes. Nodes can connect to a network via an access point (AP), through which all traffic to and from the node travels, or in a peer-to-peer fashion in an ad hoc network. These two modes of connectivity are referred to as *infrastructure mode* and *ad hoc mode*. The characteristics of wireless connectivity addressed by IEEE 802.11 are the relative unreliability of wireless links caused by path loss, interference, and multipath propagation, and the *hidden node problem* described below. Link reliability is achieved via link-layer acknowledgments, a technique also used in most WSN MAC protocols.

The hidden node problem describes the situation depicted in Figure 2.1, in which nodes A and C, which are out of radio range of each other due to distance or obstacles, simultaneously transmit data to node B. Even if nodes A and C were equipped with collision-detecting network cards, they would not be able to detect the collision at node B because neither of them is able to receive the other's signal.

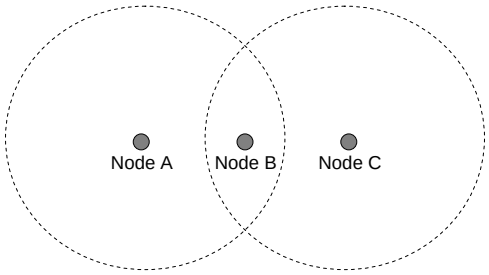


Figure 2.1: The hidden node problem. The dotted lines indicate the transmission ranges of nodes A and C.

IEEE 802.11 Distributed Coordination Function (DCF)

The IEEE 802.11 distributed coordination function (DCF) dictates distributed medium access rules such as carrier sensing and random backoffs before transmission. It also incorporates a request-to-send (RTS), clear-to-send (CTS) exchange for unicast traffic, which is used to address the hidden node problem. When node A has traffic for node B, it transmits a short RTS message indicating the length of the message to be transmitted. If node B knows of no other ongoing transmissions, it replies with a CTS message, which indicates to node A that it may send the packet, and advertises to other nodes within transmission range of node B that they should not transmit during the duration of node A's transmission. IEEE 802.11 control packets include a field specifying the duration of the data frame to be transmitted. When other nodes overhear an RTS or CTS message, they record this value, referred to as a *network allocation vector* (NAV), and defer transmission for the duration of the NAV. This mechanism is known as *virtual carrier-sensing*. Figure 2.2 depicts a typical NAV scenario. Once the data packet is received, node B transmits an acknowledgment (ACK), indicating that the packet has been received and informing other nodes that its exchange with node A is complete. The short interframe space (SIFS) and distributed interframe space (DIFS) are delays between transmissions that help to prioritize access to the wireless medium. A SIFS is a short delay that is used when a node with a high priority frame to send, such as a CTS or ACK. A node that wants to initiate a unicast exchange or send a broadcast packet waits for the duration of

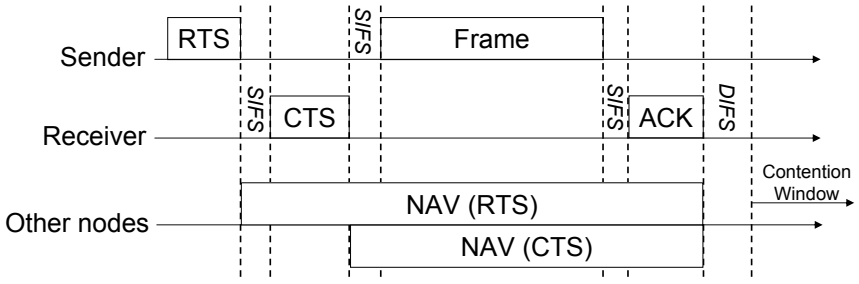


Figure 2.2: A typical NAV scenario. SIFS is the wireless protocol's short interframe space. DIFS is the distributed interframe space. These interframe delays are used to coordinate access to the wireless medium.[1]

[1]Copyright IEEE. Used through residual rights retained from [24].

a DIFS. There is still the potential for collision of RTS messages, but because these messages are short, the collision probability is low and the penalty is small if a collision does occur.

While the DCF provides collision avoidance, it does not always allow fair access to the wireless medium among network nodes. This can lead to unacceptable message latency if a node is not able to access the wireless channel.

Point Coordination Function (PCF)

To increase fairness and reduce latency in infrastructure networks, the IEEE 802.11 point coordination function (PCF) can be used if it is included in the network devices' IEEE 802.11 implementations. Under PCF, time is divided into configurable contention-free periods and contention periods. The point coordinator, normally an AP, controls medium access during the contention-free period by first transmitting a beacon frame to synchronize nodes and gain control of the medium, and then by polling each station for traffic. At the end of the contention-free period, the point coordinator transmits a control packet indicating that nodes can use the DCF to exchange traffic until the next contention-free period. This mechanism guarantees that each node will be able to send its traffic with a specified maximum delay.

IEEE 802.11 Power Management

Wireless nodes can conserve energy by turning off their RF transceivers. Power management in 802.11 infrastructure mode is achieved by buffering data at the AP, which must always remain awake to respond to new network devices that might try to enter the network. If an AP receives traffic for a sleeping node, it will buffer the traffic. Network nodes must awaken at a specified time for a beacon frame and a traffic indication message (TIM) broadcast by the AP. The beacon frame is used for node synchronization and to advertise the existence of the network to devices entering the area, and the TIM indicates which nodes have traffic buffered at the AP. Nodes for which there is waiting traffic can poll the AP for the data, and then return to reduced-power mode after receiving the data from the AP. Power conservation in PCF networks is similar to that in DCF networks, although nodes must be awake for beacon frames, for TIM messages, and for the portion

of the contention-free period when they are receiving or sending traffic.

Power management in ad hoc networks is not as efficient as it is in infrastructure networks because each sending node is responsible for ensuring that the receiving node is awake before transmitting data. When nodes organize into an ad hoc network, they use a distributed beacon process to ensure synchronization. At the beginning of the beacon window, each node selects a random backoff between zero and twice the minimum contention window for the medium. Whichever node's timer expires first transmits a beacon, and then it must remain awake for the entire beacon period to listen for new nodes trying to enter the network. Nodes that have traffic to send will transmit an *announcement traffic indication message* (ATIM), announcing stations for which they have traffic. The period between beacon transmissions is divided into multiple ATIM windows, during which each node that has traffic will contend for the medium so that it can send out ATIM messages. Each node in the network must awaken for these ATIMs in case the node sending the ATIM has traffic for it, resulting in an increased percentage of awake time than in infrastructure networks.

2.2.2 IEEE 802.15.1

The IEEE 802.15.1, or *Bluetooth* standard provides physical and MAC layer specifications for wireless personal area networks (WPAN) [14]. Bluetooth was designed to replace the wires and cables connecting peripheral devices while not sacrificing security. Bluetooth devices form small networks referred to in the specification as *piconets*. In a Bluetooth piconet, up to seven slave devices can be connected in a star topology to a master device.

Bluetooth devices transmit data at up to 3 Mbps, using frequency hopping to reduce the effects of interference and background noise. The added complexity of the frequency hopping radio makes it more expensive and less energy efficient than simpler, single frequency devices, which makes them less desirable for WSN platforms. ZigBee devices, discussed in the next section, do not suffer this same limitation.

2.2.3 IEEE 802.15.4

The IEEE 802.15.4, or *ZigBee* [25], standard provides MAC and physical-layer specifications for low-rate wireless personal area networks (WPAN), similar to Bluetooth. ZigBee devices are designed to be more energy efficient than Bluetooth, however, at lower data rates and with less sophisticated security mechanisms. The ZigBee standard was designed for low-rate, low power, applications such as private and industrial automation and infrastructure-based sensor networks, although it can be adapted for ad hoc WSN deployments. The maximum data rate for compliant devices is 250 kbps, compared with rates of up to 540 Mbps for the IEEE 802.11n wireless standard.

Power conservation mechanisms for infrastructure-based IEEE 802.15.4 networks mirror those for PCF-based IEEE 802.11 networks. Techniques for power management in multi-hop ad hoc networks, which is the proposed deployment paradigm for most sensor networks, are not addressed in the standard. Energy-efficient protocols, such as those discussed in the next section, can be layered on top of the IEEE 802.15.4 MAC to provide power management in these scenarios.

2.3 Medium Access Control Protocols

MAC protocols dictate frame formats and are used at the link layer to arbitrate access to the communication channel. Well-known MAC protocols include Ethernet [2] and the MAC used in the IEEE 802.11 (Wi-Fi) [3] family. This section discusses the various types of MAC protocols, then introduces the WSN MAC protocols considered in this research.

2.3.1 MAC Protocol Classifications

MAC protocols can be broken down into three basic classifications: channel partitioning protocols, taking-turns protocols, and random access protocols [23].

Channel Partitioning MAC Protocols

Channel partitioning techniques, such as time division multiplexing (TDM) and frequency division multiplexing (FDM), partition the channel among all nodes sharing that channel. The advantage of these protocols is that if the channel is divided equally among the nodes, the protocol is completely fair, allowing each node equal access to the channel. Furthermore, if nodes are properly synchronized and each node follows the protocol properly, there are no frame collisions. There are, however, disadvantages. Since the channel is partitioned and each node can only transmit within its specified frequency or timeslot, bandwidth is wasted if nodes do not have the same amount of traffic to transmit. In a TDM network, nodes must wait for their turn to transmit, adding delay to the transmissions. These shortcoming are exacerbated in WSN networks, in which nodes attempt to minimize the time that their radios are on. In a TDM network, for example, potential receivers must remain awake for all timeslots in case any of the other nodes in the network has traffic for them. TDM protocols also require fine-grained synchronization. Such synchronization can be achieved in WSN using techniques such as Reference Broadcast Synchronization (RBS) [26], however, these protocols have high control traffic and memory overhead. For these reasons, channel partitioning protocols are generally not ideal for large-scale WSN deployments.

Taking-Turns MAC Protocols

An example of this type of MAC protocol is a polling protocol, where a master node polls each node in the network, asking if it has traffic to send. Nodes are generally polled in a round-robin fashion and nodes that have traffic may only transmit when they are polled. This type of protocol avoids collisions, since nodes are only allowed to transmit when they are specifically given access to the channel. Polling, however, adds additional delay as transmitters wait for the poll. Another disadvantage is the requirement for a master node or clusterhead to conduct the polling. Energy is wasted by the master node, which must constantly poll each slave to determine whether they have traffic. Slave nodes must also awaken for each poll to receive any traffic destined for them from the polled node.

Random Access MAC Protocols

Networks that use random access protocols do not restrict channel access to specific time slots or sub-frequencies. Nodes that have traffic to transmit simply transmit their data. Some of these protocols are slotted in that nodes must wait for the next time slot before transmitting, but nodes are not restricted to particular time slots. A well-known example of such a protocol is slotted Aloha [23]. This channel access technique reduces packet delay and increases utilized bandwidth in networks where nodes have varying traffic loads. The nature of these networks also allow some of the power-saving features of the WSN MAC protocols that are introduced in Section 2.3.3.

Modern random access MAC protocols for wired networks are designed to minimize collisions and to maximize throughput and fairness, usually using carrier sensing to ensure the medium is free before packets are transmitted. In the case of a collision, a random backoff mechanism is used to avoid future collisions. This combination of features describes the common *carrier sense multiple access with collision detection* (CSMA/CD) MAC protocol used in most wired networks and codified in the IEEE 802.3 standard [3].

MAC protocols for wireless networks do not use CSMA/CD for two primary reasons. First, detecting collisions requires that a network interface sense the medium while transmitting. Since the transmitted signal would normally overpower the received signal, a wireless network card that could detect collisions would be difficult to produce and expensive [23]. Second, wireless networks suffer from the hidden node problem, described in Section 2.2.1.

All WSN MAC protocols considered in this research use carrier sensing to avoid collisions. While some of these protocols use RTS/CTS exchanges to avoid the hidden node problem, others do not to reduce control packet overhead and protocol complexity. WSN MAC protocols also differ from traditional wireless network protocols in their endeavor to minimize energy consumption. Research by Sohrabi, et al., has shown that in WSN, communications dominate energy consumption [27]. WSN MAC protocols, therefore, attempt to minimize energy consumption by keeping the radio in sleep mode as much as possible. The following sections explore sources of energy loss in sensor networks and describe current WSN MAC protocols and corresponding energy-saving techniques.

2.3.2 Sources of Energy Loss in Sensor Nodes

MAC layer protocols designed for WSNs use various algorithms to save battery power by placing the radio in low-power modes when not actively sending or receiving data. Table 2.1 illustrates the importance of maximizing a node's sleep ratio because the transmit and receive power can be up to three orders of magnitude greater than the sleep power. Let the sleep ratio, or R_{sleep}, equal $T_{sleep}/(T_{active} + T_{sleep})$, where T_{active} and T_{sleep} are active time and sleep time. A node's lifetime is:

$$T_{sensorlife} = \frac{C_{battery(mWh)}}{(R_{sleep})\left(P_{sleep(mW)}\right) + (1 - R_{sleep})\left(P_{active(mW)}\right)}, \tag{2.1}$$

where P_{active} and P_{sleep} are active mode power draw and sleep mode power draw and $C_{battery}$ is the total amount of available energy. Depending on the platform, P_{active} can be 2 to 3 orders of magnitude greater than P_{sleep}, so it is important to keep nodes in sleep mode as much as possible. The Tmote Sky consumes 64.68 mW in receive mode and 0.114 mW in sleep mode [22]. Using two standard 3,000 mAh AA batteries, it will last 3,300 days in sleep mode, but only 5.8 days in receive mode. The disparity between receive cost and sleep cost leads to an exponential increase in network lifetime as sleep time increases, suggesting that an attack that decreases sleep time by even a few percentage points can have a dramatic impact on network lifetime.

The amount of energy that can be saved depends largely on the MAC protocol's ability to overcome the radio's four primary sources of energy loss: *collisions, control packet overhead, overhearing,* and *idle listening.*

Collisions

Collision loss refers to energy wasted due to packet collisions on the wireless medium. If a transmission of sufficient signal strength interferes with a data packet being sent, the data will be corrupted at the receiving end. Corrupted data can sometimes be recovered using error correcting codes (ECC), however, ECCs add transmission overhead, which is contrary to the goal of reducing radio transmit time.

Control Packet Overhead

Depending on the MAC protocol used, control packets may have to be received by all nodes within radio range of the sender, resulting in power drain in a potentially large number of nodes. If nodes are forced to stay awake for spurious control packets, battery life can be greatly impacted. Examples of control traffic are the RTS and CTS frames used by the IEEE 802.11 protocols.

Overhearing

Overhearing loss refers to energy wasted by a node having its radio in receive mode while a packet is being transmitted to another node. Most WSN MAC protocols reduce overhearing by trying to ensure that a node is only awake when there is traffic destined for it. One way to prevent overhearing is to ignore packets destined for other nodes after hearing an RTS/CTS exchange. After overhearing the RTS and CTS, nodes sleep during the NAV period as described in Section 2.2.1.

Opportunities for NAV sleep are significantly reduced on new platforms because the time required to transition to sleep and back is longer than packet transmit times for even the longest packets. The Tmote Sky, for example, takes 6.81 ms to transition the radio from receive to sleep mode and back [28], while the time required to send a maximum-sized IEEE 802.15.4-compliant frame of 128 bytes is only 4.09 ms.

Idle Listening

A node's radio consumes the same amount of power simply monitoring the channel as it does when it is receiving data. If an attack can make a node listen even when there is no traffic destined for it, power is wasted.

2.3.3 WSN MAC Protocols

Section 4.3 analyzes the impact of denial-of-sleep attacks against S-MAC, T-MAC, B-MAC, and G-MAC. Selecting a particular protocol for a given sensor network deployment depends on several factors such as the sensor platform selected, the energy efficiency of the protocol, the expected

duration of the deployment, the expected amount of unicast and broadcast traffic, and the memory footprint required by the protocol.

S-MAC

The S-MAC protocol [29] uses a fixed duty cycle, with a default of 10%, during which traffic is exchanged between nodes. By placing radios in sleep mode the rest of the time, node lifetime is significantly increased. In S-MAC, sensor nodes organize themselves into virtual clusters using periodic broadcast synchronization (SYNC) messages. Upon deployment, a node listens for a SYNC message. If it does not hear one before timeout, it broadcasts a SYNC message announcing its sleep cycle. Nearby nodes overhear this message and synchronize their schedules to the sending node. SYNC messages are repeated periodically to correct time drift and keep virtual clusters' sleep cycles synchronized. If a node overhears two SYNC messages, it will adapt both duty cycles to maintain network connectivity. Figure 2.3 depicts the S-MAC frame structure. Radios in networks using this protocol will be asleep 90% of the time, thereby producing an almost 10-fold improvement in node life over MAC protocols that do not use energy-saving techniques.

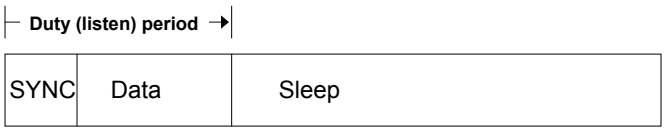


Figure 2.3: S-MAC frame structure.[2]

T-MAC

T-MAC [7] improves on S-MAC by concentrating all traffic at the beginning of the duty period as depicted in Figure 2.4, thus trading network latency for power conservation. The arrows in the figure indicate transmitted and received messages. T-MAC uses the same SYNC mechanism to form virtual clusters as S-MAC. Instead of remaining awake for a set period, however, an adaptive timeout (TA) mechanism allows nodes to transition to sleep mode when there is no more traffic

[2]Copyright IEEE. Used through residual rights retained from [30].

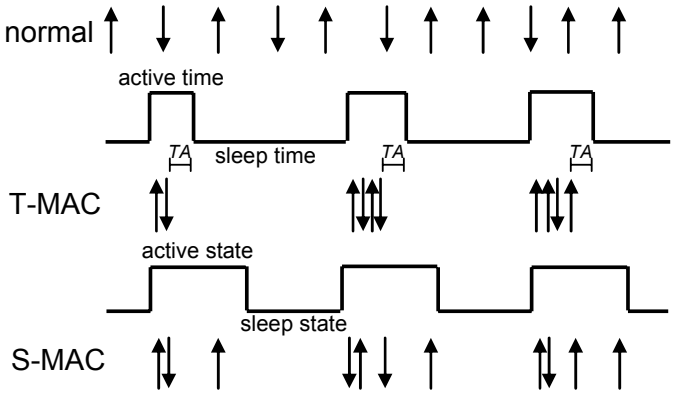


Figure 2.4: T-MAC adaptive timeout.[3]

in the cluster. When a node is awake, any time it senses activity on the wireless channel, it resets its time-to-sleep (TTS) to the TA value. If no traffic is observed before the TTS expires, the node transitions to sleep mode. According to van Dam and Langendoen [7], TA is set based on the longest time that a hidden node would have to wait before hearing the beginning of a CTS response message as follows:

$$TA = 1.5 \times (T_{CW} + T_{RTS} + T_{SIFS}), \tag{2.2}$$

where T_{CW} is the contention window duration, T_{RTS} is the time to send an RTS and T_{SIFS} is the short inter-frame space. The TinyOS implementation of T-MAC for the Mica2 platform uses a slightly different calculation of TA, which is discussed in Section 4.4. The improvement in network lifetime using this protocol depends on the amount of traffic in the network. Van Dam and Langendoen show T-MAC to have up to a five-fold increase in network lifetime over S-MAC [7].

B-MAC

B-MAC [8] does not form clusters of nodes or attempt to synchronize sleep schedules. Instead, it uses a technique called *low-power listening* (LPL) to reduce energy consumption. In low-power listening, nodes awaken briefly at a fixed interval and check the wireless channel for valid preamble bytes that indicate a pending data transmission from another node. A node with data to send transmits a preamble that is longer than the interval between receiver samplings to ensure that all

[3]Copyright IEEE. Used through residual rights retained from [30].

nearby nodes have the opportunity to detect the preamble and receive the subsequent data packet. The interval between channel sensings, or the *check interval*, is set based on average network node degree and traffic levels [8]. Figure 2.5 depicts sending node and receiving node behavior in B-MAC. Polastre, et al., showed that under ideal conditions, B-MAC could have duty cycles as low as 1% in a low-traffic network [8].

G-MAC

G-MAC [31] is an energy-efficient MAC protocol designed to coordinate transmissions within a cluster. Figure 2.6 depicts the G-MAC frame structure, which is divided into a collection period and a contention-free distribution period. During the collection period, nodes that have outgoing unicast or broadcast traffic transmit a future-RTS (FRTS) message to a gateway node. Traffic destined for other clusters is also transmitted to the gateway node during the contention period using a RTS/CTS/DATA/ACK exchange. At the end of the contention period the cluster head, or gateway, transmits a gateway traffic indication message (GTIM) which provides a mechanism for cluster synchronization while broadcasting a schedule of message transactions between nodes. Nodes then exchange data during the contention-free period. The gateway is elected using a periodic, resource-adaptive election process in which nodes volunteer based on current resource levels. New elections are indicated by a flag in the GTIM message. G-MAC eliminates overhearing, except for a minimum amount of control traffic that a node might overhear while waiting to transmit an FRTS during the contention period. Brownfield, et al., show G-MAC to improve network lifetime more than 3-fold over T-MAC in a simulated 50-node Tmote Sky network [31].

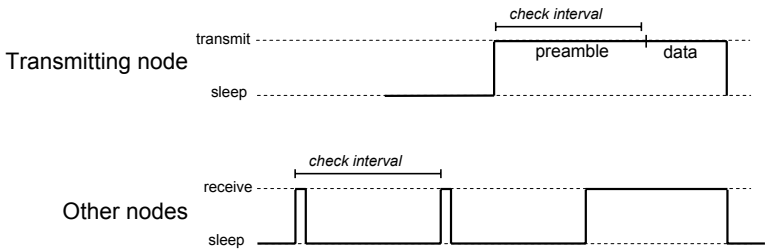


Figure 2.5: B-MAC sender and receiver behavior.[4]

[4]Copyright IEEE. Used through residual rights retained from [24].

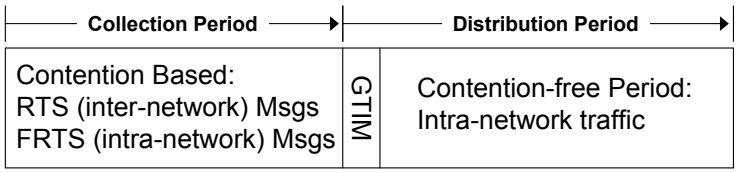


Figure 2.6: G-MAC frame structure.

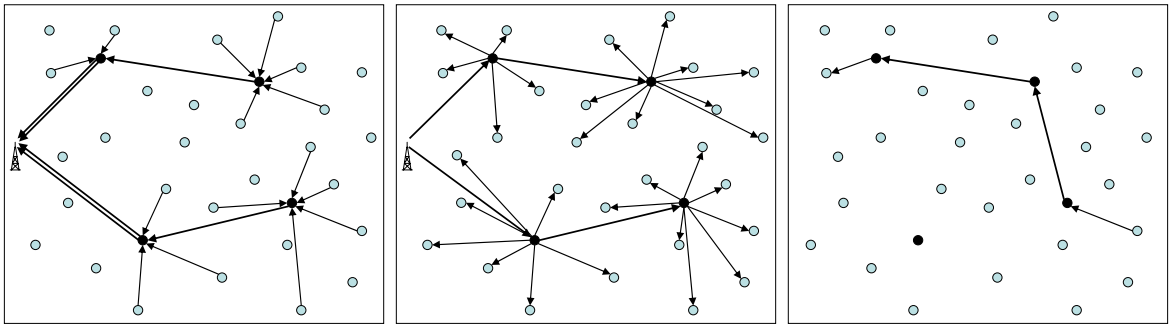
Discussion

The protocols described above represent the current state-of-the-art in energy-efficient WSN MAC protocols. As in other technologies such as ad hoc network routing protocols, a specific WSN MAC protocol has not emerged as a widely-held standard. S-MAC, T-MAC, and B-MAC are all included with the Concurrent Versions System (CVS) release of TinyOS, which is made available on the Sourceforge web site [32]. These protocols are, therefore, available for deployment in TinyOS-based sensor networks.

The selection of a MAC protocol for a particular sensor network deployment depends on several factors such as the energy efficiency of the protocol, the expected duration of the WSN deployment, the ability to replace or recharge power supplies, the memory footprint required by the protocol, the size of the network, the average node degree, and the expected network traffic levels. These variables, and others, present design-tradeoffs that affect the selection of protocols at the MAC, and other, layers.

2.4 Clustering in WSN

Some form of clustering is almost always required for scalablability in large-scale ad-hoc WSN deployments. Clustering reduces network contention by deconflicting inter-cluster interference through lower transmit power, separate channels, or spread-spectrum techniques, thereby improving spatial reuse. Reducing contention conserves energy and reduces latency in the network. Clustering can also conserve energy by aggregating and fusing data at clusterheads for transmission to a base station. Figure 2.7 depicts routine modes of cluster-based communication in WSNs.



(a) Data aggregation. (b) Full broadcast. (c) Node-to-node.

Figure 2.7: Routine communication models for cluster-based WSNs.

Although different in the details of their execution, clustering algorithms generally follow the sequence of events outlined here.

1. Some subset of network nodes volunteer (or are elected) to become clusterheads. This decision is often randomized, but is based upon current energy reserves, the number of times a node has served as clusterhead, some desired clusterhead node-degree, or other parameters.

2. Each remaining node in the network "joins" a cluster by communicating its intent to do so to one of the clusterheads. Deciding which cluster to join is often done by using received signal strength values to decide which clusterhead can be reached with the lowest transmit power.

3. Clusterheads establish routes to other clusterheads or to a base station according to the communication model used by the network.

Several sensor network clustering mechanisms exist in the literature. One of the first to show the potential energy-saving benefits of clustering in WSN is *Low-Energy Adaptive Clustering Hierarchy* or LEACH [33]. Support for clustering based on node-degree is included in the HEED [34] and ACE [35] protocols, among others. Secure cluster formation is addressed by Sun, et al. [36], and secure clusterhead election is explored by Crosby, et al. [37].

WSNs recluster periodically to distribute the clusterhead burden across nodes, thereby

equalizing energy consumption. Between these clustering events are steady-state periods, during which routine network traffic is exchanged. Clustering is infrequent, both to reduce the overhead of clustering events and to allow the network maximum time for routine operations.

2.5 Security in Sensor Networks

Sensor network deployments present unique security challenges. This section explores those challenges, and then describes current research into security mechanisms tailored to the limitations of these devices.

2.5.1 Sensor Network Vulnerabilities

Because WSN operate on the wireless medium, they are vulnerable to physical-layer attacks such as monitoring, packet injection, and jamming. The limited computational power and storage capacity of WSN platforms also reduces the feasibility of certain encryption techniques, an issue explored in the next section. Because sensor networks use some of the same routing protocols as other ad hoc networks, they are susceptible to routing attacks such as black hole attacks and HELLO floods [38]. Fortunately, most of these attacks can be prevented using existing authentication and encryption techniques.

Tradeoffs must be made to provide sufficient security while operating within the computational limitations of sensor platforms and minimizing the energy consumption overhead of the security techniques. Fortunately, the shorter packets and significantly lower rates of network traffic in most sensor networks provide much less data for intruders to analyze than in traditional computer networks, thereby reducing the vulnerability of encryption techniques to cryptographic attack. WSN energy reserves are sensitive to packet overhead, which increases transmission and reception times, thereby increasing energy consumption. Security mechanisms for WSN must, therefore, take care to minimize per packet overhead.

2.5.2 Packet Encryption and Authentication

General Techniques for Encryption and Authentication

Symmetric and asymmetric, or public key, cryptography are the two basic encryption techniques used for secure network communications [39]. In a symmetric cipher, both sender and receiver share an encryption key, which is used for both encrypting and decrypting messages. Common key lengths range from 32 to 256 bits, depending on the encryption algorithm used and the desired level of security. Symmetric encryption algorithms use multiple rounds of substitution and transposition to convert plaintext to ciphertext. Substitution involves mapping one element (byte or other bit grouping) with another element of the same size, and transposition simply rearranges elements. The computational overhead of these simple transformations is low. One shortcoming of symmetric encryption is the potential difficulty of securely distributing the encryption key throughout the network, which is necessary before secure communication can take place. Some well known examples of symmetric ciphers include the Data Encryption Standard (DES), and the Advanced Encryption Standard (AES) [39].

Asymmetric, or public key, cryptography eases the problem of key distribution by removing the requirement for security in the key distribution mechanism [39]. In a public key system, each node has a private key, which it alone maintains, and a public key, which is distributed in plaintext to all potential communication partners. The public and private keys are related in that text encrypted with the public key can only be decrypted using the private key, and vice versa. If a node wants to send secure data to another node, it simply encrypts the data using the receiver's public key, and only the receiver is able to decrypt it. Furthermore, public key encryption provides for message integrity. If a sending node encrypts a message with its own private key, another node that decrypts the packet with the sender's public key knows that the message must have come from that sender (assuming the private key has not been compromised).

Two characteristics of asymmetric encryption make it less attractive for WSN than symmetric ciphers. First, asymmetric encryption uses longer key sizes, up to 1024 bits, which consumes more RAM on sensor devices. More importantly, the encryption and decryption operations are

computationally expensive. The most popular algorithm, called RSA (named after the developers, Ron Rivest, Adi Shamir, and Len Adleman) encrypts and decrypts data by calculating extremely large exponents.

While the security of both symmetric and asymmetric encryption is comparable, symmetric encryption is generally preferred in WSN due to the significantly lower computational and key storage overheads. Secure key distribution is less of a concern in WSN than in other networks since these networks are usually deployed simultaneously by a single entity, who can hard-code encryption keys before network deployment.

Since symmetric encryption does not have a built-in mechanism for message integrity, networks must include an additional authentication mechanism. Authentication is generally provided using a computationally secure hash function to compute a message authentication code, which is appended to the outgoing message. The receiver simply recomputes the authentication code and compares it with the one received with the message to confirm message integrity.

In symmetric encryption, messages are broken down into blocks of data and transformations are executed over these blocks. A given plaintext block that is encrypted using a specific encryption key will always result in the same ciphertext, so repeated blocks of plaintext will result in repeated blocks of ciphertext. This fact might make it easier for an adversary to break the encryption mechanism, or to remove or substitute message blocks. To prevent these types of attacks, block cipher modes of operation are used to ensure that repeated plaintext blocks result different ciphertexts by using a portion of the previous encrypted block when encrypting a block of data. A unique initialization vector (IV) is used in the initial stages of this process to ensure that ciphertexts will be different, even for the same plaintext. This IV must be transmitted along with the message for it to be used in the decryption process. Transmitting the IV adds packet overhead to secure exchanges. Examples of block cipher modes of operation are cipher block chaining (CBC) mode and counter (CTR) mode [39]. Since sensor network energy reserves are sensitive to any unnecessary packet overhead, it often desirable in WSN to authenticate, but not encrypt, packets when message integrity is required but secrecy is not [40].

Encryption and Authentication in WSN

Perrig, et al., introduced two building blocks for security in WSN: μTesla, and *Secure Network Encryption Protocol* (SNEP). These two protocols comprise a security suite called *Security Protocols for Sensor Networks* or SPINS [41]. μTesla is a "micro" version of the *Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol*, or TESLA [42], modified for the resource-constrained environment of WSN. It uses delayed disclosure of symmetric keys to overcome the inability to use asymmetric security schemes for authentication due to their high processing overhead and long key lengths. SNEP provides data confidentiality and two-party authentication using symmetric encryption. SNEP also provides a mechanism to support data freshness using monotonically increasing counter values shared by sender and receiver, as described in Section 2.5.3. While this work was one of the first to describe mechanisms for security that are compatible with the resource constraints inherent to sensor node platforms, the SPINS suite was never fully specified nor implemented.

TinySec is a link-layer security architecture designed specifically for sensor networks [40]. It provides support for both authenticated encryption and authentication only. Authenticated encryption is achieved with a message authentication code computed over an encrypted message while pure authentication is provided using an authentication code computed over an unencrypted packet. TinySec uses *Skipjack* [43] as the default encryption mechanism and cipher-block chaining (CBC) [39] as the block cipher mode of operation. Energy consumption overhead in TinySec is relatively low and is caused primarily by longer transmission times due to increased packet length. Per-packet power consumption is increased by 10% for authenticated encryption and 3% for authentication only. TinySec is included with current releases of TinyOS.

For those devices using IEEE 802.15.1, or *Bluetooth*, radios, the Bluetooth specification includes support for encryption using an encryption system called E_0 [14], which is based on the SAFER+ block cipher [44]. E_0 uses a 128-bit stream cipher to provide encryption, integrity, and authentication.

For devices equipped with IEEE 802.15.4 compliant RF transceivers, the specification provides hardware support for data confidentiality and integrity [25]. It mandates the use of *Advanced*

Encryption Standard (AES) [39] encryption and a CBC message authentication code to provide support for access control, data encryption, and frame integrity.

2.5.3 Anti-replay Support

General Anti-Replay Techniques

Anti-replay protection allows a receiving node to identify replayed messages and discard them. This is often referred to as a guarantee of *weak* or *sequential freshness*. IPSec, for example, uses an incrementing counter included with each packet to ensure sequential freshness [45]. Each node maintains a table of each communication partner's latest packet counter value and compares each incoming packet with the stored counter value to make sure that the received packet has not been received already. IPSec's *sliding window* allows for out-of-order packet arrival. As long as packets arrive in order, or within the sliding window, they are accepted regardless of the time interval between packets.

A more stringent guarantee is *strong freshness*, which ensures that a message was sent within a certain, usually short, preceding time period. Strong freshness can be provided in one of two ways: either via a challenge/response sequence, usually involving a nonce value, or using a timestamp that is added to the packet during transmission and compared by the receiver with the global clock. All of these anti-replay techniques require that traffic be authenticated so that counter values and timestamps cannot be modified without detection.

Anti-Replay for WSN

Timestamps are not a practical alternative for anti-replay support in ad hoc sensor networks. Timestamps not only require that nodes maintain fine-grained synchronization, but also require that all nodes synchronize with a global clock. As mentioned in Section 2.3.1, synchronization mechanisms such as RBS require significant memory overhead and control traffic, severely taxing sensor nodes' limited resources.

One of the first anti-replay mechanism proposed for sensor networks was in the description

of the SPINS protocol by Perrig, et al. [41]. In their mechanism, weak freshness is a byproduct of CTR-mode encryption. A monotonically increasing counter is used as a nonce value for encrypting outgoing packets, and this counter is also used as an anti-replay counter. The counter need not be sent with the outgoing packet because the recipient maintains its own copy of the counter per peer node and uses this local copy in the decryption process. However, a dropped packet interrupts the sequence numbers and requires an expensive challenge-response handshake to resynchronize counters. In WSN scenarios, where moderate to high rates of packet loss are expected, the overhead imposed by this technique could be significant. Furthermore, the correct operation of the SPINS anti-replay mechanism requires that traffic be encrypted, so it is not compatible with the energy-saving authentication-only security model. SPINS also does not address the replay counter storage problem.

An improvement on the *implicit counter* technique used in SPINS is a mechanism for minimizing data transmission overhead in anti-replay protection described by Gouda, et al. [46]. The authors describe their *mixed sequencing protocol*, so called because it combines an *explicit counter* that is sent with transmitted packets, and an *implicit counter* that is maintained at both sender and receiver but is not transmitted. This protocol provides sequential freshness while allowing for lost and, if desired, reordered packets. An overview of the basic function of the mixed sequencing protocol is as follows. A sending node maintains an implicit sequence number, h, and an explicit sequence number, s. The value of s monotonically increases with each transmitted packet. The value of h is only incremented when s reaches its maximum value, at which time s wraps around to 0. The sending process computes a message digest, m, as follows:

$$m = MD_{sk}(s|h|x), \tag{2.3}$$

where $MD_{sk}()$ is the message digest function computed using an encryption key, sk, shared by sender and receiver and x is the packet to be transmitted. The transmitter then sends the message, x, along with the explicit counter, s, and the message digest, m. Upon receipt of the message $< x|s|m >$, the receiver compares the explicit counter, s, with its own version, s' and if $s \leq s'$, increments its own implicit counter, h', assuming that s has rolled over. The receiver then calculates

its own digest as follows:

$$m' = MD_{sk}(s|h'|x). \tag{2.4}$$

If m' equals m, the message is accepted. This construct allows for lost packets up to the maximum value of s (if s were to roll over twice, h' would only be incremented once, so m' would not equal m). Since packets must be authenticated for meaningful anti-replay support, the only additional overhead imposed by the mixed-sequencing protocol is the explicit counter value. While this technique does not reduce replay counter storage requirements, it does reduce per-packet anti-replay overhead.

For IEEE 802.15.4-compliant devices, the standard includes optional anti-replay support via frame counters included in the access control list (ACL) entries implemented in hardware [25]. The ACL contains cryptographic information, to include encryption keys, that a node will use to communicate with other devices in the network. Unfortunately, the IEEE 802.15.4 anti-replay mechanism is unsuitable for most ad hoc WSN applications. The primary weakness in this mechanism is the sharing of the encryption nonce and anti-replay counter. A counter, which is incremented with each sent packet, is used as the nonce value during the encryption process. This nonce is sent with the encrypted packet and used by the receiver(s) during the decryption process, and also as the anti-replay counter for the sending node. Each receiver must, therefore, have an ACL entry for each potential sender with the encryption key and the last received nonce/anti-replay value associated with that sender. The danger in this construct is the potential for a *same nonce* attack [47]. Since most networks must support broadcast traffic for clustering, control traffic, or other purposes, network-wide or cluster keys must be supported. In these cases, it is possible, and even likely, that a node will transmit multiple messages in the network using the same encryption key and the same nonce value. Multiple packets sent using the same key and nonce leaves the encryption mechanism open to cryptographic attack. For this reason, both Xiao, et al. [47], and Sastry, et al. [48], recommend that the anti-replay counter be decoupled from the nonce in future updates to the IEEE 802.15.4 specification. To avoid this attack, nodes must not keep the same encryption key in multiple ACL entries, which prevents the use of received nonces as anti-replay counters for broadcast traffic. Unfortunately, these replay counter values are not exposed to the

application layer, so they cannot be used by a higher-layer mechanism to maintain replay counters for individual senders.

The requirement for an ACL entry per sender to support sequential freshness points out another problem with the IEEE 802.15.4 anti-replay mechanism. This standard does not dictate a minimum number of ACL entries for a given implementation. The Motorola ChipCon CC2420 [13], which is currently the only IEEE 802.15.4-compliant radio used in available sensor platforms, has two ACL entries. Since IEEE 802.15.4 anti-replay requires that each node maintain an ACL entry for every peer node from which it might receive traffic, the two ACL entries included in the CC2420 implementation are insufficient for meaningful anti-replay support. This highlights a limitation with any hardware-based anti-replay mechanism: the size of a protected network is limited by the hardware implementation. Finally, IEEE 802.15.4's anti-replay mechanism is only compatible with encrypted traffic since nonces are not used for authentication.

2.5.4 Jamming Detection

Physical-layer jamming can simultaneously prevent traffic flow within a WSN and rapidly drain sensor batteries. Most wireless systems that must be protected from jamming use spread spectrum anti-jamming techniques such as frequency hopping spread spectrum (FHSS) or direct-sequence spread spectrum (DSSS) [49]. Sensor platform transceivers share the simplicity and low cost of other platform components, and most are not equipped to protect against jamming. For example, the Chipcon CC1000 transceiver [15], which is used on Mica2 and other sensor devices, is a relatively low data-rate radio with no spread spectrum capability. The IEEE 802.15.4, or *ZigBee*, specification defines physical and MAC layer functionality for a class of WSN transceivers [25]. WSN platforms that use a ZigBee-compliant transceiver, such as the Chipcon CC2420 [13], use DSSS to protect against noisy channels [25]. However, spreading codes are fixed according to the ZibBee standard, so anyone wishing to jam ZigBee devices need only use an IEEE 802.15.4-compliant transmitter to do so. Jam resistant spread spectrum communications rely on the sharing of a secret key between sender and receiver [50]. If communication is not secured using a cryptographically strong key, an attacker can predict the hopping sequence or DSSS bitstream and disrupt

communication. The overhead of producing and sharing an additional encryption key is another reason that these techniques are not well-suited for sensor networks.

Similarly, the Bluetooth physical layer uses frequency hopping to reduce the effects of noise and interference. The hopping sequence is a pseudorandom pattern that is determined algorithmically by the master node and shared with the slave nodes [14]. It is not determined in a cryptographically secure manner and, therefore, is not sufficient for jam prevention.

A potential defense against power consumption caused by jamming is to go to a low-power state when such attacks are in progress, waking only periodically to sense the channel [51]. A prerequisite for such a mechanism is for nodes to identify that a jamming attack is ongoing. Xu, et al., classify jamming attacks as *constant, deceptive, random,* or *reactive* [52]. Constant jamming normally involves a constant high-power transmission which consumes maximum energy by the attacker and may not be feasible if the attacker is under similar power constraints as the target network. A *deceptive jammer* sends a constant stream of seemingly legitimate data into the network to make it appear that the medium is being filled with legitimate traffic. A *random jammer* randomly alternates between sleep and jamming to save power, and a *reactive jammer* only sends a jam signal when it senses traffic to cause collisions.

The techniques for identifying jamming attacks explored by Xu, et al., include statistical analysis of received signal strength indicator (RSSI) values, average time required to sense an idle channel (*carrier sense time*), and packet delivery ratio (PDR) [52]. All of these techniques require that the network not be jammed upon deployment so that baseline measurements can be taken, and none of them alone identify all types of jamming. By combining techniques and introducing the notion of a consistency check, however, all four types of jamming can be reliably identified. One such algorithm first identifies poor link utility through PDR analysis and then uses a statistical RSSI analysis as a consistency check to determine whether the poor network performance is due to jamming. This method is explored further in Chapter 8.

Wood, et al., describe a jammed area mapping (JAM) protocol designed to allow sensor network traffic to be routed around jammed regions [53]. They mention factors that might be used to detect jamming, such as the inability to capture the channel, failed cyclic redundancy

checks (CRCs), illegal values in message fields, protocol violations, excessive RSSI, low signal to interference-plus-noise ratio (SINR), and repeated collisions. Their protocol uses a repeated inability to capture the channel to detect jamming. This technique is only reliable in the presence of constant jamming and will not detect random or reactive jamming.

In later research, Wood, et al., describe their DEEJAM protocol [54], which allows reasonable throughput for IEEE 802.15.4-based networks against a variety of energy-efficient jamming attacks perpetrated by a mote-class attacker. DEEJAM uses frame masking, channel hopping, packet fragmentation, and redundant encoding to achieve an average PDR of 88% in the face of the most effective jammer. All of the components of DEEJAM must be used simultaneously to resist all types of jamming described in [54], resulting in energy consumption overheads exceeding 150%. Such overheads are extreme and can reduce network lifetime to a fraction of what it would be without them. In the case that a jamming attack is ongoing, this overhead is justified, but in the more likely case that there is no jamming attack, it might be prohibitively expensive. A better alternative is to detect jamming, then impose the overheads of DEEJAM, or some other jam-defeating mechanism, until the attack is lifted.

2.5.5 Intrusion Detection

In this research, intrusion detection techniques will be used to identify the presence of network activity that might be part of a denial-of-sleep attack. This section introduces traditional intrusion detection, and then describes previous intrusion detection research in ad hoc and sensor networks.

Traditional Intrusion Detection

In traditional networks, the term *intruder* usually refers to someone trying to bypass encryption mechanisms or access-control protocols to gain access to information or resources to which they are normally not allowed access [39]. A more general definition of an intruder includes someone attempting to access a network for malicious purposes. In this definition, accessing the network does not necessarily refer to penetrating the network to gain unauthorized access to data, but also encompasses accessing network resources for malicious purposes, such as launching a denial-of-

service or denial-of-sleep attack. Intrusion detection systems (IDS) are agents designed to detect the presence of unauthorized users and, if possible, react appropriately to prevent the intruder from accessing or damaging the system [39, 55].

There are two basic approaches in traditional intrusion detection systems: *statistical anomaly detection* and *rules-based detection* [39]. Statistical anomaly detection requires the collection of "normal" network behavior of legitimate users over a period of time and uses statistical techniques to determine, with a high level of confidence, whether new behavior is legitimate or not. Rules-based detection compares user behavior against what is considered to be proper behavior and flags traffic that does not fit within expected parameters.

Rules-based intrusion detection can be further divided into two categories: *rules-based anomaly detection* and *rules-based penetration identification*. Rules-based anomaly detection is similar to statistical anomaly detection in that historical network traffic is analyzed to automatically generate rules that define proper user behavior. Rules-based penetration identification does not use previous network behavior to define rules or identify improper behavior. Instead, rules are specified by experts or are developed by expert systems based on known weaknesses or previous penetrations. Such systems do not require extensive records of previous network behavior, but they do necessitate an expert's knowledge of potential system vulnerabilities.

Intrusion Detection in WSN

There has been little research on IDS for sensor networks. The resource limitations of sensor networks described in Section 2.1.1 make it difficult to implement a traditional IDS in a sensor network scenario.

Da Silva, et al. [56], described their design for an IDS for WSN that uses monitor nodes distributed throughout a sensor network to perform traditional IDS functions. The monitors use rules-based anomaly detection to identify failures that might indicate network intrusions. The proposed intrusion detection algorithm executes in three phases:

Phase I: Data Acquisition. In this phase, monitors keep their radios in promiscuous mode to capture network traffic to which one of the rules might be applied. Traffic to which a rule might be

applied is placed in a buffer for later analysis. Other traffic is discarded.

Phase II: Rule Application. In this phase, the buffer, an array of data traffic captured during the acquisition phase, is analyzed to determine if any of the rules are violated. If a rule is violated, a counter associated with that rule is incremented.

Phase III: Intrusion Detection. In this phase, counters are compared with historical thresholds indicating failure levels for each individual node in the neighborhood. Only if counter values are greater than expected failure levels is an intrusion indicated. Expected levels of failure are determined during an initial learning phase, during which the monitors assume there is no malicious traffic in the network.

Da Silva, et al.'s mechanism runs concurrently with application software that carries out sensing and traffic forwarding duties on the monitor nodes, so these monitors are a functioning part of the sensor network. Their simulation results show that their mechanism has high probability of detecting jamming, replay, wormhole, and black-hole attacks, however, false-positive rates are often very high. Detection probabilities of other attacks such as delay, data alteration, and selective forwarding range from 20% to 90%, even with large buffers. This mechanism is not energy-efficient in that it keeps monitor nodes' radios in promiscuous mode, constantly sensing the wireless channel. Their results also do not include an analysis of memory overheads, which could be significant. Their best results were achieved when buffers were large enough to maintain 200 to 400 data packets. Two hundred standard-sized TinyOS data packets would consume almost 8 KB of memory (200×40 bytes), far outstripping the available memory on most WSN platforms.

Roman, Zhou, and Lopez [55] explored a general architecture for IDS in sensor networks, applying techniques proposed for ad hoc networks but acknowledging the differences between ad hoc and sensor networks. In their architecture, each node has an IDS agent, with functionality split between *local agents* and *global agents*. Local agents monitor traffic incoming to the local node. Because local agents are only concerned with communications involving the local node, they are only active when the local node is active and overheads are, therefore, low. Local agents might be programmed to identify such criteria as the node being physically moved, the reception of traffic from an unknown neighbor, or jamming. Global agents are responsible for monitoring commu-

nication among a neighborhood, perhaps behaving as watchdogs [57], promiscuously listening to traffic within the neighborhood and reporting potential intrusions. Global agents are employed on a cluster basis, either in conjunction with duties of the cluster-head in a clustered network, or after having been elected for the specific purpose of monitoring the network.

The authors introduced the idea of *spontaneous watchdogs*, in which the local and global agents reside on all nodes and each node chooses, with some probability, to act as a watchdog, listening to and analyzing traffic within their neighborhood. Increasing the probability of a node serving as a watchdog increases the likelihood that traffic within a neighborhood will be monitored, and also increases the overhead associated with overhearing and processing traffic. They claim that this mechanism will not incur any radio-related overhead because all nodes within a one-hop neighborhood must receive and process all packets sent by their neighbors. This, however, is not true in all WSN MAC protocols as some of them specifically avoid overhearing as discussed in Section 2.3.3. In S-MAC, for example, unicast traffic is often exchanged at the beginning of a frame's sleep period, during which all nodes except those involved in the unicast data transfer are sleeping.

Roman, et al., assumed that the primary responsibility of the IDS is to report intrusions to a user via a base station and not necessarily to trigger mechanisms to mitigate the potential effects of the intrusion.

2.5.6 Previous Research in Sensor Network Denial-of-sleep

Brownfield, et al. [58], introduced the *denial-of-sleep broadcast attack*, and modeled the effect of a malicious host obeying the MAC layer protocol and broadcasting unauthenticated traffic into the network. Even though the malicious broadcast traffic is dropped due to authentication failure, network lifetime is significantly reduced for networks using the S-MAC and T-MAC protocols. The authors introduced the G-MAC protocol, which weathers this type of malicious broadcast attack particularly well. In G-MAC, requests to broadcast traffic must be authenticated by the gateway node before the traffic can be sent to other nodes, so only the gateway suffers power loss due to unauthenticated broadcast. In Chapter 4, a more potent version of the denial-of-sleep broadcast

attack is modeled and its impact on WSN MAC protocols is examined.

The only other work known to address denial-of-sleep in sensor networks was preliminary research conducted as part of this dissertation research [30, 24].

2.6 Resource Consumption Attacks and Defense on Handheld Devices

Defenses against resource consumption attacks are becoming increasingly important in recent years as the number of energy-constrained networked devices increases rapidly. Most current research into resource consumption attacks has been limited to battery-powered, handheld devices. Resource consumption attacks were first introduced by Stejano and Anderson [59]. They called these attack *sleep deprivation torture* and investigated their impact on battery-powered mobile devices. This type of attack was further examined by Martin, Hsiao, Ha, and Krishnaswami [60], who classified potential attacks as *service request power attacks*, *benign power attacks*, and *malignant power attacks*. Service request power attacks are those in which an attacker makes repeated attempts to access a particular service on the mobile computer, causing the device to expend power reacting to the request even if the attacker cannot get access to the service. Benign power attacks are those in which the victim is duped into repeatedly executing an energy-hungry task such as displaying an animated GIF repeatedly showing the same frame, thus making it look like a non-animated GIF to the user. The malignant power attack alters the kernel or application programs such that energy consumption is increased, generally using a Trojan horse or virus.

Martin, et al., proposed a two-tiered power-secure architecture to defend against these attacks. First, multi-layer authentication would prevent abuse of untrusted services by only committing resources to requesters that have gained necessary levels of trust with the goal of preventing service request attacks. The second line of defense is an energy monitoring unit (EMU) that monitors the energy consumption of running applications and compares them to energy signatures of all applications.

This framework of attack classifications and power-secure architecture is well-suited for

battery-powered mobile systems that are designed to operate for a few to several hours between battery recharges. The attacks proposed by Martin, et al., target the application layer, which is reasonable for multi-purpose computing devices that run standard consumer operating systems such as Windows XP or Windows CE. The resource constraints of current sensor devices preclude the use of consumer operating systems and are less likely to be susceptible to the attacks in this study. As will be discussed in later chapters, packet authentication is a basic network security requirement and prevents many of the most devastating denial-of-sleep attacks.

Nash, et al. [61], describe the design of an IDS for detecting resource consumption attacks on mobile devices. The system alerts the user when device energy consumption exceeds a given threshold and identifies particularly power-hungry processes. The user can then kill any process that is deemed a threat. In their *Battery-sensing Intrusion Protection System* (B-SIPS), Buennemeyer, et al. [62], use system state information such as backlight settings and the active process list to establish a threshold of acceptable power consumption. Consumption above this threshold potentially indicates the presence of a network attack. A backend server attempts to correlate devices' network activity with known WiFi and Bluetooth attacks, and the user and the network administrator are alerted to the potential attack. Both of these systems are designed with the constraints of mobile devices in mind, but these devices far outstrip the resources available to WSN platforms.

An example of a resource consumption attack implementation was presented by Racic, Ma, and Chen [63]. Their attack takes advantage of two insecure cellular protocols: the Multimedia Messaging Service, an extension of the short messaging service (SMS) that is used to send text and multimedia files (JPG, GIF, MP3, MPEG, etc.) from one mobile device to another, and Packet Delivery Protocol (PDP) context retention, which is a mechanism by which state is maintained when a mobile user is in an active General Packet Radio Service (GPRS) session. GPRS is used by Global System for Mobile Communications (GSM) phones, currently the most popular mobile phone standard in the world [64].

Cell phones, like sensor platforms, use their transceivers sparingly. When a cellular phone is turned on, it generally uses its transceiver less than 3% of the time, keeping it in "standby"

mode the rest of the time [63]. The attack implemented by Racic, et al., keeps the device in receive mode permanently and drains the battery at up to 22 times the normal rate, depending on the platform. Because their attack takes advantage of general protocols that allow communication between mobile cellular devices and IP networks, it is platform independent.

2.7 Summary

In this chapter, necessary background on WSN and wireless networking was provided, with special emphasis given to MAC protocols and, in particular, those used in sensor networks. Current WSN MAC protocols use a variety of techniques to limit the amount of time that the transceiver is active, thus reducing energy consumption and increasing network lifetime. Later chapters show how security weaknesses in these protocols allow energy conservation mechanisms to be subverted to keep WSN transceivers on all, or most, of the time. These denial-of-sleep attacks drastically reducing network lifetime.

WSN clustering and the current state-of-the-art in WSN security were also introduced. Clustering is used to route traffic and conserve energy in sensor network deployments. Chapter 6 describes an anti-replay mechanism that leverages clustering to limit replay counter overhead. While the WSN security mechanisms described in this chapter are not sufficient to defend against attacks aimed at reducing network lifetime, they provide the foundation for the denial-of-sleep defense techniques introduced later.

Resource consumption attacks are not isolated to WSN deployments. A brief overview of related work in the area of resource-consumption attacks on battery-powered handheld devices was given to demonstrate active research in this area across multiple energy-constrained platforms.

The following chapter describes the methodology for determining the impact of denial-of-sleep attacks on WSN and the techniques used to design and evaluate prevention mechanisms.

Chapter 3

Research Methodology and Simulation Models

The art of war teaches us to rely not on the likelihood of the enemy's not coming,
but on our own readiness to receive him.
- Sun Tzu, *The Art of War*

The purpose of this research was to explore denial-of-sleep vulnerabilities in WSN MAC protocols, and then to design and evaluate mechanisms that identify and mitigate the effects of attacks that attempt to exploit those vulnerabilities. This chapter describes the methodology that was used to determine the impact of these attacks, as well as the methodology used to design and evaluate mitigation mechanisms. Section 3.1 discusses the system design goals and Section 3.2 gives a list of assumptions made in the design process. Section 3.3 describes the process by which denial-of-sleep mitigation mechanisms were developed. Section 3.4 discusses the design of the experiments that were conducted during each phase of this research and Section 3.5 describes the simulation models used during the design and analysis phases of this research. Finally, Section 3.6 provides a chapter summary.

3.1 System Design Goals

The goal of this research was to design and implement mechanisms to preserve sensor device energy in the face of denial-of-sleep attacks. If possible, it would be preferred for the network to maintain some level of functionality while the attacks are ongoing. The design criteria used during the development of denial-of-sleep mitigation mechanisms follow.

1. *Effectiveness.* To be worthy of inclusion in future deployed sensor networks, any resource consumption mitigation mechanisms must provide significant benefit. In other words, these mechanisms must extend the life of sensor devices under attack such that the network lifetime is close to what it would be if the network were not attacked.

2. *Low overhead.* The mechanism must be lightweight so as not to consume memory or processor time that should be devoted to the application for which the network is deployed. It should also have minimum network overhead so as to minimize the potential reduction in network throughput during times when the network is not under attack.

3. *Minimum modification to existing protocols.* To the extent possible, the mechanism should be easily incorporated into networks using existing MAC protocols.

4. *Platform independence.* The mechanism should be as general as possible so it can be used on various sensor platforms and should integrate with currently existing WSN protocols.

5. *MAC protocol independence.* As discussed in Chapter 2, there is currently a range of WSN MAC protocols, none of which has emerged as a standard that is preferred under all network conditions. Therefore, it was not reasonable to simply select a protocol and add features that prevent or mitigate denial-of-sleep attacks. Instead, a suite of tools, selectable at compile time, were developed to address various MAC protocol vulnerabilities.The concepts used to develop these tools can also be applied to anti-denial-of-sleep mechanisms built into future MAC protocols and some of these tools might be incorporated as-is into future protocols at compile time.

6. *Automatic reaction to suspected network attack.* Because sensor networks are not constantly monitored, denial-of-sleep defense techniques must sense malicious activity and react without

human intervention to protect the network from denial-of-sleep attacks.

7. *User configurability.* Depending on the sensor platform, the MAC protocols used, and the level of protection desired, the user should be able to configure the denial-of-sleep mitigation mechanisms as desired. This is in keeping with the paradigm of TinyOS programming in which only required features of the operating system are loaded into a device to conserve resources [65].

Table 3.1 provides metrics for the above design goals against which denial-of-sleep mitigation mechanisms were measured. The quantitative metrics used in the table, which include network lifetime, network throughput, and memory overhead, are defined in Section 3.3. The degree to which each of these goals were met in the various denial-of-sleep mitigation mechanisms is discussed in Chapters 6 through 8 and summarized in Chapter 9. One of the costs associated with some denial-of-sleep mitigation mechanisms is an increase in average packet latency when a network attack triggers one of the denial-of-sleep mitigation mechanisms. This cost is examined where appropriate. The average packet latency metric is also defined in Section 3.3.

3.2 Assumptions

Assumptions were necessary to define both the boundaries of this research and the scope of the problem that was to be addressed. The following assumptions apply to the design of WSN denial-of-sleep mitigation mechanisms.

1. To be deployed unobtrusively and in large numbers, wireless sensor platforms must be small and inexpensive. Therefore, constraints on sensor devices will not change significantly as technology improves. Rather, the devices will get smaller and less expensive. The assumption is, therefore, that future sensor platforms will operate under constraints similar to those of current platforms.

2. Researchers have shown that it is feasible to provide data confidentially and authentication in the resource-constrained environment of wireless sensor platforms. Security mechanisms discussed in Section 2.5, including TinySec[40] and SPINS [41], use lightweight, symmetric

Table 3.1: System Design Goals and Metrics

Design Criteria	Goal
Effectiveness	Maintain network lifetime of 90% or better while under attack as compared to a network not under attack.
Low overhead	1. Use an average of less than 5% total memory per node. 2. Reduce network lifetime by 5% or less while **not** under attack. 3. Cause 2% or less reduction in network throughput while **not** under attack. 4. Minimize processor overhead through efficient coding and by avoiding floating point calculations.
Minimum modification to existing protocols	1. Protocols function normally when **not** under attack. 2. Allow application compilation with or without added denial-of-sleep mitigation mechanisms.
Platform independence	Show effectiveness on multiple platforms through simulation and implementation.
MAC protocol independence	Demonstrate effectiveness on S-MAC, T-MAC, B-MAC, and G-MAC protocols.
Automatic reaction to attack	React to attacks based on all of the vulnerabilities identified in Chapter 4.
User configurability	Allow user to easily modify which denial-of-sleep mitigation components are used and to modify component parameters in configuration file.

encryption schemes to provide encryption necessary to secure sensor data. Encryption and authentication are, in fact, readily available for sensor platforms using the TinySec suite, which is included with current releases of the TinyOS [16] operating system, the most widely supported operating system for the wireless sensor platforms considered here [66]. Some currently available sensor devices employ radios that are compliant with the IEEE 802.15.1 and 802.15.4 specifications [14, 25], which require encryption and authentication support at the link layer. It is, therefore, assumed that WSN traffic can be protected from tampering and spoofing via authentication.

3. Under normal circumstances, a sensor network will experience little MAC-level replayed data

and unauthenticated traffic. If sensor devices are working properly, most MAC protocols only replay packets if they are not properly acknowledged by the receiver. Even these replayed packets would carry a properly incremented MAC-level replay-counter or time-stamp so this type of replay would not be flagged as malicious traffic by an anti-replay mechanism. Unauthenticated traffic would normally only occur when a transmitted packet is not received properly due to interference on the wireless medium. It is assumed, therefore, that anything other than low rates of replayed or unauthenticated packets indicate either a malfunctioning node or a malicious attack, either of which can lead to improper network operation and can have a negative impact on sensor lifetime.

4. It is likely that some (or even most) of the proposed sensor network deployment scenarios will not involve constant user interaction with the network. Consider a network deployed for environmental monitoring in a remote location. While sensor data might be received and stored constantly, the network may be monitored at a control center that is only staffed during routine business hours. Depending on the location, a network might only be visited periodically, every few days or weeks, to download collected data. It is, therefore, assumed that not all networks will be monitored full-time and that the network must be able to take immediate action without human intervention when malicious activity is identified.

5. This research assumes a network of homogeneous, stationary nodes, each with a fixed power supply. The specific platforms considered are the Crossbow Mica2 [21] and the Tmote Sky [22], however, the approach taken in this research will apply across the various sensor platforms introduced in Section 2.1.1. The Mica2 and the Tmote Sky are representative of the range of available sensor platforms and they both support the TinyOS operating system, the most widely used sensor network OS.

3.3 System Design Methodology

The mechanisms for mitigating the effects of denial-of-sleep attacks were developed using an iterative application of the ten-step systematic approach to performance evaluation described by

Jain [5]. The following paragraphs list these steps and describe how they were applied to this research.

Step 1: State Goals and Define the System. The first step in a well conceived performance analysis study is to clearly state the goals of the analysis and define the system under consideration. The goals of this study were to analyze the effectiveness and overhead of mechanisms designed to mitigate MAC-layer denial-of-sleep attacks. The system is defined as a network of sensor devices such as those described in Section 2.1. The specific platforms used in this research are the Tmote Sky and Crossbow Mica2, which are representative of the range of currently available sensor network platforms. To keep the problem tractable, the design and analysis assumed a one-hop neighborhood of homogeneous sensor devices with limited power supplies.

Step 2: List Services and Outcomes. Here, the pertinent services provided by the system under consideration are given. Sensor network traffic is generally characterized by broadcast traffic originating from a base station and eventually being broadcast within each cluster of nodes, and unicast traffic originating from sensor devices which is aggregated at a clusterhead or simply routed back to a base station. The system offers two basic services: broadcast traffic and unicast traffic.

Step 3: Select Metrics. Performance metrics can be categorized into three utility classes: *Lower is Better* (**LB**), *Higher is Better* (**HB**), and *Nominal is Best* (**NB**) [5]. The following performance metrics, with corresponding utility classifications, were used.

(a) Network lifetime (**HB**). This metric is defined as the average time elapsed between network deployment and the time that nodes' power supplies are exhausted, assuming all nodes are deployed at the same time. It was used to evaluate the effects of denial-of-sleep attacks on WSN MAC protocols, the effectiveness of mitigation mechanisms, and power consumption overheads imposed by these mechanisms. The nature of the workloads and protocols used in this research tend to result in an even distribution of energy consumption across network nodes.

(b) Average network throughput (**HB**). This is defined as the average amount of data successfully received by nodes in the network during the period being measured. It is measured in bits per second (bps). It was used to determine the extent to which denial-of-sleep attacks and mitigation mechanisms reduce the amount of data successfully transferred in the network.

(c) Average data packet latency (**LB**). This metric is defined as the average time elapsed between the time that a packet arrives for sending at the MAC layer of the originating node to the time that it is fully received at the MAC layer of the destination node. It was used to determine the extent to which packet latency is increased by denial-of-sleep mitigation mechanisms.

(d) Memory overhead (**LB**). This metric is defined as the number of bytes required both for additional program storage and for storage of data used by denial-of-sleep mitigation methods. WSN platforms are generally extremely resource-constrained. Any memory overhead reduces the amount of memory available to sensor applications.

As described in Section 3.1, mechanisms designed to mitigate the effects of denial-of-sleep attack should maintain the lifetime of an attacked network similar to that of a network not under attack while not decreasing network lifetime for a network that is *not* under attack. Furthermore, these mechanisms should, to the extent possible, maintain network throughput and have minimum impact on data packet latency and memory requirements.

Step 4: List Parameters. What follows is a parameter list, divided into system parameters and workload parameters. For this study, system parameters, which are those parameters involving hardware and software configuration, include the following.

(a) Number of nodes in the network

(b) Average node degree, or average one-hop cluster size

(c) Platform radio data rate

(d) Physical layer packet overheads

(e) MAC protocol overheads

Workload parameters are affected by the load placed on the system by applications and users. The workload parameters affecting this system are as follows.

(a) Rate of unicast and broadcast traffic

(b) Presence or absence of various denial-of-sleep attacks

Step 5: Select Factors to Study. Here, the parameters that were varied during the performance study, or *factors*, are listed, along with the *levels* of those factors that were examined. These must be selected carefully—there are not enough resources to conduct an exhaustive comparison of all possible factors and levels. The factors and levels that were studied included the following.

(a) Cluster size. This factor was varied from 5 nodes to 50 nodes. This range of cluster sizes is consistent with cluster sizes in current sensor network research [67, 68, 69]. The corresponding levels varied in different stages of this research. Levels for each experimental phase are given in Section 3.4.

(b) Platform radio data rate. Two popular sensor platform radios, with corresponding data rates, were examined. The ChipCon CC1000 has a data rate of 78 kbps and the ChipCon CC2420 has a data rate of 250 kbps.

(c) MAC protocol. Four of the MAC protocols introduced in Section 2.3.3 were examined at various stages of this work: S-MAC, T-MAC, B-MAC, and G-MAC. Overheads were those associated with each protocol.

(d) Denial-of-sleep attacks. As discussed in Chapter 4, certain characteristics of the various MAC protocols leave them susceptible to different attacks. Table 3.2 shows the attacks that were tested against each MAC protocol. These attacks were selected because their implementations do not require an attacker to penetrate existing link-layer encryption mechanisms. The constant deceptive jamming attack, which will be executed against all of the MAC protocols, is the easiest for an attacker to implement. However, this attack is inefficient and obtrusive in that it

requires the attacker to transmit a constant jamming signal, thus consuming maximum power and preventing legitimate traffic on the network. The other, protocol-specific, attacks are all more efficient because none of them require the attacker to constantly transmit a signal. These attacks are also more subtle in that they can be executed such that collision-avoidance rules are followed, which means that legitimate traffic can continue in the network. An analysis of the efficiency of these attacks is in Chapter 4.

Table 3.2: Denial-of-sleep Attacks Tested By Protocol

S-MAC	T-MAC	B-MAC	G-MAC
Constant deceptive jamming	Constant deceptive jamming	Constant deceptive jamming	Constant deceptive jamming
SYNC (replay) attack	Subtle broadcast attack	Subtle broadcast attack	FRTS replay attack

(e) Components included. The different MAC protocol vulnerabilities explored in Chapter 4 necessitate a variety of mitigation techniques and not all MAC protocols are vulnerable to the same attack types. Because sensor platforms are so resource-constrained, including unnecessary components is discouraged. The set of components included is, therefore, protocol-dependent. The denial-of-sleep prevention and mitigation mechanisms recommended for each MAC protocol studied here are given in Chapter 5.

Step 6: Select Evaluation Techniques. The three available evaluation techniques are analytical modeling, simulation, and measuring a real system [5]. The initial phase of this research used analytical models and real-system measurements to determine the effects of denial-of-sleep attacks on WSN MAC protocols. During the design phase, analytical models were used to estimate the overheads associated with various design trade-offs encountered in the development of denial-of-sleep mitigation mechanisms. Simulation was also used during the design and testing of mitigation mechanisms. Once the design was complete, mechanisms were simulated using OPNET Modeler [70] to measure effectiveness and overheads across the range of factors given above. To

validate the effectiveness of the proposed techniques, a prototype implementation was developed and measurements were taken on sensor devices that are part of a real sensor network.

Step 7: Select Workloads. The workload is the load placed on the system by users or malicious network intruders.

Legitimate network traffic workload. Packet rates vary widely in sensor network applications. Park, et al., describe a sensor network application to detect and track gaseous plumes [68]. Their communication model has nodes in a 25-sensor network accumulating sensor data and sending a packet every 2 seconds. With an average cluster size of 7 nodes, this model results in 3.5 packets per second (pps) in each cluster. A 49-node sensor network deployed in 2003 by researchers at the University of California, Berkeley, for environmental monitoring had nodes sampling and sending data via a multi-hop sensor network at rates of 5 to 20 minutes per sample, per node [67]. This packet rate produced an equivalent of 6 to 24 seconds per packet (or 0.17 to 0.04 pps). This research examines cluster-wide offered loads ranging from one packet every 4 seconds (0.25 pps) to 4 pps.

Data packet sizes depend on the MAC protocol and radio platform. For the CC1000 radio platform, the maximum size for S-MAC and T-MAC packets is 250 bytes with physical-layer and MAC-layer headers included. For the CC2420 radio, which conforms to the IEEE 802.15.4 standard, the maximum packet size is 128 bytes. Although S-MAC, T-MAC, and G-MAC have not yet been implemented for the CC2420, it is assumed that these protocols will adhere to this standard and limit packet sizes to 128 bytes. B-MAC packets are limited to 36 bytes on both platforms.

Specific legitimate traffic workloads for analytical models, simulation, and real-time system measurement are detailed in Section 3.4.

Malicious traffic workload. In this work, the other workload component is malicious network traffic in the form of denial-of-sleep attacks. Malicious traffic included the various network attacks listed in Table 3.2.

Step 8: Design Experiments. Experimental design is a critical step in performance analysis as it allows the analyst to gain the maximum amount of useful data using minimum resources. Descriptions of the design of analytical modeling, simulation, and real-system measurement experiments follow in Section 3.4.

Step 9: Analyze and Interpret Data. The techniques that were used to analyze results for the various evaluation techniques are summarized in Section 3.4.

Step 10: Present Results. Analytical results demonstrating the effects of denial-of-sleep attacks on unprotected systems are given in Chapter 4. Results of simulation and implementation experiments for the denial-of-sleep mitigation mechanisms developed during this research are presented in Chapters 6 through 8.

3.4 Experimental Design

This research was executed in three distinct phases. Phase I involved researching WSN MAC protocols and identifying vulnerabilities that expose them to potential denial-of-sleep attacks and, then, analyzing the effects of these attacks. In Phase II, mechanisms were designed to prevent or mitigate denial-of-sleep attacks targeting the vulnerabilities identified in Phase I. Finally, in Phase III, the mechanisms developed in Phase II were tested for effectiveness and to validate overheads estimated during Phase II. The goals and evaluation techniques for each phase are summarized in Table 3.3. This section describes how these evaluation techniques were used during the three phases of this research.

3.4.1 Analytical Modeling

During Phase I of this work, analytical models were used to calculate network lifetime both under routine network traffic and when the networks were under attack. The analytical models were based on those developed by Brownfield, et al. [58, 69]. These models assume a 50-node one-hop

Table 3.3: Research Phases and Evaluation Techniques

Phase	Goals	Evaluation Techniques
I	Analyze the effects of denial-of-sleep attacks.	1. Analytical modeling 2. Implementation and measurement
II	Design denial-of-sleep mitigation mechanisms.	1. Analytical modeling 2. Simulation
III	Analyze effectiveness and overheads of denial-of-sleep mitigation mechanisms.	1. Analytical modeling 2. Simulation 3. Implementation and measurement

neighborhood of nodes with no denial-of-sleep mitigation mechanisms in place. The workload for this analytical model was different for the two WSN platforms tested. The CC1000 (Mica2) workload was one 39-byte unicast or broadcast packet per second, in addition to control traffic used by the protocol. For the CC2420 (Tmote Sky) platform, with its significantly higher data rate, the workload consisted of four 64-byte unicast or broadcast packets per second. The factors that were varied in the analysis of denial-of-sleep attacks were WSN platform, the MAC protocol used in the network, and the specific denial-of-sleep attacks being executed. Details of the analytical model are given in Chapter 4, along with an analysis of the results of these experiments.

During Phase II, analytical models were used primarily to estimate potential energy savings and the energy consumption overhead of mechanisms designed to prevent specific types of denial-of-sleep attacks. Performance metrics of interest included power consumption overhead, which impacts network lifetime, and memory required for storage of data directly related to mitigation techniques.

Once the design phase was complete, Phase III began with initial overheads being calculated using the factors and levels given in Table 3.4. A *full factorial design (without replication)* [5], in which all combinations of factors and levels were tested, was conducted to determine overheads across the spectrum of possible configurations. These results are presented in Chapters 6 through 8 where appropriate.

Table 3.4: Preliminary Factors and Levels for Overhead Analytical Study

Variable	Factor	Levels
A	Cluster size	5, 10, 15, 20, 25, 30, 35, 40, 45, 50
B	Radio platform	CC1000, CC2420
C	MAC protocol	S-MAC, T-MAC, G-MAC, B-MAC
D	DOS attack	Constant deceptive jamming, Protocol specific

3.4.2 Simulation

Simulation was used in Phases II and III to assist in the development of denial-of-sleep mitigation mechanisms, to verify the effectiveness of proposed anti-denial-of-sleep algorithms, and to validate the overhead results from the analytical models. Simulation was required in this research for multiple reasons. First, one of the protocols studied, G-MAC, has not been implemented in hardware and two of the protocols, S-MAC and T-MAC, have not been ported to newer sensor platforms using the IEEE 802.15.4-compliant CC2420 radio. Simulation was, therefore, required to measure effectiveness and overheads across the range of platforms and MAC protocols. The development of denial-of-sleep mitigation mechanisms was also accomplished much more easily in simulation than on actual sensor platforms. Developing and debugging software for embedded systems such as WSN platforms is difficult due to the inability to get detailed data from the platform at run time. Simulation platforms, on the other hand, are designed to provide an abundance of information so that detailed device interactions can be analyzed and optimized. Developing first in simulation allowed for rapid development and debugging of embedded code, which was later ported to the Mica2 sensor platform.

Sensor Network Simulation and Simulators

For this research, a simulator was needed that was flexible enough to support the various sensor platforms and MAC protocols considered. Of the 16 simulation platforms considered, three were not available for general use because they were still in development (SensorSimII [71], TOSSF [72]) or were defunct (SensorSim [73]). Of the remaining 13, five are general-purpose net-

work simulation environments (Ns-2 [74], OPNET [70], OMNeT++ [75], Sidh [76], J-Sim [77]), four are special-purpose sensor network simulators (SENSE [78], SENS [79], VisualSense [80], and SWANS [81]), and four are sensor platform emulators (TOSSIM [82], ATEMU [83], Avrora [84], Emstar [85]).

In general, the special-purpose WSN simulators were developed for one of three reasons: 1) to improve scalability so large sensor deployments could be simulated; 2) to allow simulated sensor input; or 3) to allow interaction between simulated nodes and actual sensor deployments. None of the special-purpose simulators provide detailed MAC protocol implementations of any MAC other than IEEE 802.11. This research did not involve long-term simulations of large networks, nor was it concerned with simulating sensor input or interacting with actual deployments.

All of the WSN emulators except TOSSIM support the Mica2 platform only and, therefore, could not be used to study overheads and effectiveness of security mechanisms on the Tmote Sky platform. TOSSIM requires modifications to the MAC protocol to make it compatible with the TOSSIM physical-layer model. A modified version of B-MAC is available for use with TOSSIM, but there are no existing TOSSIM compatible versions of the other MAC protocols. For these reasons, TOSSIM was not used for this research.

Most previous WSN MAC protocol testing has been done using general-purpose simulators, such as Ns-2, OMNet++, and OPNET. The OPNET simulation platform was selected for this work. OPNET is a powerful and flexible simulation environment that allowed the simulation of multiple sensor node platforms and various protocols. Furthermore, there was an OPNET implementation of the G-MAC protocol available that was developed by Brownfield, et al. [69]. Although there was a requirement to develop OPNET node models of the Mica2 and Tmote platforms and MAC protocol process models, the language compatibility between the nesC language used by TinyOS and OPNET's Proto-C language eased the protocol development process. The OPNET process models were based largely on the original nesC code, which helped to ensure that protocol behavior was the same between simulation and implementation. The sensor node and MAC protocol process models developed for this research are described in Section 3.5 and in Appendix B.

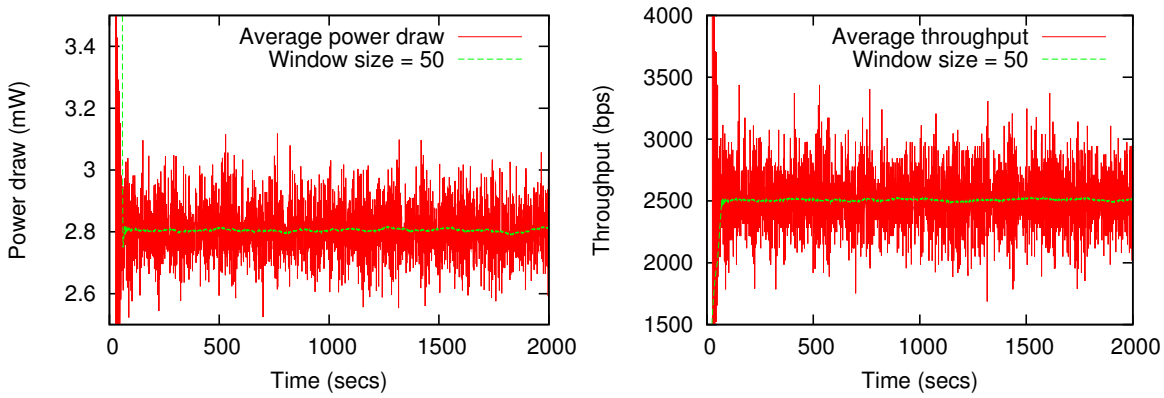
Determining Simulation Duration and Number of Replications

To confirm the effectiveness of denial-of-sleep mitigation mechanisms, all MAC protocols under consideration and the attacks given in Table 3.2 were simulated. The primary performance metric of interest in this simulation study was node lifetime, which provides a natural stopping point for runs and makes a terminating simulation a potential simulation technique [86]. However, sensor network lifetimes range from just under a week to several months. Simulating network behavior for the duration of network lifetime was, therefore, infeasible, so steady-state simulation was used. To determine average network node lifetime based on a steady-state simulation, monitors were included in the simulations to track energy consumption of each node. From this data, average energy consumption was calculated as follows:

$$E_{avg} = \frac{C_{battery(mWhr)} \times n}{\sum_{i=1}^{n} C_{i(mWhr)}}, \tag{3.1}$$

where n is the number of nodes in the network, $C_{battery(mWhr)}$ is the amount of available energy per node, and $C_{i(mWhr)}$ is node i's average energy consumption during the steady-state simulation. A similar technique was used to calculate throughput, which was a secondary performance metric.

The procedure developed by Welch [87] as described by Law and Kelton [86] was used to determine the *warm-up period*, otherwise known as the *initial transient*, or the amount of simulated time required for networks to reach steady-state for each simulation configuration. Welch's test requires the capture of average per-second performance data across several simulation replications using varying random seeds. A windowed moving average of the results is graphed to determine when the simulation reaches steady-state. For the Welch's analysis, twenty 10,000-second simulations were run using a set of default parameters. Since the primary performance metrics for this research were network lifetime and throughput, per-second energy consumption and throughput were captured during these simulations. The average per-second results were graphed, along with the windowed average results using window sizes ranging from 50 to 100. As an example, Figure 3.1 shows the Welch's test result for routine WSN traffic on a simulated Tmote Sky network



(a) Power Draw.

(b) Throughput.

Figure 3.1: Welch's test results for power draw and throughput in a simulated Tmote Sky network running the T-MAC protocol.

using the T-MAC protocol. For this example, the offered load is four packets per second and there is no network attack. It is clear from these figures that the network reaches steady state, both in terms of power draw and throughput, by 500 seconds. Statistics collection, therefore, begins at 500 seconds. The collection period selected was 10-times the initial transient, or 5000 seconds. These tests were repeated for each platform and protocol and for various network conditions to arrive at the warm-up periods and simulation durations shown in Table 3.5. Notice that all of the Mica2 simulations used the same warm-up periods and simulation duration, as do all of the Tmote simulations. This was done primarily to avoid confusion and to help ensure simulations were executed for the proper duration. The selected duration of the initial transient for each platform is a

Table 3.5: Simulation Druations Based on Welch's Test Results (Seconds)

	Mica2		Tmote Sky	
MAC Protocol	Begin collection	Simulation duration	Begin collection	Simulation duration
S-MAC	1500	16500	500	5500
T-MAC	1500	16500	500	5500
B-MAC	1500	16500	500	5500
G-MAC	NA	NA	500	5500

sufficient warm-up period for all MAC protocols.

Due to the stochastic nature of the discrete-event simulations, multiple replications of each simulation were conducted to ensure that results met desired confidence intervals. Network lifetime was the primary metric of interest in these simulations. To ensure statistically significant results, a 95% certainly that network lifetimes fall within ± 0.01 of the mean lifetime for each network configuration was desired. Based on this desired confidence, the number of replications, R, required for each configuration was calculated. First, an initial set of five simulation replications was run for each scenario to be tested. For the confidence interval to be within $\pm\epsilon$ of true value Ω with probability of $1 - \alpha_s$, we must have:

$$\epsilon \geq \left[\frac{t_{\frac{\alpha}{2},(R_0-1)} \times s}{\sqrt{R}} \right], \tag{3.2}$$

or

$$R \geq \left(\frac{t_{\frac{\alpha}{2},(R_0-1)} \times s}{\epsilon} \right)^2, \tag{3.3}$$

where ϵ is 0.01 times the mean value and s is the standard deviation of the five replications for that scenario. R_0, the initial number of replications, is five, and $t_{\frac{\alpha}{2},R_0-1}$ is the appropriate quantile value of the t distribution corresponding to $\alpha = 0.05$. In this case, the t value is 2.776. For this research, at least ten replications were used for each network configuration and this was a sufficient number of replications to meet the desired confidence intervals for most of the configurations. The rate-limiting simulations for both the Mica2 and the Tmote Sky platform discussed in Section 7.2.3 were the only cases in which ten replications was not sufficient to ensure a 95% confidence interval within 1% of the mean. For those simulations, sufficient simulations were conducted to provide results with the desired confidence interval, up to a maximum of 100 replications for any one scenario. Since all scenarios do not meet desired confidence levels, the sizes of the 95% confidence intervals, both for network lifetime and throughput for all scenarios and expressed as a percentage of the mean result for that scenario, are provided in Section 7.2.4.

Due to the resources required to conduct simulations and the requirement to execute multiple replications of each scenario to ensure statistical accuracy, not all scenarios were simulated.

Table 3.6 lists the factors and levels that were used in the simulation study.

Table 3.6: Factors and Levels for Effectiveness and Overhead Simulation Study

Variable	Factor	Levels
A	Cluster size	20
B	Radio platform	CC1000, CC2420
C	MAC protocol	S-MAC, T-MAC, G-MAC, B-MAC
D	Legitimate traffic rate	Platform dependent (0.25 pps to 4 pps)
E	DOS attack	Replay attack, Subtle broadcast attack

3.4.3 Implementation and Measurement

During Phase I, denial-of-sleep attacks were implemented to confirm the effectiveness of these attacks and to determine the efficiency with which they can be executed. Details of these implementations are given in Chapter 4.

To show that the denial-of-sleep defense techniques can be applied to actual WSN deployments, a prototype implementation was developed for the Mica2 sensor network platform during Phase III of this research. After implementation, the protocols were tested to verify effectiveness and to determine actual memory overheads by comparing program footprints both with and without denial-of-sleep mitigation mechanisms included.

To test the effectiveness of denial-of-sleep mitigation mechanisms, they were loaded and tested on a small one-hop network of Mica2 sensor devices. Implementations details for each of the mitigation schemes are included in Chapters 6 through 8. Attacks described in Chapter 4 were launched to determine the devices' abilities to detect and react appropriately. Sensor lifetime was determined primarily by the percentage of time that the radios were active. This information was maintained at the MAC protocol layer and periodically passed to a connected computer using serial output. Average sensor lifetime across a network of n nodes was then calculated as follows:

$$T_{sensorlife_{avg}} = \frac{\sum_{i=1}^{n} \left(\frac{C_{battery(mWh)}}{(S_i)\left(P_{sleep(mW)}\right) + (1 - S_i)\left(P_{active(mW)}\right)} \right)}{n}, \tag{3.4}$$

where S_i is the percentage of time that node i was in sleep mode, P_{active} and P_{sleep} are active mode power draw and sleep mode power draw, and $C_{battery}$ is the total amount of available energy per node. Table 3.7 lists the factors and levels that were examined during these real-system experiments.

Table 3.7: Factors and Levels for Effectiveness Real-System Measurement Study

Variable	Factor	Levels
A	Cluster size	5
B	MAC protocol	S-MAC, T-MAC, B-MAC
C	Legitimate traffic rate	0.25 pps, 0.33 pps, 0.5 pps, 1 pps
D	DOS attack	Jamming, Replay attack, Subtle broadcast attack
E	Anti-DOS components included	Protocol specific

In some situations, the Avrora emulator, mentioned in Section 3.4.2, was used during the testing of denial-of-sleep attack implementations and denial-of-sleep mitigation mechanism implementations. This was done when the results of experiments on actual Mica2 motes were inconclusive due to technical difficulties. The use of the Avrora emulator is discussed as appropriate in Chapters 4 and 7.

Statistical Accuracy of Real-system Experiments

Real-system measurements are much more time-consuming and work-intensive than simulation and analytical experiments. Furthermore, the purpose of the real-system experiments were primarily to demonstrate the proper functioning of the various denial-of-sleep mitigation techniques and to show that their energy consumption and throughput results are comparable with those observed in simulation. The simulation study provides a more accurate and precise estimate of the potential for these mechanisms to conserve WSN energy and throughput in the face of denial-of-sleep attack. As a result, real-system experiments were conducted for shorter durations and fewer replications were executed in most cases.

The statistical accuracy of the experiments are reported in the appropriate sections. This primarily applies to Section 7.3, where multiple replications of the combined CARP and CARL

mechanism are examined. The number of replications varies depending on the scenario, but range from ten to twenty replications. Sleep percentage and throughput data were captured for each node during each replication and averaged over the replication. Based on the sleep percentage data captured from nodes during these experiments, network lifetimes were calculated using (3.4), above. A 90% confidence interval is calculated over the replication data for each scenario using

$$\overline{X} \pm t_{[0.95,n-1]} \frac{s}{\sqrt{n}}, \tag{3.5}$$

where $\overline{X}$ is the mean, s is the standard deviation, n is the number of samples, and $t_{[0.095,n-1]}$ is the (0.95)-quartile of a t-variant with $n-1$ degrees of freedom. The results of this statistical analysis is provided where appropriate in the following chapters.

3.5 OPNET WSN Node and Process Models

To study the effects of denial-of-sleep attacks and defenses on WSN deployments, WSN node and MAC protocol models were developed in OPNET Modeler (version 11.5) [70]. This section provides an overview of these node and protocol models, using the T-MAC protocol model as an example. More details on each of the MAC protocol simulation models are included in Appendix B.

3.5.1 Top-down Simulation Design

OPNET simulations are developed using a top-down design process. Figure 3.2 shows a network of 20 WSN nodes in a 100 $\times$ 100 meter area. Each node represents an instance of the WSN platform node model shown in Figure 3.3, which is described in the next section. If this were a wired network, links between nodes would also be represented in Figure 3.2. Each block in the node model represents a finite state machine (FSM) that describes the functioning of that component. This top-down development process provides a structured and logical procedure for implementing a wide variety of network components and protocols.

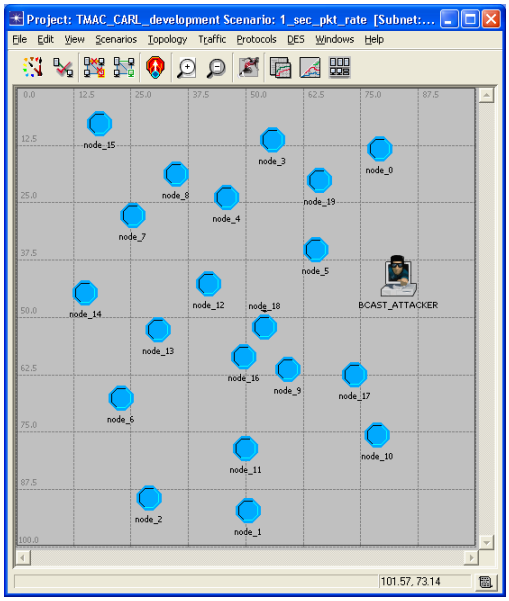


Figure 3.2: OPNET WSN simulation scenario.

3.5.2 Sensor Platform Node Models

Figure 3.3 shows the basic node model used in these simulations. At the bottom of the protocol stack, the physical layer models the CC1000 transceiver on the Mica2 WSN platform or the CC2420 transceiver on the Tmote Sky, depending on the device being simulated. The maximum data rate of the CC1000 transceiver is 76.8 kbps, however, the data rates used in the Mica2 implementations of the MAC protocols examined here is 38.4 kbps, using Manchester encoding. This results in an effective data rate of 19.2 kbps. In the Mica2 model, TinyOS Mica2 defaults for frequency (915 MHz) and transmit power (0 dBm) were used. The Tmote model has a data rate of 250,000 kbps and a default frequency of 2.4 GHz. The next layer up the protocol stack is the TinyOS active messaging (AM) link-layer, which uses a simple carrier sense, multiple access (CSMA) protocol to arbitrate channel access. This process model incorporates the delays caused by the radio switching between transmit, receive, and sleep modes. For accurate link-layer functionality, transitions between states are delayed by the times given in Table 2.1. The next higher layer is the MAC protocol. The development and functioning of the MAC process models are discussed in the next section and in Appendix B. At the top of the protocol stack, the applica-

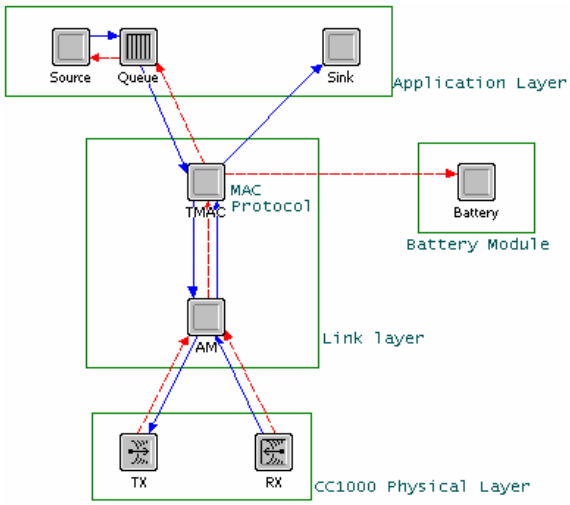


Figure 3.3: OPNET WSN platform node model.

tion layer model is comprised of a packet source, which generates random traffic according to the desired network offered load, and a sink, which records statistics for received packets that pass authentication and anti-replay checks at the MAC layer. The application layer also has a queue to accumulate packets waiting to be sent. Since the MAC implementations, including T-MAC, only accept a single packet at a time for transmission, the application layer is responsible for queuing packets. Because of the resource constraints of sensor platforms, a large packet queue is unrealistic, therefore, the size of the queue is limited to five packets for these experiments. Due to their resource constraints and the application-specific configuration and programming of these devices, the protocol stacks for WSN generally do not include a separate transport or network layer. The battery module is used to track platform energy consumption and distinguishes the Mica2 model from most existing OPNET node models. The battery module is described in Section 3.5.4, below, and in Appendix B.

3.5.3 MAC Protocol Process Models

For this research, process models for the MAC protocols S-MAC, T-MAC, and B-MAC were developed to reside between the application and AM layers in the node model shown in Figure 3.3. As an example, the T-MAC process model FSM diagram is shown in Figure 3.4. As with the

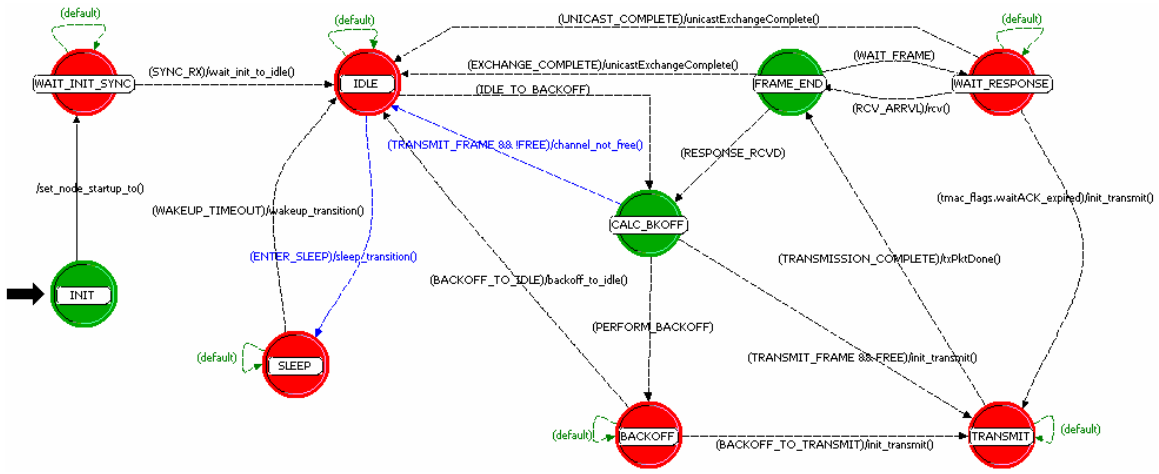


Figure 3.4: OPNET T-MAC protocol process model.

other MAC protocols, T-MAC was written for the TinyOS sensor network operating system in the programming language nesC [19]. To ensure proper simulated protocol functionality, the T-MAC nesC code was ported into OPNET's Proto-C where possible and inserted into the protocol process models' code blocks. Since nesC and Proto-C are both extended versions of the C programming language, the basic syntax is almost identical and porting the code into OPNET was straightforward.

TinyOS is an event-driven operating system designed specifically for the limited resources available to wireless sensor nodes [16]. Programs are built out of components, each of which provides an interface to its available tasks and to the events that must be handled by the interface's user. As an example, T-MAC provides such tasks as *unicastMsg* and *broadcastMsg*, which are called by the application layer to transmit unicast and broadcast packets, respectively. In turn, T-MAC calls the *txPkt* task from the physical layer interface to transmit packets. Most tasks are associated with an event, which is triggered when the task is completed. T-MAC provides a handler for *txPktDone*, which is triggered by the physical layer when the packet is successfully transmitted (or when the call to *txPkt* fails). Likewise, when T-MAC completes a unicast or broadcast transmission, it triggers a *sendDone* event, which must be handled at the application layer. This functionality was replicated in OPNET using interrupt-based state transitions.

The primary difference between the nesC protocol implementations and the Proto-C ver-

sions is that the nesC versions use a simple event loop based on a 1-ms timer to decrement countdown timers, check the status of various variables, and update node state. Using this model in OPNET would have resulted in extremely long simulations. Instead of using the 1-ms event loop, the OPNET MAC protocol implementations use interrupts to trigger state changes. Since state changes only occur when timers expire or when an event occurs, such as a packet arrival from the higher (application) or lower (physical) layers, it was possible to replicate MAC protocol functionality using OPNET Modeler's scheduled interrupts. Details on the functioning of the OPNET MAC protocol process models are given in Appendix B.

3.5.4 Energy Consumption Model

To compare network lifetimes among various network configurations and in the presence of denial-of-sleep attacks and defenses, an accurate energy consumption model had to be developed. Energy consumption is based on the amount of time a node spends in each of the three transceiver states: transmit, receive, and sleep. Radio state is controlled by the MAC protocol. Each time the radio transitions between states, a signal is sent on a statistics wire to the Battery Module shown in Figure 3.3, which sets the power consumption level for the new radio state. Transceiver state changes are not instantaneous. To accurately model energy consumption and protocol behavior, the energy cost and time required to switch states must be considered. Table 2.1 shows the time required to transition between modes and the energy consumed during each of these transitions, along with energy consumption levels for the three transceiver states. These values were derived for the Mica2 platform through experiments on Mica2 sensor platforms conducted by Polastre, et al. [8]. The Tmote values were determined through experiments conducted by Brownfield, et al. [69]. The battery model developed for this research also provides a local statistic that reflects the instantaneous energy consumption level. Energy consumption can, therefore, be selected in OPNET Modeler as an individual node discrete event simulation (DES) statistic. At the end of the simulation, the battery module uses accumulated time in each state, along with the number of transitions between transceiver states, to calculate the total and per-second energy consumption. This data is written to an output file for later analysis. The FSM for the energy consumption

process model is detailed in Appendix B.

3.5.5 Model Validation and Verification

The platform and MAC protocol models were developed to replicate the functionality of the protocols operating on actual Mica2 and Tmote Sky devices. The physical layer properties of the Mica2 and Tmote Sky were reproduced in their respective node models and the original nesC code was ported into the T-MAC and other protocol models. The OPNET Simulation Debugger (ODB) was used for initial model verification during protocol model development. To further verify the MAC protocol models, simulations were conducted to mirror the analytical models of energy consumption and throughput for an empty network, and for various rates of unicast and broadcast traffic. These simulations were conducted for networks ranging in size from 5 to 50 nodes using each of the protocols S-MAC, T-MAC, and B-MAC. The network attacks listed in Table 2 were also simulated to ensure agreement between analytical and simulation models. Average node energy consumption was used to calculate network lifetime for each configuration and those results were compared to network lifetimes determined in the analytical models. In all cases, simulated network lifetimes were within 2% of the analytical models and they were usually much closer. Figure 3.5 shows the difference between network lifetime determined using analytical models versus simulated network lifetime for a network of Mica2 sensors using the S-MAC protocol. The maximum difference between analytical results and simulation results is 1.3% (at 5 nodes).

3.6 Summary

This chapter described the goals of this research and assumptions that were made in the design of protocols to mitigate the effects of WSN MAC-layer denial-of-sleep attacks. It also described the process by which denial-of-sleep mitigation mechanisms were developed and provided the design of analytical, simulation, and real-system measurement experiments used to determine both the effectiveness of denial-of-sleep attacks and the effectiveness of the various mitigation mechanisms. Finally, it described the OPNET platform node models and WSN MAC protocol process

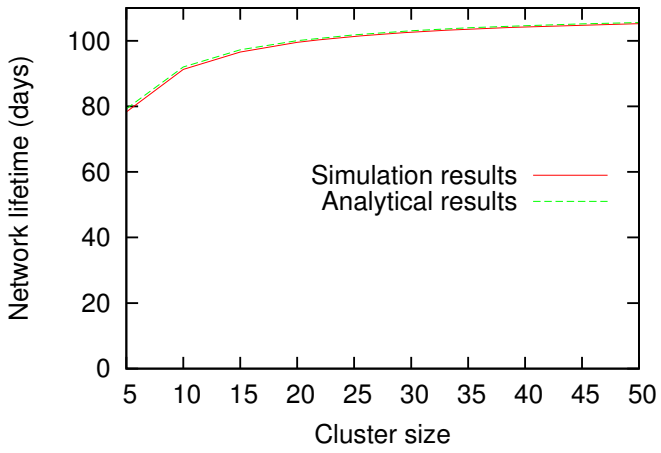


Figure 3.5: Analytical versus simulated network lifetime for a sample Mica2 S-MAC network.

models used in this research. More details on the various OPNET process models are provided in Appendix B.

The next chapter examines the denial-of-sleep problem in detail, discussing vulnerabilities identified in a representative set of WSN MAC protocols and the impact of a variety of denial-of-sleep attacks on these protocols. Subsequent chapters describe each of the denial-of-sleep defense techniques and provide experimental results.

Chapter 4

The Denial-of-Sleep Problem

If we continue to develop our technology without wisdom or prudence,
our servant may prove to be our executioner.
- GEN Omar Bradley

This chapter primarily focuses on results obtained during Phase I of this research in which the effects of denial-of-sleep attacks in sensor networks were analyzed. First, Section 4.1 points out specific vulnerabilities for each of the WSN MAC protocols studied herein to denial-of-sleep attack. Section 4.2 classifies potential denial-of-sleep attacks based on the level of protocol knowledge and the level of penetration of link-layer encryption mechanisms. In Section 4.3, several attacks are modeled on S-MAC, T-MAC, B-MAC, and G-MAC to demonstrate the potential severity of attacks from each attack class on these protocols. To validate these results, Section 4.4 describes the implementation and efficiency analysis of selected attacks on S-MAC, T-MAC, and B-MAC on the Mica2 platform. Section 4.5 examines traffic characteristics of each of the MAC protocols explored here and describes how an attacker might determine the MAC protocol being used in a sensor network simply by analyzing network traffic. Knowing the MAC protocol used in the network allows an attacker to launch a protocol specific attack to maximize effectiveness and efficiency. Finally, Section 4.6 provides a chapter summary.

4.1 MAC Protocol Denial-of-Sleep Vulnerabilities

Each of the protocols examined for this research display features that can make them vulnerable to a denial-of-sleep attack. In this section, these specific features and vulnerabilities for S-MAC, T-MAC, B-MAC, and G-MAC are discussed.

4.1.1 S-MAC Vulnerabilities

The original specification of the S-MAC protocol, detailed by Ye, Heideman, and Estrin in [6], has a fixed duty cycle that is set at compile time. With its fixed duty cycle, this protocol might seem to be the least susceptible to a MAC-layer denial of sleep attack. In fact, Law, et al., argue that a resource consumption attack at the link layer against S-MAC cannot be effective precisely because of its fixed duty cycle [88]. Although S-MAC is not vulnerable to a simple resource consumption attack using RTS packets to elicit CTS responses, the currently available S-MAC version, which was implemented by the authors of the S-MAC protocol and is available with the TinyOS CVS release on Sourceforge[1], is vulnerable to the attacks described in the following paragraphs.

Attacks on S-MAC SYNC Mechanism

As described in Section 2.3.3, S-MAC uses periodic SYNC packets to prevent clock drift from desynchronizing clustered nodes' schedules. Figure 2.3 depicts the S-MAC frame structure. Each frame is divided into the SYNC period, during which SYNC messages are sent, a data period, during which broadcast data traffic is transmitted and unicast exchanges are initiated, and a sleep period, when nodes that are not involved in a unicast exchange place their radios in sleep mode to conserve energy. Frame length is determined at compile time based on control packet size, transceiver bandwidth, and the duration of platform-specific interframe spacings (DIFS and SIFS). The frame length for the Mica2 using the default S-MAC configuration is 1,300 ms. The default SYNC period is 47 ms and the data period is 83 ms, for a 130 ms active period during each frame. The SYNC period at the beginning of each frame is time set aside for sending SYNC

[1]The S-MAC version analyzed here is from `tinyos-1.x/contrib/s-mac/`, in the TinyOS CVS repository [32] as of August 1, 2006.

packets, however, a receiving node processes SYNC packets whenever they are received. The default *SYNC_PERIOD*, or time between SYNC packets, is 12,000 ms, or 10 frames.

Each 10-byte SYNC packet includes a 2-byte *sleepTime*, which indicates to nodes receiving the packet when the sending node will next enter sleep mode. When a node receives a SYNC packet from another node on its same schedule, it recalculates its next sleep time to maintain synchronization. Instead of simply resetting its next sleep time according to the value in the SYNC packet, the receiving node splits the difference between its next sleep time and the time in the received SYNC packet as follows:

$$sleepTime = \left\lfloor \frac{sleepTime + receivedSYNCpkt.sleepTime}{2} \right\rfloor . \tag{4.1}$$

This allows nodes on the same schedule to improve synchronization over time, rather than having them radically change their schedules upon receipt of a SYNC message.

If network traffic is not secured with encryption or authentication, bogus SYNC packets can be crafted with arbitrary *sleepTime* values. This attack is explored further in Section 4.4.1. If traffic is secured, replayed SYNC packets can also be used to mount an effective denial-of-sleep attack. Even if they are encrypted, these packets are easily identified by an attacker monitoring a network cluster by their size and timing. S-MAC SYNC packets are 10 bytes long and occur during the first few milliseconds of an S-MAC frame. WSN encryption mechanisms are careful to minimize packet overhead, thereby limiting data transmission overhead, which consumes unnecessary power. Therefore, even if all packets are encrypted, various types of packets are still identifiable because their sizes increase by a constant amount (this is further explored in Section 4.5.1).

Replaying a constant stream of back-to-back recorded SYNC packets is sufficient to keep targeted nodes awake. To maximize attack efficiency, an attacker should send SYNC packets as far apart as possible to minimize attacker awake time while still keeping targeted nodes awake. Recall that each node receiving a SYNC packet calculates its new sleep time according to (4.1). The correct interval for a SYNC replay attack is, therefore, slightly less than one-half of the *sleepTime* specified in the SYNC packet. The receiving node's next sleep will be scheduled for approximately $\left\lfloor \frac{1.5 \times sleepTime}{2} \right\rfloor$, before which the node will receive another SYNC packet delaying its next sleep

opportunity by the same amount and keeping the node's radio awake permanently.

Attack on Adaptive Listening

Adaptive listening is an optional feature in S-MAC that was introduced to reduce message latency [29]. This feature leads to another denial-of-sleep vulnerability. S-MAC allows unicast message exchanges to extend into the period that nodes would normally be asleep. Adaptive listening has nodes that overhear an RTS/CTS exchange awaken at the end of the data transmission to listen for a potential RTS in case the receiver is required to forward the message in the network. This technique reduces latency because it allows data to be forwarded in the network immediately instead of the network having to wait until the next frame for each hop.

To exploit this mechanism, an attacker sends (or replays if traffic is authenticated) an RTS indicating a short unicast message. The Mica2 S-MAC implementation available for TinyOS has an effective data rate of 19.2 kbps. Nodes that overhear an RTS message do not enter NAV sleep until they receive the CTS reply, an exchange that takes 13 ms at 19.2 kbps. A minimum sized (12-byte) S-MAC data packet takes 5 ms to send and it takes a Mica2 2.8 ms to transition from receive mode to sleep mode and back [8]. Each victim device is, therefore, only able to sleep 2.2 ms out of every 18 ms (13 ms + 5 ms) period, or 12% of the time. For the 250-kbps CC2420 radio on the Tmote Sky, the maximum-sized 128-bit packet takes only 4.09 ms to transmit. It takes the Tmote 6.81 ms to transition to sleep mode and back [69], so nodes are never able to enter NAV sleep. Whether or not a network uses adaptive listening is set at compile time. This vulnerability is easily avoided by not enabling adaptive listening when preparing a sensor network for deployment.

4.1.2 T-MAC Vulnerabilities

Although T-MAC uses the same SYNC mechanism as S-MAC, the SYNC attack above is not effective on T-MAC because of T-MAC's adaptive timeout mechanism (which is described in Section 2.3.3). Under adaptive timeout, nodes transition to sleep mode after the network has been idle for a period of time defined as TA, even if the cluster-head's transmitted $sleepTime$ indicates that it will remain awake.

The adaptive timeout mechanism, although designed to improve network lifetime by increasing node sleep, leaves it vulnerable to a simple denial-of-sleep attack. By sending a constant stream of small packets at an interval just short of the network's adaptive timeout, sensor devices on a T-MAC network will remain constantly awake. The TA interval is platform-dependent and, as shown in (2.2), is based on the maximum contention window duration, the duration of an RTS message, and the duration of a short interframe space (SIFS). On the Mica2 platform implementation, which was written by the original designers of T-MAC, the authors use a slightly different formula for calculating TA than the one described in [7] (which is repeated in (2.2))[2]. Instead of using T_{SIFS}, they use two times the time that a node should wait for a CTS message after receiving a RTS message, which is calculated as the amount of time required to send a 12-byte CTS packet, plus a 10-ms processing delay. The value of TA, therefore, becomes:

$$TA = 1.5 \times (T_{CW} + (2 \times T_{Wait\_CTS})). \tag{4.2}$$

where $T_{Wait\_CTS} = T_{CTS} + 10ms$. By simply sending a few preamble bytes at a rate just short of TA, a T-MAC network can be kept awake indefinitely. An analysis of the efficiency of this attack is given in Section 4.4.2.

4.1.3 B-MAC Vulnerabilities

Deceptive Jamming Vulnerability

B-MAC uses low-power listening (LPL) to try to maximize sleep time as described in Section 2.3.3. To filter out a noisy channel, B-MAC uses a channel sampling algorithm to maintain a threshold at which the channel is considered to be clear [7]. This Clear Channel Assessment (CCA) keeps B-MAC from being susceptible to a denial-of-sleep attack based on simple jamming, since a physical-layer jamming signal would be observed by B-MAC as a noisy channel. Recall, however, that according to the low-power listening mechanism, if a receiver awakens to hear a valid series of

[2]The T-MAC version analyzed here is from `tinyos-1.x/contrib/t-mac/`, in the TinyOS CVS repository [32] as of August 1, 2006.

preamble bits, it will remain awake to wait for the arrival of the expected data packet. As first pointed out by Xu, et al. [52], a constant stream of preamble bytes transmitted by an attacker is sufficient to keep B-MAC nodes in constant receive mode. This attack is not efficient for the attacker, since the attacker must be awake and transmitting a constant signal. Although all nodes in a B-MAC network have the same check interval, the nodes are not synchronized, so an attacker cannot take advantage of victim node clustering.

Unauthenticated Broadcast Vulnerability

LPL is also vulnerable to unauthenticated broadcast and replay attacks. By sending back-to-back maximum-sized packets, each with full preamble, all nodes are required to awaken and receive each packet. Nodes must remain awake to overhear, on average, one-half of the preamble plus the duration of the data packet. The percentage of time that a Mica2 is awake per broadcast packet is determined as follows, where $T_{preamble}$ and T_{pkt} are the preamble duration and data packet duration, respectively:

$$\begin{aligned} awake\ percentage &= \left[\frac{\left(\frac{T_{preamble}}{2} + T_{pkt}\right)}{(T_{preamble} + T_{pkt})}\right] \\ &= \left[\frac{\left(\frac{113\ ms}{2} + 17\ ms\right)}{(113\ ms + 17\ ms)}\right] \\ &= 56.5\%. \end{aligned} \tag{4.3}$$

The preamble duration of 113 ms and packet duration of 17 ms are based on a 271-byte preamble and 36-byte data packet (with headers) transmitted at 19.2 kbps. If the packets are broadcast back-to-back, the overall awake percentage is the same as the per-packet awake percentage, allowing the victim to sleep only 43.5% of the time. Substituting the preamble duration and data packet duration for the Tmote Sky yields an awake percentage of 55.3%.

4.1.4 G-MAC Vulnerabilities

Of the WSN MAC protocols explored in this research, G-MAC is the most robust in terms of denial-of-sleep vulnerability. It is, however, susceptible to attacks involving its gateway traffic indication message (GTIM) mechanism and partially vulnerable to replay attacks.

Attack on GTIM mechanism

As described in Section 2.3.3, each G-MAC gateway uses a GTIM once during each frame to specify the schedule of broadcast and unicast messages during the upcoming frame. This message is easily identified in the network due to the precise timing of its transmission (as described in Section 4.5.4). According to the G-MAC protocol, each node awakens at a specified period during each frame to listen for the GTIM broadcast from the gateway node. G-MAC uses the point coordination function (PCF) defined in the IEEE 802.11 standard, which is similar to the distributed coordination function (DCF) described in Section 2.3, except that traffic is coordinated by a single host, in this case the gateway, instead of in a distributed fashion. The gateway node waits for the duration of a PCF interframe space, or PIFS, before transmitting the GTIM, as depicted in Figure 4.1, to ensure that it has control of the medium before transmitting [69]. This PIFS backoff provides a window of opportunity for an attacker to send a forged GTIM message into the network. By forging a GTIM message from the gateway node that specifies a schedule full of broadcast messages during the upcoming frame and transmitting just before the gateway's GTIM message, all nodes in the cluster except for the gateway will remain awake for the duration of the frame expecting broadcast messages. An attacker can repeat this forged GTIM during each frame to keep

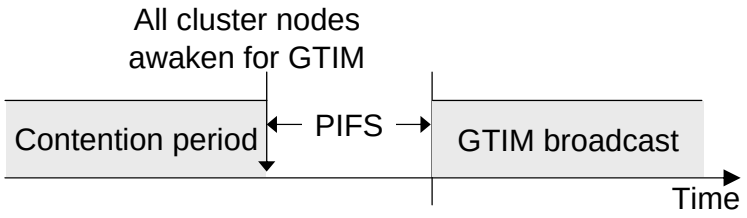


Figure 4.1: GTIM message timing.

cluster nodes awake indefinitely. Since the attacker must send only this short message during the GTIM portion of each frame, this is a very efficient attack.

If network traffic is authenticated and an attacker is unable to forge a GTIM message, it is reduced to trying to keep cluster nodes awake by recording and replaying a GTIM into the network before the PIFS timer expires. This attack is not likely to keep cluster nodes awake for a large percentage of the time, particularly in a network with low traffic rates. An attacker might maximize its effect on the network by monitoring the network and replaying the GTIM message that is associated with the highest number of broadcast packets. Broadcast packets can be identified based on the *packetType* field in the header or by the lack of a returned acknowledgment as described in Section 4.5.4.

Attacks on FRTS Mechanism

G-MAC nodes use future request-to-send (FRTS) messages during the contention period of the frame to reserve slots during the contention-free period. In an unauthenticated network, an attacker can forge broadcast FRTS messages and flood the gateway with these messages during the contention period, causing the gateway to schedule broadcast packets during the entire contention-free period. Assuming the contention period and contention-free periods are of equal length, this attack would keep cluster nodes awake during the GTIM broadcast period and the contention period, or for more than half of each frame. If packets are authenticated but not encrypted, this same attack is possible as soon as the attacker can record a broadcast FRTS packet for replay. If traffic is encrypted, the attacker can record random FRTS messages and flood the gateway with them. Although the attacker would not know which of these packets were for broadcast and which were for unicast messages, the attack still has an effect on network lifetime. Analysis of the effectiveness of each of these attacks under various conditions are in Section 4.3.2.

4.1.5 Vulnerability Analysis

So in war, the way is to avoid what is strong and strike at what is weak.

- Sun Tzu, *The Art of War*

The attacks explored above take advantage of vulnerabilities associated with techniques used to improve energy-efficiency. These vulnerabilities are shared with other protocols that use the same or similar energy-conserving mechanisms. Those vulnerability categories that lead to the denial-of-service attacks that are described in subsequent sections are the following.

Packet Spoofing Vulnerability The synchronization mechanism used by S-MAC uses control packets that specify sleep timings to form logical clusters of nodes. Similarly, the GTIM packet in G-MAC dictates to each node in the cluster when it must be awake and when it can sleep. Without link-layer encryption or authentication, an attacker can forge these packets and keep nodes awake. An attacker can also inject large numbers of legitimate-looking data packets into the network, consuming power on all nodes in receive range of the attacker. These mechanisms are vulnerable to efficient denial-of-sleep attacks in networks with no encryption or authentication. While encrypted and authenticated control packets are only as secure as the encryption algorithms used, taking these simple steps makes forging these packets extremely difficult and provides significant protection to these networks.

Replay Vulnerability The attacks on S-MAC and G-MAC using replayed control packets, such as SYNC, RTS, FRTS, and GTIM packets, highlight another vulnerability for both of these protocols. Anti-replay support to prevent these replayed control packets would prevent this category of attacks. Replayed data packets are also a threat because they waste node energy if they are forwarded through the network. By identifying these packets as replayed, they would be dropped by the receiver and not forwarded.

Jamming Vulnerability The deceptive jamming and intelligent jamming attacks on T-MAC exploit the requirement that network nodes must sense an idle wireless channel for a certain amount

of time before the nodes can assume that there is no more legitimate traffic in the network. While this feature is intended to allow nodes to go into sleep mode when there is no network activity, it is easy for an attacker to exploit. B-MAC's low power listening mechanism is also vulnerable to a deceptive jamming attack. While the attack on B-MAC cannot be mounted as efficiently as the one on T-MAC, it is still effective. These jamming attacks have the effect of not only keeping nodes awake when they should be sleeping, they also prevent communication between nodes.

Unauthenticated Broadcast Vulnerability Even if transmitted data cannot be authenticated, packets must be received and message authentication codes must be calculated before the bogus packets are dropped. This type of attack can rob bandwidth and prevent nodes from using NAV sleep, but has little resource consumption impact on fixed duty-cycle protocols such as S-MAC. In adaptive-timeout protocols such as T-MAC, the impact is devastating. A constant stream of packets spaced just short of the adaptive timeout will keep nodes constantly awake to receive the unauthenticated traffic.

4.2 Classifying MAC Layer Denial-of-Sleep Attacks

MAC layer denial-of-sleep attacks on WSNs can be categorized based on the level of protocol knowledge required to initiate them and on the level of network penetration achieved by an attacker. Penetration refers to an attacker's ability to read and send trusted traffic. A network is easily penetrated if cryptographic mechanisms are not used for communication or are compromised. While there are mechanisms available for secure communication in WSN, they are not as robust as those found in traditional networks due to resource constraints. Any shared medium can be attacked with physical layer jamming. Jamming, however, is a blunt instrument for executing a denial-of-sleep attack on a WSN. Depending on the MAC protocol, the lifetime of a WSN can be significant even in the face of jamming, requiring that an attacker jam the network for an extended period to render it ineffective. Furthermore, conducting a jamming attack requires considerable resources. A more efficient attack strategy is to use knowledge of MAC protocols to initiate an assault aimed at draining power from sensor platforms, thereby rendering the network

unusable and nullifying any other security mechanisms. In the ensuing discussion, the following three classifications of MAC layer denial-of-sleep attacks are used.

Class I: No protocol knowledge, no ability to penetrate network

With no knowledge of MAC layer protocols, attacks are limited to physical layer jamming and unintelligent replay attacks. In an *unintelligent replay attack*, recorded traffic is replayed into the network causing nodes to waste energy receiving these extra packets. If nodes in the network do not use an anti-replay mechanism, this attack causes the replayed traffic to be forwarded through the network, consuming power at each node on the path to the destination. Undetected replay has the added benefit (to the attacker) of causing the network to resend data that could subvert the network's purpose. For example, as described in Section 4.1.1, replaying traffic in a military sensor network deployed to sense enemy movement could cause combat units to be misdirected.

Class II: Full protocol knowledge, no ability to penetrate network

Traffic analysis can determine which MAC protocol is being used in a sensor network. With this knowledge, an attacker could expand the attack types beyond those listed above to include *intelligent jamming*, injecting unauthenticated unicast or broadcast traffic into the network, or being more selective about replaying previous traffic. Intelligent jamming [89] uses knowledge of link-layer protocols to reduce network throughput without relying on a constant jam signal, for example, by jamming only RTS packets. Such attacks improve over constant physical-layer jamming in that they preserve attacker energy, which can be important if attacking nodes have constraints similar to those of the target nodes. Even when attacker power consumption is not a factor, intelligent jamming might be used to make it more difficult for a network to detect an attack.

If valid source and destination addresses are inserted by an attacker, unauthenticated traffic requires that nodes stay awake to receive packets even if they are later discarded due to invalid authentication. If packets are encrypted, a node must receive the entire packet before decrypting and discarding it. The number of nodes impacted by unauthenticated broadcast traffic depends on the MAC protocol. For example, if the protocol uses a cluster-head or gateway node to authenticate

broadcast traffic before other nodes are compelled to receive it, then only the gateway node energy is impacted. Replay attacks can also be more cleverly executed with knowledge of the protocol, even if the messages cannot be deciphered. It has been shown that if the MAC layer protocol is known, traffic analysis can be used to distinguish data from control traffic [90]. Depending on the protocol, effective denial-of-sleep attacks can be mounted by replaying specific control messages even without the ability to decrypt the traffic. For example, properly timed SYNC retransmission in the S-MAC protocol could potentially prevent nodes from entering their duty/sleep cycle and could keep all nodes in receive mode until their batteries were depleted.

Class III: Full protocol knowledge, network penetrated

Attacks in this category can be devastating to a WSN. With full knowledge of the MAC protocol and the ability to send trusted traffic, an attacker can produce traffic to gain maximum possible impact from denial-of-sleep attack. The types of attacks that can be executed against each MAC protocol and the impact of such attacks are analyzed in Section 4.3.

Table 4.1 classifies the types of denial-of-sleep attacks available based on the attacker's protocol knowledge and ability to penetrate the network. A fourth case, no knowledge of the protocol but an ability to penetrate the network, is not considered since the ability to penetrate the network assumes full knowledge of the MAC layer protocol.

4.3 Effects of Denial-of-Sleep Attacks on MAC Protocols

In this section, attacks from each of the three classifications and their impacts on S-MAC, T-MAC, B-MAC, and G-MAC are analyzed. This section explores the impacts of constant physical layer jamming, unauthenticated broadcast, intelligent replay, and a full domination attack for each of the protocols considered. A full domination attack assumes that the attacker has penetrated the network and has full knowledge of the MAC protocol. In each case, a full domination attack can reduce network lifetime to 10 days for the Mica2 platform and 6 days for the Tmote Sky platform, equivalent to network lifetime under IEEE 802.11 with no power saving features.

Table 4.1: Classification of WSN Denial of Sleep Attacks

Attack Type	*Class I* *No protocol knowledge, no network penetration*	*Class II* *Full protocol knowledge, no network penetration*	*Class III* *Full protocol knowledge, network penetrated*
Constant jammer	√	√	√
Deceptive jammer	√	√	√
Random or reactive jammer	√	√	√
Intelligent jammer		√	√
Untrusted unicast/broadcast		√	√
Trusted rogue unicast/bcast			√
Unintelligent replay	√	√	√
Intelligent replay		√	√
Full domination			√

4.3.1 Network Model

Each network was modeled in MATLAB using similar configurations. The Mica2 models were based on the TinyOS protocol implementations available on Sourceforge.net [32]. Since none of these protocols have been implemented for CC2420-based platforms at the time of this writing, the Tmote Sky models assume the basic functionality of the protocols, adapted to the increased data rate of the CC2420 transceiver and the specified IEEE 802.15.4 interframe spacing durations. Table 4.2 provides network and protocol parameters for the Mica2 and Tmote Sky networks. G-MAC is specifically designed for ZigBee-compliant platforms and is, therefore, not included in the Mica2 models. The B-MAC check interval of 20 ms for the Tmote Sky was determined to give the longest network lifetimes for this model. The LPL sampling cost for the CC2420 is based on the cost of transitioning from power-down mode to receive mode and back [28].

The system models 50 Crossbow Mica2 or Tmote Sky nodes in a single-hop neighborhood. Network lifetimes are based on Mica2 and Tmote Sky power consumption for receive, transmit, and sleep given in Table 2.1. The IEEE 802.11 MAC protocol with no power management provides the baseline case. This MAC is used as the baseline because its behavior is well-known

Table 4.2: Network and Protocol Analytical Model Parameters

	Mica2	Tmote Sky
Number of nodes	50	50
Effective data rate	19.2 kbps	250 kbps
Frame duration	1300 ms	500 ms
Legitimate traffic rate	1 pkt per frame	2 pkts per frame
Payload size	29 bytes	64 bytes
S-MAC duty cycle	10%	10%
T-MAC sleep timeout (TA)	81 ms	13.5 ms
B-MAC check interval	100 ms	20 ms
G-MAC GTIM size	NA	20 bytes
G-MAC collection period	NA	250 ms
G-MAC distribution period	NA	250 ms

and its basic features, including CSMA/CA and the DCF, are replicated in most contention-based WSN MAC protocols. In regular IEEE 802.11, the radio is always in receive mode unless transmitting, resulting in a 10-day lifetime for the Mica2 network and a 6-day lifetime for the Tmote Sky network. All traffic is node-to-node and the effects of all denial-of-sleep attacks are regardless of legitimate traffic rates.

4.3.2 Denial-of-Sleep Attacks and Impacts

Results of each of the attacks are given in Table 4.3. These models assume that all nodes are deployed simultaneously with fresh batteries and that new nodes are not added to the network during its lifetime. Network lifetime is defined as the average time between network deployment and the time that nodes' power supplies are exhausted.

Physical Layer Jamming Attack

The first attack classification in Section 4.2 considers an attacker with no protocol knowledge and no ability to penetrate the network. This classification of attack is modeled using a deceptive jamming attack as described in [52], in which a constant stream of bytes is broadcast into the

Table 4.3: Effects of Denial-of-Sleep on Network Lifetime on the Tmote Sky and Mica2 for Selected MAC Protocols (Network Lifetime Given in Days)

Platform	Routine Network Traffic				Attack Traffic (see Table 4.1 for classifications)			
					Class I	*Class II*		*Class III*
	MAC Protocol	Empty Network	Unicast Traffic	Broadcast Traffic	Deceptive Jamming	DoS Broadcast	Intelligent Replay	Full Domination
Mica2	802.11	10	10	10	10	10	10	10
	S-MAC	99	122	98	99	99	10	10
	T-MAC	227	119	160	10	10	10	10
	B-MAC	775	140	140	10	18	18	10
Tmote Sky	802.11	6	6	6	6	6	6	6
	S-MAC	56	56	56	56	56	6	6
	T-MAC	194	111	133	6	6	6	6
	B-MAC	120	58	58	6	10	10	6
	G-MAC	1024	828	295	237	371	160	6

network. Under this attack, S-MAC is unable to transmit data and nodes remain awake during the entire 10% duty cycle because they are not able to enter NAV sleep. T-MAC fares much worse under this type of attack because nodes must sense an idle channel for the period dictated by the network's adaptive sleep timeout (TA) before going to sleep. Under deceptive jamming, nodes will never sense an idle channel and will delay their sleep cycle continuously, thus remaining in receive mode constantly. This results in a network lifetime of 10 days on the Mica2 and 6 days on the Tmote Sky. B-MAC also suffers under deceptive jamming. A constant stream of preamble bytes forces all nodes in the network into a cycle of receiving and analyzing preamble bytes looking for the beginning of the data transmission. Therefore, nodes remain in receive mode during the entire jamming period, resulting in a lifetime of 10 days on the Mica2 and 6 days on the Tmote Sky. In the G-MAC protocol, the gateway node will remain constantly awake because there is no network idle time to allow it to go to sleep and will, therefore, last for 6 days on the Tmote Sky. Other nodes will wake up and timeout once during each frame listening for a GTIM. Waking up for these small GTIM messages results in 0.16% duty cycle and a lifetime of 1287 days (or battery shelf-life) for all of the other nodes in the network. A more effective attack against G-MAC would be to periodically lift the jamming attack so that a new gateway is elected, thereby distributing the maximum power draw among all nodes. This would cause the average node per frame power

consumption to be

$$P_{NodeAverage} = \frac{(P_{Gateway}) + (n-1)(P_{OtherNodes})}{n}, \tag{4.4}$$

where n is the number of nodes, $P_{Gateway}$ is the gateway power consumption while always awake and $P_{OtherNodes}$ is the power consumption of the rest of the nodes. Under this attack, G-MAC network lifetime is reduced to 237 days.

DoS Unauthenticated Broadcast Attack

The second attack classification considers an attacker with full protocol knowledge, but no ability to penetrate the network. In this case, the attacker broadcasts traffic into the network following all MAC protocol rules for timing and collision avoidance. Under S-MAC, T-MAC, and B-MAC, these messages are received by all nodes, but are discarded because they cannot be authenticated. Even though the broadcast messages are not authenticated, the fact that all nodes stay awake to receive the messages has significant impact on network lifetime. Sensor nodes using the S-MAC protocol are unable to save power using NAV sleep, keeping them in receive mode during their entire 10% duty cycle and resulting in a network lifetime of 99 days on the Mica2 and 56 days on the Tmote Sky. To minimize network lifetime for networks running the T-MAC protocol, short broadcast messages are sent at a period just short of the adaptive timeout (TA) to prevent nodes from going to sleep. This attack will keep the sensor nodes awake during the entire frame and reduce lifetime to 10 and 6 days for the Mica2 and Tmote Sky, respectively, while keeping the attacker's power requirements to a minimum. Since B-MAC nodes do not synchronize schedules, the most effective broadcast attack against B-MAC is to transmit back-to-back packets as described in Section 4.1.3, which keeps victim nodes awake more than half of the time. This attack results in a network lifetime of 18 days for the Mica2 and 10 days for the Tmote Sky.

Under G-MAC, only the gateway receives the broadcast FRTS during the collection period. Since it cannot be authenticated, the broadcast message is not scheduled during the distribution period. To maximize the impact of this attack on G-MAC, the gateway should be kept awake during the entire collection period. The G-MAC gateway uses the same adaptive timeout mechanism as

T-MAC to go to sleep during the contention period if there is no more traffic for it. An attacker should, therefore, send short broadcast messages at a rate just short of the adaptive timeout period to prevent the gateway from transitioning to sleep mode. Assuming no other traffic in the network, the other nodes would only wake up to receive an empty GTIM and, then, sleep for the remainder of the time, resulting in an overall network lifetime of 371 days on the Tmote Sky. Any legitimate network traffic in addition to the unauthenticated broadcast packets further reduces this lifetime.

Intelligent Replay Attack

Another attack in the category of full protocol knowledge but no network penetration is an intelligent replay attack. If an attacker can distinguish control traffic from data traffic under S-MAC, SYNC packets can be replayed at an interval short of the sensor cluster's duty cycle, effectively restarting the duty cycle and pushing back the sleep period each time. This would keep all nodes awake until they run out of power. In G-MAC, FRTS messages should be replayed such that the corresponding NAV periods fill the contention-free portion of each frame. For a message size of 64 bytes, 75 FRTSs would fill the contention-free period ensuring that at least one node is awake at all times. This effect, combined with a longer GTIM message which all nodes must receive, results in a network lifetime of 160 days, assuming all of the FRTSs are for unicast packets. If any of the replayed FRTS messages happen to be broadcast FRTSs, network lifetime is further degraded because all nodes must wake up during the contention-free period to listen for the broadcasts. If only 10% of the FRTS messages, or 7 FRTS messages per frame, are for broadcasts, the network life is cut by almost 50%, dropping to 83 days. The worst case is if all FRTSs are for broadcast messages. In this case, network lifetime is reduced to 12 days as discussed below. Even if the message size is not known, the attacker could simply attempt to resend recorded FRTS messages until the gateway quits accepting them. The maximum number of FRTSs that an attacker can send can be determined based on the length of the collection period as follows:

$$N_{FRTS} = \left(\frac{T_{CollectionPeriod}}{T_{Cont} + T_{DIFS} + T_{FRTS} + T_{SIFS} + T_{ACK}} \right), \tag{4.5}$$

where T_{Cont} is the average contention period, T_{DIFS} and T_{SIFS} are the IEEE 802.11 distributed

and short inter-frame space periods, T_{FRTS} is the time required to send a 13-byte FRTS message, and T_{ACK} is the time required for the gateway to send a 5-byte acknowledgment. With a 250-ms collection period, a maximum of 138 FRTS messages can be sent. With the potential for 138 FRTSs, the attacker can easily maximize traffic during the contention-free period.

Full Domination Attack

The final attack classification is one in which an attacker has full protocol knowledge and has penetrated the network. This type of attack might be mounted using one or more compromised nodes in the network. Once this level of network penetration is achieved, all of the MAC protocols are susceptible to worst-case power consumption. An attack against S-MAC is simply to send a SYNC message at a frequency just short of the duty cycle to keep delaying the transition to sleep mode. T-MAC network lifetime is minimized by continually sending packets at an interval slightly shorter than the adaptive timeout (TA) so that none of the nodes can ever transition to sleep. Although not efficient for the attacker, a deceptive jamming attack is the most effective attack against B-MAC. A full domination attack against G-MAC has the attacker broadcasting a bogus GTIM packet filled with broadcast messages before the gateway node, as described in Section 4.1.4. If the attacker fills the GTIM with broadcast messages that fill the entire frame up to the next GTIM, all nodes will remain in receive mode during the entire frame waiting for the broadcast traffic. By repeating this pattern for each frame, all nodes are kept awake and the network lifetime is reduced to 6 days on the Tmote Sky. A simpler full domination attack against G-MAC would simply have the attacker send broadcast FRTSs to the gateway such that the contention-free period is filled with broadcast messages. With 89 64-byte packets, the 250-ms contention-free period would be filled, resulting in a 50% duty cycle for all nodes and a network lifetime of 12 days.

4.3.3 Discussion

The analysis of these attacks shows that with knowledge of the MAC protocol, even without the ability to penetrate encryption, attacks can be constructed that have more significant impact on

network lifetime than even constant physical layer jamming. Some attacks not only reduce network lifetime significantly, but they are subtle enough that the network may not even be able to identify that it is under attack. Furthermore, these attacks can be sustained longer because the attacker can conserve power by not transmitting a constant jam signal. The efficiency of these attacks is explored in the next section.

4.4 Denial-of-Sleep Attack Implementations

Selected attacks were implemented and tested on the Crossbow Mica2 wireless sensor platform to validate the threat of the denial-of-sleep attacks on S-MAC, T-MAC, and B-MAC described in Section 4.3. The attacks were programmed in nesC, the programming language for TinyOS [16]. Attacks on G-MAC were not tested because the G-MAC protocol has not yet been implemented on a hardware platform. This section describes the details of these attacks and analyzes the efficiency with which they can be executed. An attack that minimizes power consumption for the attacker is preferred. If the attacker uses less power than the victim nodes, attacks can be executed using mote-class devices, which are easier to deploy and are less prone to detection because their physical-layer properties are similar to those of the deployed network devices. By following the carrier-sense based collision avoidance mechanisms inherent in these protocols, some of these denial-of-sleep attacks can be carried out in such a way that the targeted networks maintain throughput and latency similar to that of the network when it is not under attack. This is discussed further in Chapter 7.

The Mica2 platform was selected for these tests because it is a widely-used commercially-available sensor node platform that is compatible with the TinyOS operating system and with the S-MAC, T-MAC, and B-MAC implementations that were included in the TinyOS CVS release available on Sourceforge.net [32] when these tests were developed[3]. All experiments were carried out in a laboratory setting, with nodes approximately 1.5 meters apart in an open area. S-MAC and T-MAC parameters were used in their default settings as downloaded from Sourceforge except

[3]All tests with S-MAC, T-MAC, and B-MAC were performed with the implementations in `tinyos-1.x/contrib/s-mac/`, `tinyos-1.x/contrib/t-mac`, and `tinyos-1.x/tos/platform/mica2` in the TinyOS CVS repository [32] as of Aug. 1, 2006.

where specified otherwise. Those parameters are discussed in the following sections where appropriate. The B-MAC implementation used in these experiments has six selectable LPL modes, which correspond to check intervals of 20, 85, 135, 185, 485, and 1085 ms. For these experiments, B-MAC was configured using LPL-mode 3, which has a 135-ms check interval and uses 371-byte preambles. Results presented by Polastre, et al. [8], show that this check interval is ideal for small networks (five to ten nodes) such as the one used in these experiments. In the S-MAC and T-MAC experiments, nodes were turned on and allowed to synchronize by exchanging SYNC packets before attacks were initiated. B-MAC nodes were turned on and allowed to enter LPL-mode before attacks were begun. Because these tests were not designed to measure the effects of the attacks on network performance metrics such as throughput or latency, there was no traffic in the network other than MAC-layer SYNC packets used by S-MAC and T-MAC to synchronize schedules. Figure 4.2 depicts the experimental layout. For the SYNC attacks described below, the cluster-head node depicted in Figure 4.2 was started first so that its schedule would be adopted by the other nodes in the one-hop neighborhood.

A signal generator was used for the constant physical-layer jamming attacks against all protocols. A Mica2 sensor device identical to the victim nodes was used for all other attacks.

The following paragraphs first describe and analyze attacks on networks using the S-MAC protocol corresponding to the three attack classes described in Section 4.2. They go on to analyze attacks on T-MAC and B-MAC. Finally, the efficiency with which these attacks can be executed is

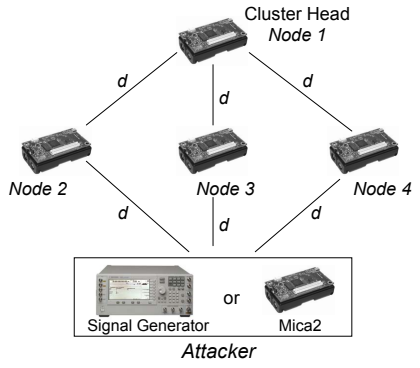


Figure 4.2: Experimental setup for denial-of-sleep attacks ($d \simeq 1.5$ m).[4]

[4]Copyright IEEE. Used through residual rights retained from [24].

examined.

4.4.1 Attacks on S-MAC

Class I: Constant Jamming Attack

Two types of constant jamming attacks were conducted to determine the impact of each on the S-MAC protocol. The first attack was executed using a signal generator to transmit a constant 10-dBm signal on the same frequency as the targeted sensor devices. The jam signal was fine-tuned to the frequency of the sensor devices using a spectrum analyzer. Jamming was initiated after the targeted devices synchronized schedules. As expected, constant jamming did not impact the devices' ability to maintain their default 10% duty cycle, so nodes were not prevented from sleeping 90% of the time. The second type of attack was the deceptive jamming attack described in 2.5.4. This attack was executed using a Mica2 device to transmit a constant stream of TinyOS preamble bytes (0xAA) after the victim nodes synchronized their schedules. This attack was also unsuccessful in keeping the victim nodes awake beyond their default duty cycle. These tests show that while an attacker is able to prevent communication between nodes using both constant jamming and deceptive jamming, it is not able to impact the lifetime of an S-MAC network. Furthermore, these attacks are inefficient because the attacker is awake and transmitting constantly while the victim nodes maintain their fixed duty cycle.

Class II: Intelligent Replay Attack

The intelligent replay attack implemented on the Mica2 takes advantage of the S-MAC SYNC vulnerability detailed in Section 4.1.1. To execute an efficient SYNC replay attack, an attacker must be able to closely estimate the value of *sleepTime* in replayed SYNC packets. If traffic is authenticated, but not encrypted, the attacker can read (but not modify) the *sleepTime* value and send SYNC packets at a rate slightly less than $\left\lfloor \frac{sleepTime}{2} \right\rfloor$. If packets are encrypted and the attacker cannot read the value of sleepTime in the recorded packet, the attacker can mount an efficient attack by assuming a minimum value for *sleepTime* in a SYNC packet. The value of *sleepTime* is set based

on the value of a decrementing counter that begins when the node transitions from the sleep portion of a frame to the SYNC period in the next frame. On the Mica2 implementation using default S-MAC settings, this counter is initialized to 130 and decrements every 1 ms. Each node checks whether it should send a SYNC packet at the beginning of the SYNC period and, if so, begins to backoff for the duration of a DIFS (10 ms) plus a random number of milliseconds within its 15-ms contention window, for a maximum backoff of 25 ms. Using these values, the minimum value of *sleepTime* is 105 ms. Experiments were conducted under the conditions shown in Figure 4.2 using a rate of 50 ms between packets, which is slightly less than one-half of the minimum assumed sleepTime of 105 ms. This SYNC interval proved to be effective and all nodes except the cluster-head were kept awake constantly. The cluster-head node is not kept awake because it disregards the counterfeit SYNC packet with its own address as the source.

The S-MAC implementation for the Mica2 uses an effective data rate of 19.2 Kbps using Manchester encoding [21]. At that data rate, a SYNC packet, including preamble and all headers, takes 13.6 ms to send. The Mica2 requires 2.6 ms transition time from transmit mode to sleep mode and back to transmit mode [8]. The attacker's radio must, therefore, be awake for 16.2 ms of every 50.0 ms, or 32% of the time.

Class III: Full Domination Attack

In a full domination attack, an attacker is assumed to be able to set the value of *sleepTime* in an S-MAC SYNC packet, either because the network is not using encryption or authentication, or because the attacker has broken the encryption mechanism. The *sleepTime* in a SYNC packet is 2 bytes, so the maximum value that could be used as the *sleepTime* is 65,535 ms. An attacker cannot, however, mount an effective attack by simply setting *sleepTime* to this maximum value and sending a SYNC packet at a rate of $\lfloor 65535/2 \rfloor$ ms. This is because the bogus SYNC packets are sent using the same node identification number as the cluster-head so that the victim nodes do not interpret the SYNC as a new, additional schedule in the network, but as a modification to their primary schedule. By setting a *sleepTime* that is longer than the frame length, none of the nodes in the cluster except the cluster-head is able to enter sleep mode. To maintain this attack, the

attacking node must send out its forged SYNC packet after each SYNC sent by the cluster-head. The default S-MAC configuration has the cluster-head sending a SYNC packet every 10th frame, or every 13,000 ms. The full domination attack was implemented using these parameters and using a *sleepTime* value of 6,000 ms, which is longer than the frame duration. As expected, all attacked nodes were kept awake 100% of the time except for the cluster-head node, which maintained its default 10% duty cycle.

Since the attacker is awake during every tenth duty cycle to receive the cluster-head's SYNC packet and send its own, the attacking node is awake for 130 ms out of every 13,000 ms, or 1% of the time. Note that the SYNC attacks described here rapidly drain the batteries of all nodes except for the cluster-head nodes (and the attacker). Once all non-cluster-head nodes have been exhausted, the remaining nodes will form clusters if they are within communication range and can be attacked as described above. If the remaining nodes cannot form clusters because they are too far apart to communicate, the network is no longer usable.

4.4.2 Attacks on T-MAC

Class I: Constant Jamming Attack

A constant jamming attack using a signal generator has the same effect on T-MAC as on S-MAC. Nodes are prevented from exchanging traffic, but they are not prevented from placing the radio in sleep mode after the adaptive timeout has expired. The deceptive jamming attack, however, forces the victim nodes to reset their time-to-sleep based on the *TA* value after each received start symbol. This attack, therefore, keeps the victim devices awake 100% of the time. While effective in terms of a denial-of-sleep attack, it is not efficient in that the attacker must be awake and transmitting 100% of the time and could potentially deplete its energy supply before the victim nodes.

Class II: Adaptive Timeout Attack

Recall from Equation 4.2 that for the currently available T-MAC implementation, $TA = 1.5 \times (T_{RTS\_CW} + (2 \times T_{Wait\_CTS}))$. For the Mica2, $T_{RTS\_CW} = 12$ ms and $T_{Wait\_CTS} = 21.2$ ms.

Substituting these values into Equation 4.2 gives a duration of 81.6 ms for the adaptive timeout. To confirm the effectiveness of an attack taking advantage of the adaptive timeout mechanism, a network running the T-MAC protocol was attacked by sending a stream of one-byte packets. The attack used an interval of 70 ms between packets, which is 10 ms shorter than the TA value to account for processing, transmission, and propagation delays. Unfortunately, despite significant debugging, the T-MAC testbed on which this attack was implemented maintained a higher-than-normal packet loss rate of 20 to 25%, both for routine T-MAC traffic and for attack traffic. The T-MAC protocol was originally developed for the EYES WSN platform [91] and later ported to the Mica2. The physical layer used in this port is closely tied to the MAC layer and is not completely reliable. The attack at 70 ms was, therefore, not as effective as expected, keeping the network awake an average of 83% of the time. By setting the attack interval to 35 ms, the attack effectiveness increased, keeping targeted nodes awake an average of 92% of the time. To determine the effectiveness of the adaptive timeout attack on a network with better physical layer properties, the 70 ms attack was carried out on a simulated network using of the *Avrora* [84] emulation environment[5]. Avrora emulates the Mica2 platform on a personal computer and allows the user to directly run TinyOS code that is compiled for the Mica2. When the T-MAC physical layer is used in the Avrora emulator, there is no packet loss due to channel fading or background noise, so the only packet loss is that due to collisions. Under these circumstances, the adaptive timeout attack worked as intended, keeping the victim devices awake 100% of the time.

It takes the attacker 2.9 ms to send a 1-byte packet, along with preamble. With a total of 2.6 ms to wake up before sending the packet and go to sleep afterward, the attacking node must be awake 5.5 ms every 70 ms, or 8% of the time.

Class III: Full Domination Attack

While T-MAC uses SYNC messages which specify *sleepTime* much like S-MAC does, this mechanism cannot be exploited to keep nodes awake in the same way it can be done in an S-MAC network. The T-MAC SYNC messages help to keep nodes synchronized and inform nodes of

[5]Tests with Avrora were performed using version 1.7.59, the version available in the Avrora CVS repository [32] as of 1 August, 2006.

other active schedules, but the time that a node is awake is still dictated by the adaptive timeout (*TA*). Therefore, the most efficient denial-of-sleep attack on T-MAC is the adaptive timeout attack described in the previous section. It should be noted that to mount this attack against T-MAC, it is simply necessary to know that the network is running the T-MAC protocol. No penetration of link-layer encryption or authentication is required.

4.4.3 Attacks on B-MAC

Class I: Constant Jamming Attack

A constant jamming attack using the signal generator has the same affect on B-MAC as on the other protocols. It is recognized as background noise and is ignored. As a result, this jamming prevents nodes from exchanging packets, but it does not prevent them from entering sleep mode. Under the experimental configuration, nodes are awake to conduct an LPL sample for approximately 2.6 ms out of every 135 ms, for an awake percentage of 2%. Unlike constant jamming, deceptive jamming prevents B-MAC nodes from entering sleep mode as discussed in Section 4.1.3. As soon as a B-MAC node awakens and recognizes a preamble being transmitted, it begins a cycle of receiving bytes and searching for a pair of *synchronization bytes* that indicate the beginning of the expected data packet. The victim node will sample incoming preamble bytes indefinitely until a packet is observed, remaining awake 100% of the time. This attack is effective as a denial-of-sleep attack, but the attacker must also be active 100% of the time, making it inefficient. It is especially costly for the attacker on the Mica2 platform since the power consumption during data transmission is approximately twice that of receive-mode power consumption. On the Tmote Sky platform, transmit cost is lower than receive cost making this a more practical attack.

Class II: Unauthenticated Broadcast Attack

The unauthenticated broadcast attack described in Section 4.3.2 is achieved by sending a constant stream of back-to-back broadcast packets. An attacker might also record a maximum-size broadcast packet and replay this packet into the network. Both attacks have the same denial-of-sleep

impact. Table 4.3 gives the expected B-MAC network lifetime under such an attack, as determined using (4.3). In practice, however, the percentage of victim node awake time was higher than that predicted by (4.3). This is because in the Mica2 B-MAC implementation, nodes do not immediately transition to sleep mode when packet reception is complete. Instead, they remain in receive mode, only transitioning to sleep mode when a periodic timer fires and a check is made to determine whether the node is in transmit, receive, or idle mode (not actively sending or receiving packets). Under this attack, victim nodes were kept awake an average of 79% of the time, making it much more effective than expected. Since the attacker must transmit 100% of the time, however, it is not as efficient an attack as constant deceptive jamming. The advantages of this attack over deceptive jamming are twofold. First, the attacker can follow MAC-layer collision avoidance rules and only transmit when the channel is idle, thus allowing legitimate traffic to be transmitted and increasing the subtlety of the attack. Second, this attack would not be recognized as a jamming attack if the network were protected from jamming using techniques described by Xu et al. in [52] because packet delivery ratio is not adversely affected.

Class III: Full Domination Attack

The most effective attack against B-MAC is the deceptive jamming attack. Despite its inefficiency, this attack keeps victim nodes awake 100% of the time. It is also easy to execute. Simply recognizing that B-MAC is being used in a network is all the information an attacker needs to keep network nodes in constant receive mode, rapidly draining their energy supplies.

4.4.4 Attack Efficiency

Table 4.4 gives the efficiency of the various denial-of-sleep attacks on S-MAC, T-MAC, and B-MAC described in this section. The *Active Ratio* (*AR*) is the ratio of the percentage of time that the attacker is awake to the percentage of time that victim nodes are awake during the period that the attack is active. This metric is used because of the disparity between transmit and receive power consumption among sensor platforms. For example, for the Mica2, transmit cost is twice that of receive cost, while for the Tmote Sky, receive cost is slightly higher than transmit cost. Assuming

Table 4.4: Efficiency of Denial-of-Sleep Attacks

S-MAC				
Attack Class	Attack	Attacker Awake	Victim Awake	Active Ratio (AR)
I	PHY jamming	100%	10%	10.0
I	Deceptive jamming	100%	10%	10.0
II	SYNC replay	32%	100%	0.32
III	Efficient SYNC	1%	100%	0.01
T-MAC				
Attack Class	Attack	Attacker Awake	Victim Awake	Active Ratio (AR)
I	PHY jamming	100%	6%	17.0
I	Deceptive jamming	100%	100%	1.0
II	TA attack (Mica2)	8%	83%	0.09
II	TA attack (Avrora)	8%	100%	0.08
III	TA attack (Avrora)	8%	100%	0.08
B-MAC				
Attack Class	Attack	Attacker Awake	Victim Awake	Active Ratio (AR)
I	PHY jamming	100%	2%	50.0
I	Deceptive jamming	100%	100%	1.0
II	Broadcast attack	100%	79%	1.27
III	Deceptive jamming	100%	100%	1.0

equivalent energy consumption, *AR* is the units of energy expended by the attacker to deplete one unit of energy from the victim(s). An *AR* value less than 1.0 means that the victim nodes are awake for a higher percentage of time than the attacker. Constant physical-layer jamming and deceptive jamming are particularly inefficient when mounted against an S-MAC network. Even though the deceptive jamming attack keeps T-MAC nodes awake permanently, it requires the attacker to be constantly awake and is, therefore, not an efficient attack on T-MAC either. There is no energy-efficient denial-of-sleep attack against B-MAC. The most efficient attack against B-MAC is the constant deceptive jamming attack, which keeps both victims and attackers awake 100% of the time. The efficiency of attacks such as the full domination attack on S-MAC makes clear the necessity for strong link-layer authentication in deployed sensor networks. The attacks presented in which the nature and parameters of the MAC protocol are known but link-layer authentication is *not* compromised are less efficient, however, they show that an attacker can still use a mote-

class device to quickly drain the energy reserves of sensor devices using S-MAC or T-MAC with significant power to spare.

4.4.5 Discussion

The relative ease with which the denial-of-sleep attacks described in this section were implemented and the efficiency with which they can be carried out indicate that energy-efficient MAC protocols must be designed with security in mind. In the cases described above, the very mechanisms used in these protocols to conserve energy are exploited to rapidly drain power on the devices that use them.

4.5 Determining MAC Protocols via Traffic Analysis

To launch the Class II and Class III attacks described in the previous sections, an attacker must not only understand the protocols, but must also be able to determine which protocol is being used in a particular network. The various WSN MAC protocols examined here can be distinguished based on traffic characteristics such as packet size, packet timing, and preamble length. Certain information about the configuration of each protocol that is useful in constructing denial-of-sleep attacks can also be gleaned from observing network traffic.

4.5.1 S-MAC Traffic Characteristics

All S-MAC control packets, which include RTS, CTS, ACK, and SYNC packets, are ten bytes long. S-MAC data packet sizes range from a minimum of 12 bytes to a maximum of 250 bytes, with headers included. Unicast packets use the DCF to account for hidden terminals, so they are preceded by an RTS/CTS exchange and followed by an acknowledgment. Figure 4.3 depicts the format of the various S-MAC frames.

The *type* field, which is included in all packet formats, is 0x00 for data packets, 0x11 for RTS, 0x20 for CTS, 0x30 for ACK, and 0x40 for SYNC packets. Because S-MAC uses the DCF, unicast traffic within a cluster is identified by a ten-byte packet of type 0x11, followed by one of

Len (1)	Type (1)	Destination (2)	Source (2)	Duration (2)	CRC (2)

(a) Control packet (RTS, CTS, ACK).

Len (1)	Type (1)	Source (2)	State (1)	Seq# (1)	SleepTime (2)	CRC (2)

(b) SYNC packet.

Len (1)	Type (1)	Destination (2)	Source (2)	Duration (2)	Frag# (1)
Data (0 - 239)					CRC (2)

(c) Data packet.

Figure 4.3: S-MAC packet formats.

type 0x20, followed by a data packet of type 0x00 of 12 or more bytes, followed by a ten-byte packet of type 0x30. Even if the packets are encrypted, this exchange is easily recognizable and identifies the network as most likely being an S-MAC or T-MAC network.

Figure 4.4 depicts the first 30 seconds of network traffic in a 3-node S-MAC network running on Mica2s. Each vertical bar represents one packet and the height of each bar indicates the packet size in bytes. The periodic nature of network traffic imposed by S-MAC's duty cycle is apparent based on the bursts of packets every 1.3 seconds. The first three duty cycles have only SYNC packets and remaining duty cycles have a mixture of 20-byte broadcast and unicast packets along with S-MAC control packets.

Figure 4.5 is a close-up of three consecutive frames from this same trace. The first frame includes a SYNC packet, followed by a broadcast packet. The second frame includes a SYNC, followed by two RTS/CTS/Data/ACK exchanges. The third frame is a SYNC followed by a broadcast. Since control packets and SYNC packets are all ten bytes when unencrypted, they will most likely be identical in size, though slightly larger, when encrypted. TinySec, for example, has a constant 1-byte per packet overhead for authentication and 6-byte overhead for encryption [40]. If two control packets are observed immediately before a data packet, these packets are most likely to be an RTS, followed by a CTS.

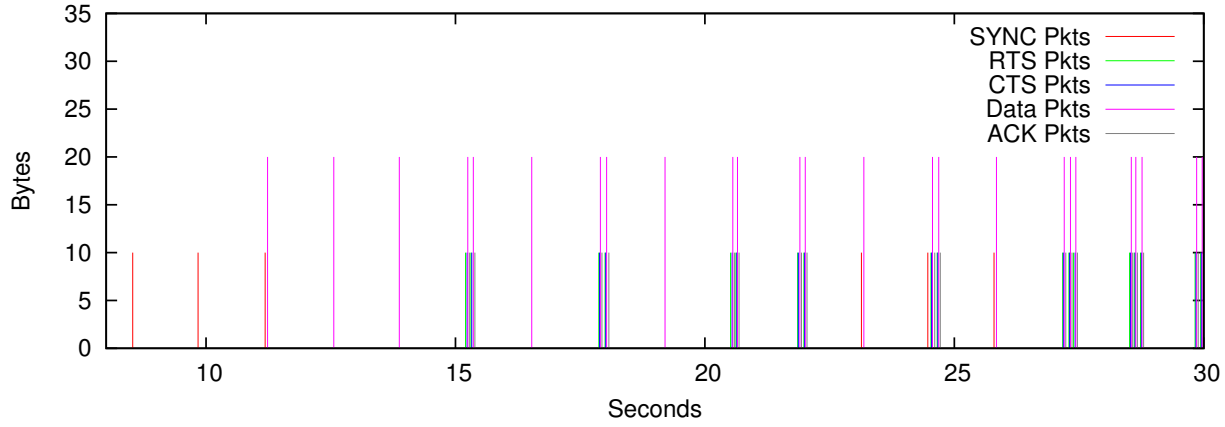


Figure 4.4: First 30 seconds of S-MAC traffic in a 3-node network of Mica2s. Each vertical bar represents a single packet.

4.5.2 T-MAC Traffic Characteristics

T-MAC uses similar control packets as S-MAC, however, the sizes are different. T-MAC SYNC packets are eight bytes, RTS, CTS, and ACK packets are 12 bytes and data packets range from 11 to 250 bytes. The type fields for data, RTS, CTS, ACK, and SYNC packets are 0x06, 0x07, 0x08, 0x09, and 0x10, respectively. The formats of T-MAC packets are given in Figure 4.6.

A T-MAC network is recognized in much the same way as an S-MAC one. Unencrypted packets are identifiable by the DCF packet exchanges and the packet type fields. Encrypted T-MAC networks are identified by the different sizes of SYNC versus control (RTS, CTS, and ACK) packets and data packets.

4.5.3 B-MAC Traffic Characteristics

The B-MAC protocol is lightweight and does not define its own link-layer control packets. This protocol focuses solely on medium access and does not implement an RTS/CTS exchange, counting on higher-layer protocols or applications to synchronize data exchange to avoid collisions. An analysis of higher-layer protocols and their associated packet formats is outside the scope of this work. Furthermore, the wide range of existing WSN protocols and applications make an exhaustive

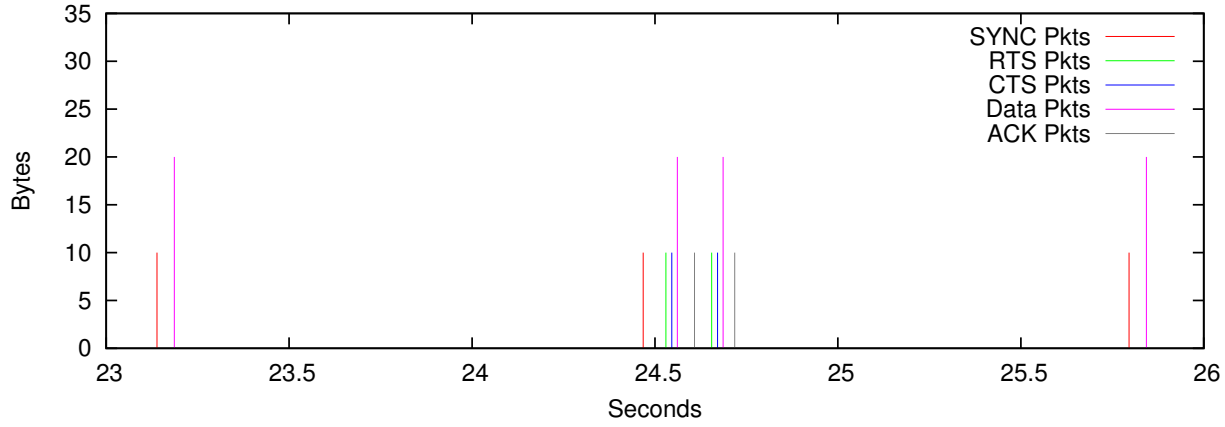


Figure 4.5: Three consecutive frames of S-MAC traffic. Even if the packet types cannot be read due to encryption, the size and sequencing of the packets makes them identifiable.

Len (1)	Type (1)	Source (2)	SleepTime (2)	Destination (2)	Duration (2)	CRC (2)

(a) Control packet (RTS, CTS, ACK).

Len (1)	Type (1)	Source (2)	SleepTime (2)	CRC (2)

(b) SYNC packet.

Len (1)	Type (1)	Source (2)	SleepTime (2)	Destination (2)
Data (0 – 240)				CRC (2)

(c) Data packet.

Figure 4.6: T-MAC packet formats.

analysis of this design space unrealistic.

B-MAC networks can best be recognized by the long preambles that precede all packets as described in Section 2.3.3. The default B-MAC preamble size on the Mica2 platform is 127 bytes [8]. The preambles for the Mica2 implementations of S-MAC and T-MAC are 40 bytes and 2 bytes, respectively. These long preambles can also be used to determine B-MAC check intervals, as the B-MAC preamble duration is slightly longer than the network's check interval.

4.5.4 G-MAC Traffic Characteristics

Without an implementation of G-MAC, characterization of G-MAC traffic must be simulated. For this analysis, G-MAC exchanges were modeled in MATLAB using the same parameters as the denial-of-sleep models described in Section 4.3.1. Traffic is modeled as two unicast packets and one broadcast packet per frame. Figure 4.7 depicts three consecutive frames of G-MAC traffic in a single-hop network. The periodic nature of the data communication, with groups of short FRTS packets and groups of longer data packets, characterizes a G-MAC network.

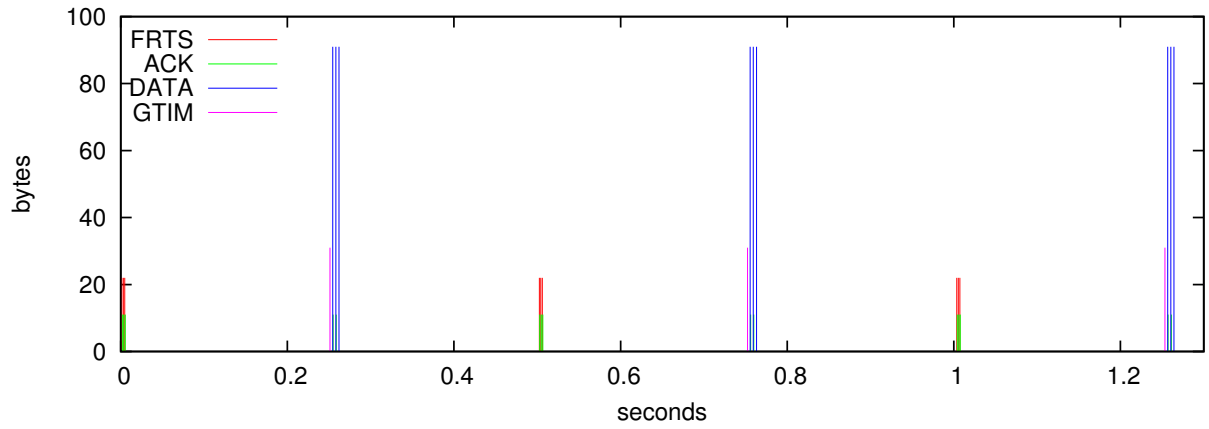


Figure 4.7: Three frames of G-MAC traffic.

Figure 4.8 is a close-up view of one frame of G-MAC traffic. The three 22-byte FRTS packets, each followed by an 11-byte ACK packet, are easily identifiable. Since the contention period of the G-MAC frame uses the same adaptive timeout mechanism as T-MAC, these FRTS packets will always be clustered at the beginning of the frame. Near the 250 ms point, a traffic indication message, or GTIM, is followed by three data packets. Unicast packets are distinguished from the broadcast packet by the ACKs that follow the unicast packets. The number of bytes in the GTIM and the data packets may differ, but the sizes of FRTS and ACK packets are fixed. Again, encryption mechanisms might modify the sizes of the packets, but will not significantly change the relative size differences or the packet timings.

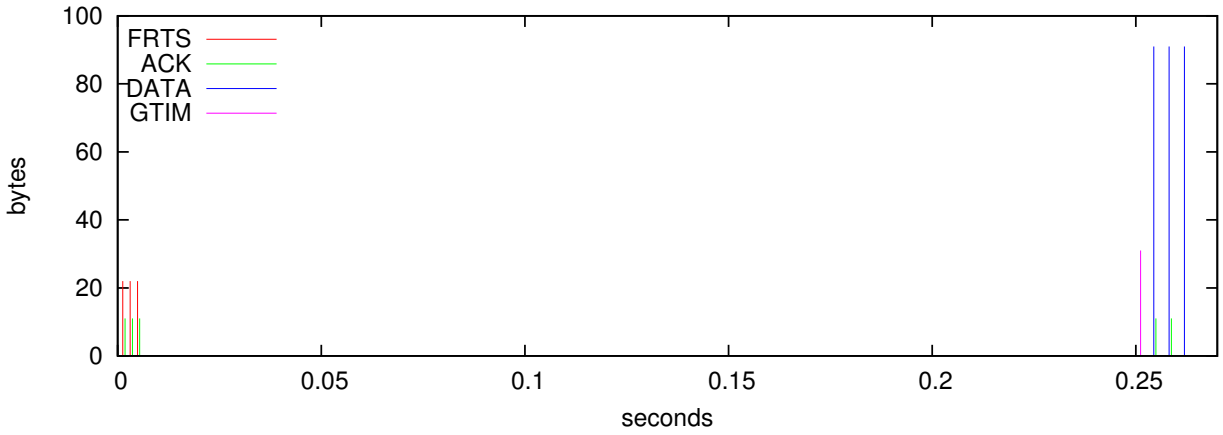


Figure 4.8: Traffic from one G-MAC frame with two unicast packets and one broadcast packet.

4.6 Summary

In this chapter, the denial of sleep problem is explored, first by identifying vulnerabilities in each of the WSN MAC protocols considered, and then by classifying potential attacks based on the attacker's level of protocol knowledge and ability to penetrate network authentication. While the most effective and efficient denial-of-sleep attacks require MAC protocol knowledge and penetration of network security measures, some very effective attacks are still possible without network security penetration.

Analytical models were used to determine the impact of various attacks on four leading WSN MAC protocols. For each of the protocols examined, attacks are described that reduce network lifetime to the minimum possible by keeping sensor node transceivers in receive mode 100% of the time. This results in network lifetimes of 10 days for networks using the Mica2 platform and 6 days for Tmote Sky networks.

To validate these models, several attacks were implemented and results of these attacks on S-MAC, T-MAC, and B-MAC were given. These results demonstrate that denial-of-sleep is a real threat to WSN deployments using techniques that are both efficient and straightforward to implement. Finally, this chapter shows how the MAC protocol used in a particular WSN deployment can be identified by analyzing WSN traffic.

The following chapter describes a suite of tools that are shown to prevent or mitigate attacks across the range of classification given in this chapter, thereby preserving sensor network lifetime in the face of denial-of-sleep attacks. Subsequent chapters describe each of these mechanisms in detail.

Chapter 5

An Overview of the Caisson System

The function of a strong position is to make the
forces holding it practically unassailable.
- Karl von Clausewitz, *On War*

This chapter explores the classes of denial-of-sleep defense required to defeat the attacks described in Chapter 4. It goes on to introduce the components of the Clustered Anti Sleep-deprivation for Sensor Networks (Caisson) system. These components include Clustered Anti-replay Protection (CARP), Clustered Adaptive Rate Limiting (CARL), and the Sensor Anti Jamming Engine (SAJE). Caisson components are selectable at application compile-time based on the vulnerabilities of the MAC protocol used in the target network and the anticipated threat model for the deployment area.

Section 5.1 reviews the classes of denial-of-sleep defense required to prevent or mitigate the effects of denial-of-sleep attacks on WSN. Section 5.2 introduces the three Caisson components listed above. Subsequent chapters describe each of these mechanisms in detail and provide results from experiments to show their effectiveness in mitigating each type of denial of sleep attack and the overheads incurred by each.

5.1 Classes of Denial-of-Sleep Defense

The vulnerabilities that lead to potential denial-of-sleep attacks identified in Section 4.1.5 are susceptibities to control-packet spoofing, packet replay, jamming, and unauthenticated broadcast. These vulnerabilities require corresponding protections as described below.

Packet Spoofing Protection

Packet spoofing takes advantage of networks in which traffic is either unauthenticated, unencrypted, or both. Encryption techniques are available that severely limit the ability of an attacker to generate authentic-looking packets. This research assumes that the networks in question use one of the available authentication mechanisms described in Section 2.5.2 to prevent packet spoofing.

Anti-replay Protection

Section 2.5.3 provides background on currently available options for anti-replay protection in WSN. To prevent the kinds of attacks considered in this research, nodes must be prevented from accepting replayed link-layer control packets such as those associated with the S-MAC, T-MAC, and G-MAC protocols. These replayed packets do not necessarily have to come from within a victim node's cluster. An attacker might record a packet from one portion of the network and replay it elsewhere, making it appear to originate from a legitimate node that is new to a cluster. As described in Section 2.5.3, current solutions for anti-replay are insufficient either because they are too resource intensive, requiring a large anti-replay table to be stored on every node, or they are tied in hardware to access-control list entries that make them inflexible and prevent their use except in trivial situations.

Unauthenticated Broadcast Protection

Attacks using unauthenticated broadcasts are particularly harmful to contention-based MAC protocols. Since nodes must receive packets in their entirety before the full message authentication code can be computed, an attacker sending a stream of back-to-back packets can keep victim nodes

awake without even penetrating the authentication protocol. By obeying collision-avoidance rules and deferring to legitimate network packets, this attack can also be employed without effecting the legitimate throughput of the victim network, making it particularly difficult to identify.

An unauthenticated broadcast attack might be mitigated using simple rate-limiting. If nodes are prevented from remaining awake beyond a specified maximum duty cycle, unauthenticated broadcast-based attacks can have only a limited negative effect on energy consumption. Strict rate-limiting, however, might prevent legitimate network traffic. Some sensor network deployment scenarios are characterized by long periods of low activity, with occasional periods of heavy activity. An example of such a network is one deployed in a military scenario to sense vehicle movement in a remote area. Under normal circumstances, the network would have a low traffic rate, with nodes exchanging packets just often enough to indicate their continued presence. When vehicles enter the area, however, the nodes nearest the vehicles might transfer data at a high rate as movement is tracked through the area. Limiting the amount of time that these nodes can be active could prohibit data exchanges that are critical to maintaining location information on the intruding vehicles. A fixed rate limit is, therefore, not desirable for many types of networks.

Jamming Protection

Xu, et al., show that jamming is identifiable through analysis of received signal strength, carrier sense time (the average time required to sense an idle channel), and packet delivery ratio [52]. With the ability to identify jamming, nodes can be placed into low-duty cycle operation to save energy, waking only periodically to sense the channel to determine if the jamming is still ongoing. This must be done carefully, however, so that it does not provide an avenue for an attacker to mount an efficient denial-of-service attack by jamming until the network hibernates and then turning off the jam signal. Mandating a maximum duty cycle using a fixed rate-limiting mechanism might also be used to mitigate the affect of jamming attacks aimed at sapping energy, however, with the same limitations as outlined above.

5.2 Caisson Design Overview

This section provides an overview of the Caisson suite of denial-of-sleep mitigation mechanisms. These mechanisms take advantage of clustering in a lightweight intrusion detection system that monitors for the types of network activity that signals a denial-of-sleep attack. Caisson has many advantages over the intrusion detection mechanisms introduced in Section 2.5.5. It is primarily rules-based. As such, it does not require a lengthy training period like some IDS agents, nor does it require promiscuous monitoring of the network or the storage of large amounts of previous network traffic. Furthermore, Caisson is completely distributed, so each node independently identifies potential denial-of-sleep attacks. This is important in a wireless network where messages to and from a centralized intrusion detection monitor could be jammed, spoofed, or replayed. Finally, instead of reporting malicious activity to a base station so that it might be stopped through human intervention, the individual Caisson mechanisms take local action to mitigate the affects of malicious traffic.

The Caisson mechanism runs primarily at the link layer of the TinyOS protocol stack, although in a clustered network it must also interface with the clustering protocol at the routing layer. Caisson also uses physical layer information about transceiver state to identify potential denial-of-sleep attacks. It does not impose any clustering on the network beyond that which is provided by the network's clustering protocol or the virtual clusters of one-hop neighborhoods of nodes.

The primary goal of these tools is to prevent the energy draining effects of denial-of-sleep attacks so that network lifetime can be preserved. During some denial-of-sleep attacks, it is also possible to maintain a reasonable level of network throughput. In these situations, a secondary goal is to preserve throughput.

Caisson includes three modules, each addressing a different denial-of-sleep vulnerability. Functionality is split between the modules because various MAC protocols have different vulnerabilities and, therefore, are susceptible to different denial-of-sleep attacks. To conserve resources on WSN devices, only those modules necessary to protect nodes against known MAC protocol vulnerabilities are included when WSN applications are deployed. Figure 5.1 shows the Caisson suite

as a cross-layer module integrated into the routing, link, and physical layers of the USC/ISI sensor platform layered network model [4]. The components are described in the following paragraphs.

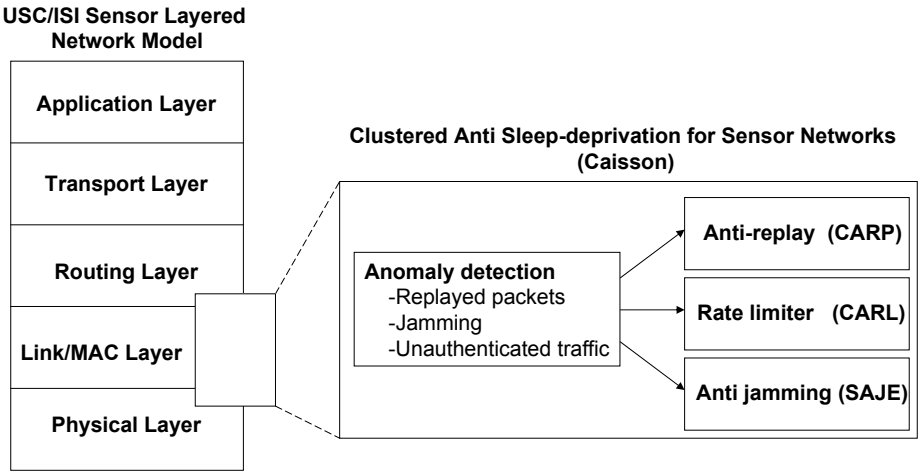


Figure 5.1: The Caisson mechanism as it is integrated into the ISI/USC sensor network stack. The mechanisms primarily interface with the MAC layer.

5.2.1 Clustered Anti-Replay Protection (CARP)

Most anti-replay mechanisms in the current literature require each network node to maintain a neighbor table that contains anti-replay information, normally in the form of an incrementing packet counter, for each node from which it might receive traffic (for example, [41, 45, 46]). On traditional computing platforms, this few bytes of information per host occupies only a tiny fraction of the devices' memories and creates no significant overhead concerns. On wireless sensor platforms, which typically have only a few kilobytes to hold program data, this table can present a significant resource challenge. For large scale WSN deployments, it not feasible for each node to maintain replay information for all nodes in the network. Since large-scale WSN deployments generally use some form of clustering to deconflict WSN traffic and aggregate data, a more efficient mechanism is for nodes to only maintain anti-replay information for the nodes in their cluster. Clusterhead nodes must also maintain anti-replay data for the other clusterhead nodes so that data can be forwarded between clusters. Many existing clustering protocols allow the network engineer to dictate the maximum cluster degree and the maximum number of clusters [34, 35]. With this

information, it is possible to set an upper bound on the size of the anti-replay table at compile time.

The difficulty with this technique is that networks re-cluster periodically to distribute the extra energy consumption of clusterheads among nodes in the network. Therefore, anti-replay counters must be securely distributed during the clustering process. Chapter 6 describes how CARP accomplishes this in a secure, low overhead manner. This technique, coupled with the mixed-sequencing protocol for anti-replay counters described in Section 2.5.3, provides low-overhead anti-replay support in large-scale WSNs.

5.2.2 Clustered Adaptive Rate Limiter (CARL)

As discussed in Section 4.1, many denial-of-sleep attacks take advantage of some WSN MAC protocols' policies to extend duty cycles when there may be more traffic in the network, without regard to whether or not previous traffic is legitimate. Strict rate-limiting should be avoided because of the potential for preventing legitimate network traffic. When there is a high rate of unauthenticated or replayed traffic, however, rate-limiting is justified because this traffic can have a profound negative impact on network lifetime.

The CARL mechanism uses lightweight network intrusion detection to classify traffic as either legitimate or potentially malicious based on whether or not it passes authentication and anti-replay checks. Establishing the intent of a node transmitting multiple unauthenticated or replayed packets is not always straightforward; such activity might be a malicious host attempting to subvert the network, or it might be a simple node malfunction. Regardless of the reason, if the traffic has the effect of reducing potential sleep time in a cluster, rate-limiting will reduce the negative impact on network lifetime. A malicious traffic threshold is used to dictate the level of unauthenticated or replayed traffic required before rate-limiting is triggered. Another threshold is used to increase the level of rate-limiting. When malicious traffic rates meet defined thresholds, nodes independently trigger rate-limiting to preserve energy. These thresholds provide the potential for traffic to be received during attacks that allow legitimate traffic in the network. If the network is not able to send any legitimate traffic because of the high rate of unauthenticated or replayed traffic, nodes increase their rate-limiting to the maximum level to preserve maximum node energy. Another threshold is

used to decrease rate-limiting levels when there is a reduction in the amount of malicious traffic in the network.

This mechanism adaptively reduces the duty cycle of cluster nodes depending on the rate of malicious traffic in the network, thereby maintaining energy reserves in the face of this malicious traffic while not limiting legitimate traffic during periods of higher than normal volume. Chapter 7 describes this technique and provides results of experiments to measure the extent to which it preserves network lifetime and throughput.

5.2.3 Sensor Anti-jamming Engine (SAJE)

As described in Section 2.5.4, mechanisms exist to reliably identify various forms of jamming in WSN. Only T-MAC and B-MAC are susceptible to jamming-based denial-of-sleep attacks. However, since the intent of jamming is to prevent traffic flow in a WSN, any energy consumed by a node when the network is being jammed is wasted. By transitioning to a low-power hibernation mode when a jamming attack is ongoing, nodes can preserve considerable energy and greatly extend network lifetime. Chapter 8 describes the implementation of a jam detection mechanism that reliably identifies ongoing jamming attacks with a very low false-positive rate and uses a technique called *random hibernation* to preserve node energy in the face of such attacks.

5.2.4 Recommended Caisson Components by Protocol

Table 5.1 lists the Caisson components that should be included in WSN deployments using each of the MAC protocols studied here to defend against denial-of-sleep attacks based on their specific vulnerabilities.

All protocols require anti-replay protection (CARP). S-MAC and G-MAC are susceptible to replay-based denial-of-sleep attacks. T-MAC and B-MAC require anti-replay support as part of the rate-limiting mechanism to identify replayed traffic that might be part of a denial-of-sleep attack. Rate-limiting (CARL) is required by both T-MAC and B-MAC due to their susceptibility to unauthenticated broadcast attacks. Anti-jamming support (SAJE) is required by T-MAC and

Table 5.1: Recommended Caisson Components by Protocol

Protocol	CARP	CARL	CAJE
S-MAC	Recommended	NA	Optional
T-MAC	Recommended	Recommended	Recommended
B-MAC	Recommended	Recommended	Recommended
G-MAC	Recommended	NA	Optional

B-MAC because they are both vulnerable to jamming-based denial-of-sleep attacks. Although S-MAC and G-MAC are not susceptible to jamming-based denial-of-sleep attacks, both protocols can benefit from jam detection by placing nodes in hibernation while being jammed. Since S-MAC and G-MAC nodes are not able to share traffic during an effective jamming attack, sleeping during these attacks decreases energy consumption while not affecting throughput.

5.3 Summary

This chapter has described the classes of denial-of-sleep defense required to prevent these attacks from severely affecting sensor network lifetimes. It also provides a high-level description of the Caisson suite of tools that are designed to preserve sensor network energy in the face of such attacks. The subsequent three chapters describe each Caisson component in detail and provide results of experiments used to evaluate each technique.

Chapter 6

Clustered Anti-Replay Protection for Wireless Sensor Networks

Two men say they're Jesus, one of them must be wrong.

- From the song "Industrial Disease" by Dire Straits

Replayed traffic can be used to subvert a sensor network in multiple ways. In Chapter 4, the effects of replay-based denial-of-sleep attacks are described. Recorded and replayed packets can also be used to make it appear that a device's sensors have been tripped when they have not. Similarly, traffic in one portion of the network can be replayed elsewhere to make it appear as if similar events have happened in multiple locations.

Providing replay protection in traditional networks is straightforward. Perhaps the most widely known anti-replay mechanism is part of IPSec [45], the security standard for the Internet Protocol (IP). IPSec, which provides anti-replay protection at the network layer, uses a per-host anti-replay counter that is incremented and transmitted with each packet. The receiving device compares the received counter value against a locally stored table of IP addresses and anti-replay counters to confirm that the received packet was not replayed. This technique is not well-suited for the severe resource constraints of WSN, primarily due to the memory overhead. A 200-entry anti-replay table takes 1200 bytes, or 30% of the memory on a Mica2 [21] sensor device, when using two-byte node addresses and four-byte anti-replay counters. Because the small amount of

available RAM is shared by the device's operating system, network protocols, security suite, and sensor applications, consuming such a large percentage of a node's memory for anti-replay is impractical.

This chapter describes *Clustered Anti-Replay Protection* (CARP), a mechanism for providing scalable anti-replay support in WSN. CARP takes advantage of node clustering to significantly reduce the amount of memory that must be reserved for anti-replay protection. It provides reliable anti-replay support while placing an upper limit on the number of anti-replay table entries required per node by exchanging a small amount of information during network clustering. It shows that the overhead of adding replay counters to network traffic is reasonably low across several state-of-the-art MAC protocols for WSNs using either the Mica2 or the Tmote Sky WSN platform. As shown in Section 6.2, the worst-case reduction in network lifetime for the Mica2 platform is 1.3% and for the Tmote Sky, 4.6%. Furthermore, CARP is tolerant of short-term network disruptions and packet loss, and is completely distributed. WSN nodes build anti-replay tables through information gathered during the clustering process and correctly make independent decisions on whether to accept a packet.

The rest of this chapter is organized as follows. Section 6.1 details the CARP mechanism and Section 6.2 examines overheads incurred by CARP across WSN platforms and MAC protocols. Section 6.3 describes the Mica2 implementation of the mixed sequencing anti-replay protocol, and Section 6.4 provides a chapter summary.

6.1 Clustered Anti-Replay Protection

Providing anti-replay support in small or sparse networks, where the number of one-hop neighbors is limited, is straightforward. Storage space for an anti-replay table is allocated and nodes simply maintain anti-replay counters for all of their one-hop neighbors. A secured query-response mechanism that allows all nodes to learn their one-hop neighbors at network startup is sufficient to populate the table. Providing reliable anti-replay protection in larger networks is more difficult. The maximum node degree must be known when nodes are programmed so enough memory can

be reserved for the maximum possible number of one-hop neighbors. Failure to reserve enough anti-replay table entries leads to potential abuse. If a node has more neighbors than replay table entries, it would have to either purge old entries for new ones, inviting replay attack, or ignore some subset of its neighbors, which is a form of denial-of-service. Furthermore, a maximum node-degree value that is based on a uniform distribution of nodes in a geographic area does not account for the dispersion patterns of air-dropped or other mechanically dispersed network deployments. The primary advantage of the CARP mechanism is that it places an upper limit on the number of anti-replay table entries that is required of any network node.

This section outlines the parameters under which CARP operates and then describes the specific functioning of the mechanism.

6.1.1 Network Model

CARP is designed to provide anti-replay protection in large, dense sensor network deployments. Since CARP assumes that the sensor network in which it will be deployed uses clustering for communication, it shares the following assumptions with most sensor network clustering mechanisms [33, 34].

- The network is made up of homogeneous, stationary nodes.
- Individual wireless links are symmetric. If node a can transmit to node b, then node b can also transmit to a at the same power. (This is required for clustering, but not for the correct functioning of the CARP mechanism. A node that is left out of a clustering round can safely rejoin the network during the next clustering event.)
- Nodes will be unattended during deployment and energy supplies will not be replenished.
- All inter-cluster traffic is routed through clusterheads, in other words, there is no direct communication between nodes of different clusters. This is a logical assumption in a clustered network.

In addition to the above, this mechanism, like any anti-replay scheme, requires that network traffic at least be authenticated so that an intruder cannot generate traffic that will be accepted as genuine and cannot modify replay counters without detection. Encrypted traffic is preferred for security reasons, but is not required, and is often not favored in sensor networks because of the added overhead.

The CARP mechanism does not require a specific clustering algorithm be used. It only requires that the clustering mechanism support specifying the maximum node degree of clusterheads and a maximum number of clusterheads, features supported by existing clustering techniques [34, 35]. To maintain generality, this mechanism does not assume that a specific MAC protocol is used during the network's steady-state periods.

6.1.2 Providing Scalable Anti-replay Protection

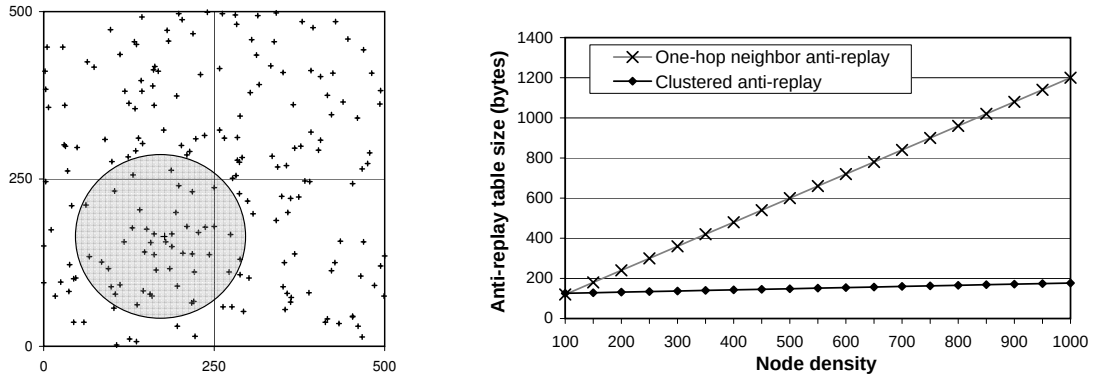
CARP provides anti-replay protection using the well-known sequential freshness technique of anti-replay counters described in Section 2.5.3. Each node sends an incrementing counter value in every packet that it transmits in the network. Nodes that receive those packets compare the counter value included in the packet to a locally maintained counter and, as long as the received value is greater than the stored value, the packet is accepted as "fresh." Since all inter-cluster traffic in a clustered network flows through clusterheads, a clusterhead's anti-replay table must contain entries for nodes within its cluster and for other clusterhead nodes. Each cluster member need only maintain anti-replay counters for the other nodes in its cluster and for its clusterhead since the clusterhead is the only source of legitimate traffic from outside the cluster. Assuming 2-byte node identification numbers (IDs) and 4-byte anti-replay counters, the size of any node's anti-replay table is bounded as follows,

$$Bytes_{replay.table} \leq (6\ bytes) \times (ND_{max} + NC_{max} - 1), \tag{6.1}$$

where ND_{max} is the maximum clusterhead node degree and NC_{max} is the maximum number of clusters, which is greater than $N_{network.nodes}/ND_{max}$. For example, in a one-hop network of 200 nodes that does not use CARP, the minimum replay table size is $(6\ bytes) \times (200) = 1200$ bytes,

or 30% of the memory on a Mica2 wireless sensor device. As the network scales to thousands of nodes, the size of the replay table could easily surpass the available RAM on a sensor device. In contrast, the amount of memory required using CARP is substantially lower. Consider a 200-node network in which $ND_{max} = 20$ and $NC_{max} = 10$. The replay table size is $(6\ bytes) \times (20 + 10 - 1) = 174$ bytes, or 4.4% of the available RAM on a Mica2.

Another example that illustrates the memory savings of this scheme, this time in a multi-hop network, is given in Fig. 6.1. Two hundred sensor nodes are shown in a uniform random distribution over a 500 $\times$ 500 meter area. The shaded area represents the 125-meter outdoor transmission range of a Tmote Sky sensor device [22], which covers slightly less than 20% of the deployment area. Nodes within the shaded area are one-hop neighbors of the node at the center of the circle. Fig. 2(b) shows how the number of bytes for the anti-replay table grows for the traditional one-hop neighbor approach versus CARP's clustering approach.



(a) Two hundred nodes uniformly distributed in a 500-by-500 meter area. The shaded are depicts the one-hop neighbors of the node in the center of the circle.

(b) Overhead of traditional versus clustering approach for anti-replay tables for the above network. The top line represents anti-replay table size for all one-hop neighbors. The bottom line represents size based on a clustered hierarchy. Both lines assume a uniform node distribution.

Figure 6.1: Comparison of minimum replay table sizes for a Tmote Sky in a 500-by-500 meter deployment as node density increases.[1]

This technique leverages network clustering to significantly reduce the size of each node's anti-replay table, but since the network periodically reclusters to distribute the clusterhead energy consumption amongst all nodes in the network, there must be a secure mechanism to distribute

[1]Copyright IEEE. Used through residual rights retained from [92].

replay counter values. This is complicated by the fact that cluster composition changes upon each reclustering event, so distribution is not a simple matter of sending a full table of replay counter values from the old clusterhead to the new one.

6.1.3 Secure Anti-replay Counter Distribution

The exchange of anti-replay counters must occur during the reclustering process. This section describes how this is accomplished in the context of the clustering algorithm used by LEACH [33], as this is a simple and well-known algorithm. It is important to note, however, that this mechanism is not tied to a particular clustering algorithm and can be easily integrated into any mechanism.

LEACH operates in *rounds*, which are broken up into set-up and steady-state phases [33]. As in most clustering mechanisms, the steady-state phase is generally much longer than the set-up phase to minimize clustering overhead. Nodes volunteer to become clusterheads by generating a random value between 0 and 1 and comparing that value to their threshold for becoming a clusterhead during that round. The threshold is based on a predetermined percentage of clusterheads and the number of times a node has been a clusterhead. The newly volunteered clusterheads then broadcast their status to the network. After receiving the clusterhead announcements, each non-clusterhead, or *member* node, chooses a clusterhead based on the signal strength of the received clusterhead announcements and sends a reply indicating membership in that cluster.

CARP's secure anti-replay counter distribution mechanism piggy-backs on this algorithm. For this discussion, the notation used in Section 2.5.3 to describe the mixed-sequencing protocol is used. Before sending out the clusterhead announcement, a newly volunteered clusterhead computes a message digest as follows,

$$m = MD_{sk}(id|ctr|round|msg), \tag{6.2}$$

where id is the node's address, ctr is the node's current anti-replay counter value, $round$ is the number of the clustering round, and msg is the cluster announcement message.

All nodes that receive a clusterhead announcement message authenticate the message and

confirm that the value of $round$ is higher than their own stored value of $round$. If so, they accept the clusterhead announcement as genuine. For the purposes of preventing replay during the clustering process, the value of $round$ is used as a global network replay counter, since clustering events are always network-wide events. New clusterheads populate their anti-replay tables with the anti-replay counter values in each of the clusterhead announcement messages that they receive. These replay counters for one-hop clusterheads will be used for inter-cluster communication.

Once non-clusterhead nodes decide with which clusterhead they will associate, they record their chosen clusterhead's replay counter value and construct a message similar to the one in (6.2), but with msg being a cluster join message rather than a clusterhead announcement message. By authenticating the message and confirming that the value of $round$ is correct, the clusterhead is sure that the join messages are not replayed from previous clustering rounds. The clusterhead populates its anti-replay table with the ctr values from the cluster-join message from each node that replies to its clusterhead announcement. If the clustering mechanism allows direct intra-cluster traffic between non-clusterheads, other nodes that overhear cluster-join messages for that clusterhead also populate their anti-replay tables with the sending nodes' addresses and replay counter values, again using the value of $round$ to verify that the messages are fresh.

After the clustering process is complete, all necessary anti-replay counter values are set for intra- and inter-cluster communication by inserting only a small amount of data into clusterhead announcement and cluster join messages. This process does not require any additional network traffic over what is needed for clustering, even in the simplest clustering algorithm. Many clustering algorithms rely on the exchange of several clustering messages, which would only reduce the percentage of overhead added by the above exchange.

6.2 CARP Overheads

This section examines the cost of CARP in terms of network lifetime overhead, or the percentage of reduced network lifetime, and memory overhead. Network lifetime is reduced because of the increased transmit and receive time resulting from the addition of anti-replay counters to each

packet. This analysis ignores the overhead of the replay counter exchange during clustering. Since the exchange of anti-replay counters adds such a small amount to clustering messages and the steady-state period is extremely long compared to the cluster setup period, the energy overhead added to clustering is assumed to be inconsequential.

6.2.1 Replay Counter Size

To provide proper protection against replayed data, the replay counter must be of sufficient size that it is not exhausted during a network's deployment. If sensor network nodes transmit one packet per second, a 16-bit counter would be exhausted in 18 hours, which is clearly an insufficient duration. Even at one packet per minute, a 16-bit counter would last only 45 days, which is far below the expected lifetime for most WSN deployments. At one packet per second, a 32-bit counter would wrap in 136 years. Even at 50 packets per second, a rate much higher than expected for sensor networks, a 32-bit counter would last for almost three years (994 days). For this research, 32-bit anti-replay counters are used, however, the counter size can be modified and reducing it can result in energy savings in networks with low traffic.

6.2.2 Overhead of Adding Anti-replay Counters to WSN Traffic

The overheads of adding anti-replay counters to traffic are examined across the WSN MAC protocols for the Mica2 and the Tmote Sky WSN platforms in this section. Overheads are presented in terms of percentage decrease in network lifetime. The overhead of adding replay counters to encrypted traffic is examined, as well as the overhead incurred when adding replay counters to traffic that is only authenticated.

Mica2 Platform Overheads

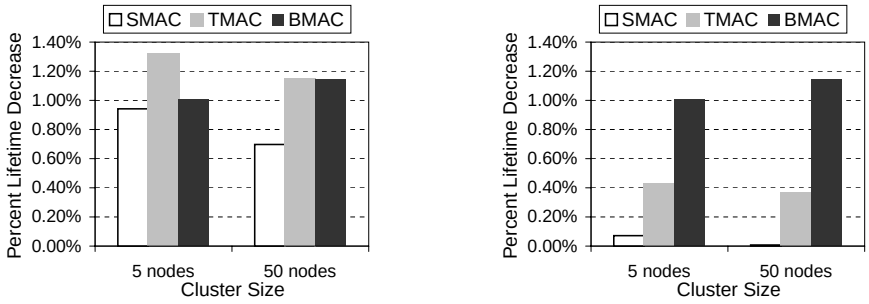
For the Mica2 platform, transmitted replay counter size is minimized using the mixed sequencing protocol [46] described in Section 2.5.3. A 1-byte explicit counter is used, which allows for up to 255 missed packets. The other three anti-replay counter bytes are implicit counters and are stored

by senders and receivers, but are not transmitted with outgoing packets. TinySec requires a 2-byte incrementing counter to be included with encrypted traffic to support its cypher-block chaining (CBC) mode encryption. When traffic is encrypted, this counter can be used for the explicit counter so there is no requirement to add additional data to encrypted traffic. With no additional data being transmitted, CARP does not add any additional energy expenditure to encrypted traffic on the Mica2 platform, so there is no overhead.

Despite the resulting reduction in data security, network designers might choose not to encrypt traffic, but only to provide authentication. Since without encryption TinySec does not add the 2-byte incrementing counter to all packets, a one-byte explicit counter must be added to provide anti-replay support. This is still much less than the 6 bytes per packet added by TinySec encryption. All protocols suffer slightly due to the extra transmitted data. These overheads are different for the various MAC protocols because of the different energy-saving techniques used by each protocol.

To calculate expected percentage decreases in network lifetime, MATLAB models were constructed based on the MAC protocol implementations of S-MAC, T-MAC, and B-MAC for the Mica2 platform. These models are the same models used in Chapter 4 to calculate network lifetimes during routine network traffic, modified to add replay counters to authenticated and encrypted traffic. Network and protocol parameters are the same as those listed in Table 4.2. To confirm the suitability of the mixed-sequencing anti-replay protocol and the energy consumption overheads of CARP, mixed-sequencing was integrated into the OPNET WSN MAC protocol models described in Section 3.5 and simulations were run to determine anti-replay overheads using the above parameters.

Fig. 6.2 depicts the overheads incurred by adding anti-replay counters to authenticated unicast and broadcast traffic on the Mica2 platform for both small (5-node) and large (50-node) clusters. T-MAC has the highest overhead, at 1.3% for unicast traffic. With its adaptive timeout, increasing the duration of transmit and receive time causes higher overheads, especially for unicast traffic, since each packet in the RTS-CTS-DATA-ACK sequence must be protected with a replay counter. S-MAC overheads are low for broadcast traffic because of the fixed duty cycle. Unicast



(a) Unicast traffic overhead.

(b) Broadcast traffic overhead.

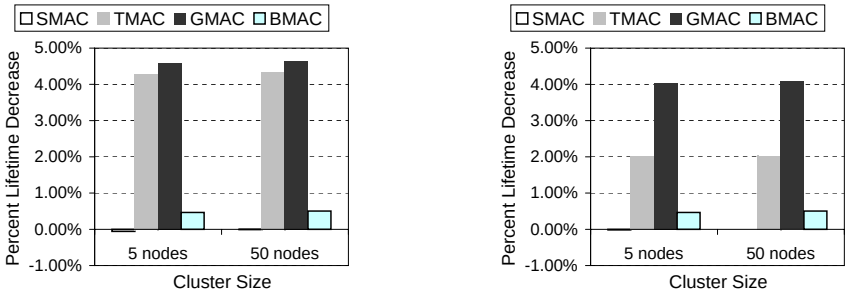
Figure 6.2: Overheads, expressed as decrease in network lifetime, incurred by WSN MAC protocol when anti-replay counters are added to authenticated traffic on CC1000-based WSN platforms.

exchanges extend further into the sender's and receivers' sleep periods, thus increasing overhead. Under B-MAC, two extra bytes of source address data must be added to the default packet format for anti-replay support. Despite this, overheads are low because the added data is overshadowed by very long preambles.

Tmote Sky Overheads

The MATLAB Tmote Sky models were also modified to include anti-replay counters. The Tmote Sky supports encryption and authentication as required by the IEEE 802.15.4 standard, which requires that each packet carry a 4-byte counter value as part of CTR-mode encryption. Unfortunately, the counter used to encrypt packets is not exposed to the application layer, so it cannot be used for this anti-replay mechanism. Therefore, packet counters must be added to both authenticated and encrypted network traffic. The mixed sequencing protocol is not used because the *message integrity code* (MIC) included with transmitted packets is computed at the physical layer right before a packet is sent and it is not possible to strip the implicit sequence number after the MIC has been computed. Therefore, a 4-byte counter must be added to each packet to provide anti-replay support. Figs. 6.3 and 6.4 depict the overheads incurred when anti-replay counters are added to authenticated and encrypted traffic.

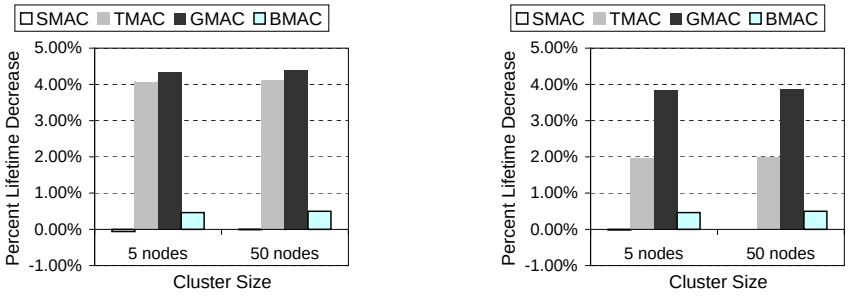
When anti-replay counters are added to S-MAC traffic, sensor network lifetime increases



(a) Unicast traffic overhead.

(b) Broadcast traffic overhead.

Figure 6.3: Overheads, expressed as decrease in network lifetime, incurred by WSN MAC protocol when anti-replay counters are added to authenticated traffic on CC2420-based WSN platforms.



(a) Unicast traffic overhead.

(b) Broadcast traffic overhead.

Figure 6.4: Overheads, expressed as decrease in network lifetime, incurred by WSN MAC protocol when anti-replay counters are added to encrypted traffic on CC2420-based WSN platforms.

slightly because the CC2420 requires less energy to send data than it does to receive it. With S-MAC's fixed duty cycle, the more time spent transmitting data in the network, the more energy is saved, so increasing the size of all data and control packets by four bytes results in a slight increase (maximum of 0.06%) in network lifetime. T-MAC incurs network lifetime overheads of up to 4.2% for unicast traffic and 2.0% for broadcast traffic. The overhead of adding anti-replay counters to authenticated traffic in G-MAC are up to 4.6% for unicast traffic and up to 4.1% for broadcast traffic. The slight decrease in available sleep time for nodes exchanging FRTS packets with the gateway, along with the increased GTIM size and increases in unicast and broadcast data, result in the relatively high overheads. B-MAC overheads are low and are the same for unicast

and broadcast traffic, with a maximum overhead of 0.5%. The 4-bytes of data added to network traffic are overshadowed by the long preambles required to ensure network nodes are awake under B-MAC's low-power-listening.

Anti-replay support was incorporated into the OPNET WSN MAC protocol process models for the Tmote Sky platform described in Section 3.5. Simulations confirmed the proper operation of the anti-replay mechanism and the overheads given above.

Discussion

The overhead of adding anti-replay information to WSN traffic is extremely low for the Mica2 platform, with no overhead for adding anti-replay to encrypted traffic and reducing network lifetime by a maximum of 1.3% when traffic is authenticated. Even for the Tmote, based on the CC2420 ZigBee-compliant radio, overheads are reasonable, ranging from no overhead to a maximum of 4.6% reduction in network lifetime.

6.3 Mixed-Sequencing Protocol Implementation

To confirm the effectiveness of the mixed-sequencing anti-replay protocol, it was implemented in TinyOS on the Mica2 WSN platform for the B-MAC, S-MAC, and T-MAC WSN MAC protocols. This implementation was also required for the proper operation of the CARL mechanism, which is described in the next chapter. For this implementation, the TinySec WSN security suite was used to provide packet authentication, which is critical to ensuring anti-replay counters are genuine.

6.3.1 B-MAC Implementation and Results

The TinySec developers provide a modified version of the B-MAC protocol that is TinySec-compatible. For ease of integration into future MAC protocol implementations, mixed-sequencing was integrated into the TinySec implementation provided with the latest release of TinyOS, version 1.1. By integrating mixed-sequencing into TinySec, any existing or future WSN MAC protocol that supports TinySec can also support anti-replay with minimum modification. The file

`${TOSROOT}/tos/lib/TinySec/TinySecM.nc`[2] was modified to support mixed-sequencing. First, a replay table was added to store node addresses and replay counter values. The two-byte node addresses used in this implementation allow for networks with up to 65,536 nodes. In clustered networks, these two bytes can also be broken into a one-byte cluster identification number and one-byte node address. Each replay table entry also includes a four-byte anti-replay counter value, as described in Section 6.2.1. The standard TinyOS packet format is defined in the header file `${TOSROOT}/tos/types/AM.h`. This file was modified to include the two-byte source node address and one-byte implicit counter that is transmitted with outgoing packets.

Adding replay counters to outgoing packets is straightforward. Recall that in the mixed-sequencing anti-replay protocol, only the low-order byte of the replay counter is transmitted to reduce packet overhead (see Section 2.5.3). For this implementation, nodes increment their replay counter values for outgoing packets in the TinySec send process. The replay counter's low-order byte (explicit counter) is then inserted into the appropriate packet header field and the three high-order bytes (the implicit counter) are placed into a buffer along with the TinyOS packet. A message authentication code is computed over the entire buffer and is appended to the end of the outgoing packet. The implicit counter is not transmitted with the packet.

The receive process is slightly more complicated. To minimize the impact of the authentication process on packet delay, TinySec computes the message authentication code incrementally as portions of the incoming packets are received. This optimization is maintained in the mixed-sequencing implementation. The receiving node inspects the sender's address and explicit counter value when a packet's header arrives and, then, updates its implicit counter for that sender, if necessary. The implicit counter is then placed into the receive buffer at the appropriate location before the message authentication code is computed over that portion of the buffer. Once the packet has been completely received and the full message authentication code has been calculated, it is compared to the authentication code received with the packet. If these authentication codes match, the packet is not only authenticated, but it is confirmed that the packet has not been previously received by this node.

[2]The environment variable `${TOSROOT}` defines the root directory of the TinyOS codebase. On most systems, this root directory is located at `/tos/tinyos_1.x/`.

This implementation was tested extensively in a one-hop neighborhood of Mica2 WSN devices running the B-MAC protocol. The mechanism properly filters unauthenticated and replayed TinyOS packets with no reduction in legitimate throughput. Adding anti-replay to TinySec via the above modifications and with a 20-entry anti-replay table increases the total amount of RAM and program memory used by 262 bytes, or less than 1% of the Mica2's total memory.

6.3.2 S-MAC and T-MAC Implementations and Results

TinySec-compatible versions of S-MAC and T-MAC have not yet been developed. The most efficient way to implement mixed-sequencing into these protocols was to integrate the mechanisms directly into the protocol implementations. The modifications were similar for S-MAC and T-MAC and the basic functioning of the protocols was not changed. All additions to the protocol code are distinguished using compiler directives (`#ifdef` blocks), so that applications can easily be compiled using the MAC protocols in their original form with minor modifications to the applications' Makefiles.

TinySec's message authentication library was used to provide message authentication for S-MAC and T-MAC. The files `${TOSROOT}/contrib/s-mac/tos/system/SMACMsg.h` and `${TOSROOT}/contrib/t-mac/tos/system/TMACMsg.h` define S-MAC and T-MAC packet formats. These formats were modified to add explicit counters and an anti-replay table was added to the MAC protocol code. When an outgoing packet arrives at the MAC layer for transmission, the full anti-replay counter is incremented and the explicit counter is inserted into the packet header. The implicit counter is placed in the send buffer with the outgoing packet and a message authentication code is computed and appended to the packet. At the receiving end, the implicit counter is computed according to the mixed-sequencing protocol and placed in the receive buffer with the received packet (which has been stripped of the authentication code). The message authentication code is then computed over the buffer and, if it matches the one in the received packet, the packet is accepted. Otherwise it is dropped.

These implementations were tested in a one-hop neighborhood of Mica2s. As with the B-MAC implementation, these implementations properly filter replayed packets with no reduction

in legitimate throughput. Table 6.1 shows the increase in Mica2 memory used, both in raw bytes and in percent of overall available program memory and RAM, when packet authentication and anti-replay support are added to a TinyOS application for each MAC protocol. The values in this table are the result of compiling a simple application in each security configuration: no security, with TinySec, and with anti-replay. Twenty anti-replay table entries were used in each of these implementations. Additional anti-replay table entries increase the size of applications with anti-replay support by six bytes per additional entry. For these memory overhead calculations, the nature and size of the application itself is unimportant. The increase in memory consumed by adding each security enhancement are the same regardless of the application size. For S-MAC, T-MAC, and B-MAC, the addition of anti-replay support consumes less than 1% of total Mica2 memory.

Table 6.1: TinySec and Mixed-sequencing Protocol Memory Overhead

	Application with TinySec authentication		Application with anti-replay	
	Additional memory used (over unsecured application)	Percent add'l Mica2 memory used	Additional memory used (over application with TinySec)	Percent add'l Mica2 memory used
S-MAC	2,382 bytes	1.8%	1,010 bytes	0.7%
T-MAC	2,568 bytes	1.9%	791 bytes	0.6%
B-MAC	9,325 bytes	6.9%	262 bytes	0.2%

6.4 Summary

This chapter described clustered anti-replay protection, a lightweight mechanism for providing anti-replay protection in WSN. CARP takes advantage of clustering to place an upper limit on the amount of memory required to store anti-replay counters in large-scale sensor network deployments. Furthermore, this mechanism is completely distributed, with each node populating its anti-replay table with information gathered during the clustering process (or designated at compile time for unclustered networks).

CARP uses the mixed-sequencing protocol to reduce the energy consumption overhead of adding replay counters to network traffic. Analytical and simulation results show that the worst-case network lifetime overhead of CARP is 1.3% for the Mica2 and 4.6% for the Tmote Sky. Mixed sequencing was implemented and tested on a Mica2 testbed to validate its effectiveness. The memory overhead of adding replay protection ranges from 0.2% of total available Mica2 memory for B-MAC applications to 0.6% of Mica2 memory for T-MAC applications. In terms of overhead caused by Caisson components, CARP has the most significant impact on network lifetime. One of the design goals identified in Chapter 3 (Table 3.1) was for denial-of-sleep defense mechanisms to reduce network lifetime by 5% or less when the network is not under attack. Network lifetime overheads meet this criteria in all cases. Another goal in Table 3.1 was to limit memory overhead to 5% or less of the total memory available per node. Again, this goal is achieved for all protocols.

The following chapter describes the simulation and implementation of the adaptive rate-limiting mechanism (CARL) developed as part of this research and provides associated results. CARL relies upon the anti-replay protection provided by CARP to identify and react to potential denial-of-sleep attacks.

Chapter 7

Adaptive Rate Limiting for Wireless Sensor Networks

Opportunities multiply as they are seized.

- Sun Tzu, *The Art of War*

Rate limiting protects networks against broadcast-based denial-of-sleep attacks by setting limits on the amount of time that nodes' radios are active. Strict rate limiting, however, can constitute a self-imposed denial-of-service attack by restricting legitimate network traffic. Many proposed military and homeland security WSN applications, such as vehicle tracking and gas plume monitoring, are characterized by bursty traffic. If these bursts of traffic exceed the rate at which traffic is allowed by the rate-limiting policy, they are improperly filtered. Furthermore, if the rate limiting mechanism does not take into account routine traffic rates, it can potentially allow attack traffic to consume energy resources much faster than routine traffic.

In traditional rate limiting, limits are placed on the amount of data that a node can transmit to reduce network contention or to provide quality-of-service guarantees. In the case of a denial-of-sleep attack, the victim network cannot limit the amount of traffic that is sent by a malicious host. The CARL mechanism, therefore, limits the amount of time that the radios of WSN nodes can be active during periods of high rates of malicious traffic. By reducing the amount of data that can be received and keeping the radio in power-down sleep mode the rest of the time, the impact

of denial-of-sleep attacks on network lifetime is significantly reduced.

The rest of this chapter is organized as follows. Section 7.1 introduces the adaptive rate-limiting mechanism and describes how it is applied to the T-MAC and B-MAC protocols. Section 7.2 provides results of simulation experiments to determine the effectiveness of this technique. Section 7.3 describes the implementation of the CARL mechanism for the Mica2 WSN platform and associated results. Section 7.4 provides a chapter summary.

7.1 Adaptive Rate Limiting

By strictly limiting the amount of time that nodes can be awake, denial-of-sleep attacks that target vulnerabilities in energy-saving MAC protocols can be mitigated. For example, by enforcing a maximum duty cycle on T-MAC or B-MAC nodes, these types of denial-of-sleep attacks cannot be successful. However, this technique increases latency in multi-hop networks and, if bursts of network traffic are generated at a higher rate than is supported by the rate-limiting policy, network traffic is lost. This could be disasterous if a WSN is deployed to provide, for example, real-time tracking of enemy movements.

A better alternative is *adaptive* rate limiting, whereby network traffic is restricted only when malicious packets have been sensed at a rate sufficient to suspect that the network is under attack. Figure 7.1 shows the correlation between received traffic and power consumption in a simulated Mica2 network. The offered load averages one packet-per-second (pps), with a burst of five pps from 120 seconds to 240 seconds. If this burst of packets is legitimate traffic, it should be allowed, despite increased power consumption during the burst. If the burst of traffic is malicious, the receiving node should take action to mitigate its energy-draining effects. This requires that the receiver be able to differentiate legitimate traffic from malicious traffic.

The rate of malicious traffic is calculated by maintaining information on recent packets. Legitimate packets are those that are properly authenticated and are not replayed. Other traffic is designated as potentially malicious. It is possible that some of the packets labeled potentially malicious are the result of bit errors during transmission or reception. It is important, therefore, that

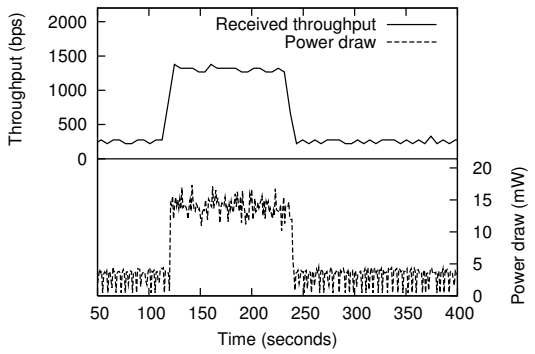


Figure 7.1: Received throughput and receiver per-second power consumption for a WSN node during a burst of network traffic.[1]

thresholds are set such that low rates of *false positive* detection of malicious packets do not trigger rate limiting. High rates of unintentional bit errors that have a negative impact on network lifetime may still justify rate limiting. The CARL mechanism functions as a distributed IDS, detecting malicious traffic at each host and taking action to reduce the adverse affects on network lifetime imposed by these attacks.

7.1.1 CARL Algorithm

Algorithm 1 describes the basic functioning of the CARL mechanism. The various threshold values (TH) are set by the network designer to control rate-limiting behavior. To minimize storage requirements, previous packets are not stored for analysis like they are in a traditional IDS. Instead, a bit array is used to track packet history. If a packet passes the authentication and anti-replay check, a "0" is shifted into the bit array. A "1" is shifted in for other packets. Old packets "age out" of the bit array and a simple bit count is used to compare the number of invalid packets against thresholds. High-order bits can also be masked to analyze the most recent packets.

Data authentication is achieved using TinySec [40], described in Section 2.5.2, which incurs a 1-byte per packet overhead. Anti-replay is provided using the *mixed-sequencing* protocol [46], described in Section 2.5.3, which incurs another 1-byte per packet overhead.

The potential negative impact of an increase in traffic on network lifetime is determined by

[1]Copyright IEEE. Used through residual rights retained from [93].

Algorithm 1 Adaptive Rate Limiting

```
1: Channel polled.
2: if (packet received) AND (failed authentication OR failed replay check) then
3:    BitArray_pkt ← (BitArray_pkt << 1) + 1 // Shift 1 into bit array.
4:    Drop packet.
5: else if (packet received) AND (passed authentication and replay check) then
6:    BitArray_pkt ← BitArray_pkt << 1 // Shift 0 into bit array.
7:    if (Pkt_dest_addr = Addr_this_node) OR (Pkt_dest_addr = Addr_broadcast) then
8:       Forward packet to higher layer.
9:    else // Pkt_dest_addr ≠ Addr_this_node
10:      Drop packet.
11:   end if
12: else if (no packet received) AND (rateLimitLevel ≠ 0) then
13:   BitArray_pkt ← BitArray_pkt << 1 // Shift 0 into bit array.
14: end if
15: CTR_bad_pkt ← count_ones(BitArray_pkt)
16: if rateLimitLevel = 0 then
17:   if (CTR_bad_pkt > TH_initiate_RL) AND (R_sleep < TH_sleep) then
18:      Initiate rate limiting.
19:   end if
20: else // In rate limiting mode.
21:   if (CTR_bad_pkt > TH_increase_RL) AND (R_sleep < TH_sleep) then
22:      if rateLimitLevel < rateLimitLevel_max then
23:         rateLimitLevel ← rateLimitLevel + 1
24:      end if
25:   else if (CTR_bad_pkt < TH_decrease_RL) OR (R_sleep > TH_sleep) then
26:      rateLimitLevel ← rateLimitLevel − 1
27:   end if
28: end if
```

monitoring the state of the transceiver to track active and sleep time. The CARL mechanism keeps track of each transition to and from the sleep state. R_{sleep} is the ratio of sleep time to total time and is maintained by periodically computing a weighted average of the percentage of time that the node has spent in sleep mode during the previous period as follows:

$$R_{sleep} = (0.75 * R_{sleep}) + (0.25 * (Time_{sleep}/Time_{period})). \tag{7.1}$$

For the experiments described in this chapter, R_{sleep} is updated every two seconds ($Time_{period} =$

$2\ seconds$). Since transceiver power consumption levels in the sleep and wake states are known, this serves to provide a good estimate of recent energy consumption. If a node observes high rates of malicious packets, but the value of R_{sleep} is above the set threshold, TH_{sleep}, it does not activate rate limiting. The malicious traffic is simply dropped.

7.1.2 Rate Limiting in T-MAC

Rate limiting is achieved in T-MAC by limiting the number of times a node's TTS can be reset to the TA value during a frame. Recall from Section 2.3.3 that in T-MAC, a node's TTS is reset after any packet is transmitted or received, so a constant stream of packets will repeatedly reset the TTS and keep nodes awake permanently. The longest that a Mica2 node can be awake per TTS reset under the network model used for these experiments is 54 ms. For testing, a maximum node duty cycle of one-third was chosen, so the maximum number of TTS resets is 7 (for a potential duty cycle of (8 $\times$ 54 ms)/1300 ms = 33.2%), which is referred to as rate-limiting level 1. With each increase in rate-limiting level, the number of allowed TTS resets is decreased by one, up to rate-limiting level 8, which allows no TTS resets. Since nodes increase their rate-limiting levels independently based on levels of good and bad traffic, this technique is completely distributed.

7.1.3 Rate Limiting in B-MAC

In B-MAC, nodes do not synchronize schedules. Rate limiting is, therefore, a matter of extending the check interval. This is done, however, without extending the duration of transmitted preambles. If preambles were to be extended along with the check interval, an attacker could simply extend its preambles and achieve the same negative impact on network lifetime as before.

If a node receives a preamble that is longer than the network's set preamble duration, the node can safely transition to sleep mode because the incoming packet is either malicious or a transmission error. By awakening only half as many times, the node's awake percentage, even if it receives a packet at every poll, is reduced to 36% (this is calculated by modifying the denominator in (4.3) to account for the increased check interval). By again doubling the check interval, the

expected awake percentage if a packet is received at every poll is 18%.

The primary shortcoming in the above technique is that the period during which nodes are awake is not synchronized, so if a node has a packet to send, there is no guarantee that other nodes will poll at the proper time to overhear a portion of the preamble and remain awake for the data packet. As check intervals increase, it is more likely that the other nodes will not be awake at the proper time. However, if all nodes increase their check intervals at the same time, a node knows that the poll times of the other nodes in the cluster are within one original check interval before or after its poll time (T_{poll}). By delaying its transmission to $T_{poll} - CheckInterval_{original}$ and transmitting two back-to-back packets, all nodes in the network will have an opportunity to sense the preamble and receive either the first or second packet. If there were no attacker present, network throughput would remain the same as if there were no change to the polling interval as long as the increased check interval is not longer than the network's offered load. Throughput suffers, however, in the face of a broadcast attack. Since the attacker is likely to be transmitting when a node's backed-off transmit time arrives, a node wishing to send traffic has to wait until the malicious transmission is complete. Nodes that awaken at the beginning of the transmit window will, therefore, receive the attacker's packet instead of the legitimate packet. Once the attacker's packet is complete, any legitimate node that wishes to transmit contends for the medium and if it wins the contention race, sends two back-to-back packets. Nodes that lose contention defer transmission until the next transmit window and go to sleep after receiving the incoming packet. Because some nodes receive the malicious packet instead of the legitimate one, overall network throughput drops, as will be seen in Section 7.2.

Even if they are in the same one-hop neighborhood, all nodes may not enter rate-limiting mode at the same time. CARL nodes, therefore, do not immediately increase their rate-limiting level upon meeting the required thresholds. If a node has a packet to transmit, it attempts to transmit that packet during the next transmit window for the current check interval and, if successful, increases its rate-limiting level. Other nodes wait until they receive a legitimate packet and then increase their rate limiting levels. Using this technique, nodes in a one-hop neighborhood are drawn into a *virtual cluster* in which all nodes check for traffic during the same window. To account for

the case of a non-subtle attacker, a timeout forces nodes to increase check intervals independently if they are unable to send or receive legitimate traffic. This way, network lifetime is still preserved when nodes are unable to send traffic due to an effectively jammed channel.

7.2 Simulation Results

7.2.1 Network Model

To examine the potential benefits and tradeoffs of the CARL mechanism, it was incorporated into the OPNET T-MAC and B-MAC protocol models described in Section 3.5 for both the Crossbow Mica2 and Tmote Sky WSN platforms. The impact of denial-of-sleep attacks and the effectiveness of the rate-limiting technique were examined on 20-node one-hop clusters of Mica2 and Tmote Sky sensor devices for both T-MAC and B-MAC. The network traffic model has all nodes transmitting at an equal rate such that the overall offered load meets the desired load to be tested. For the lower bit-rate Mica2 platform, the mechanism was tested at network offered loads of 1 pps, 0.5 pps, 0.33 pps, and 0.25 pps (or one packet every 1, 2, 3, and 4 seconds). On the Tmote Sky, simulated network offered loads of 1 pps, 2 pps, 3 pps, and 4 pps were used.

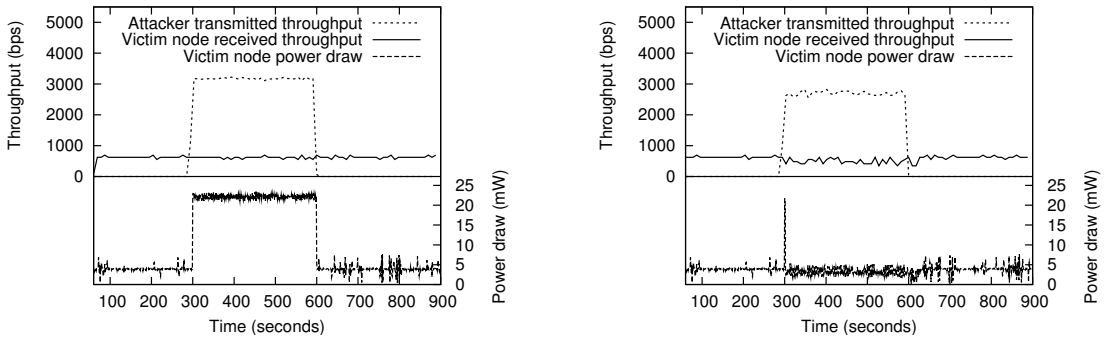
7.2.2 Threat Model

The threat is a broadcast attack in which a malicious node transmits a constant stream of back-to-back broadcast packets. The effects of these attacks on T-MAC and B-MAC for both the Mica2 and Tmote Sky platform are given in Section 4.3.2. To explore tradeoffs between network lifetime and throughput, a *subtle* attacker is assumed. As subtle attacker allows legitimate traffic in the network by pausing between packets by a period slightly longer than the network's contention window and by following the MAC protocol's CSMA policy for collision avoidance. Because legitimate traffic is allowed to be transmitted and forwarded as normal, the attack will likely not be detected until the devices' batteries are exhausted unless specific steps are taken to detect this type of threat. It should be noted that the rate-limiting mechanism effectively preserves device lifetime whether

the attack is subtle or obvious, however, throughput can only be preserved in the face of a subtle attacker.

7.2.3 CARL Effectiveness

To demonstrate how the CARL mechanism reacts to attack, Figure 7.2 shows an attacker's transmit level, a victim node's legitimate received throughput, and the victim's energy consumption in a simulated Mica2 network running B-MAC both with and without rate limiting. In both figures, the attacker's transmit throughput and the receiver's throughput are plotted against the left side y axis. Power consumption is plotted against the right side y axis. The malicious node begins a subtle broadcast attack at 300 seconds and stops the attack at 600 seconds. Without rate limiting, receiver throughput remains constant, but average energy consumption increases from 3.6 mW to 22.6 mW for the duration of the attack (Figure 7.2(a)). With rate limiting, the victim's per-second power consumption spikes as unauthenticated packets are received by the victim, then quickly returns to pre-attack levels as the victim reacts to the attack (Figure 7.2(b)). Throughput suffers somewhat during rate limiting, as discussed in the next section. When the attack ends, the victim node, along with the rest of the network, quickly returns to normal operation and throughput returns to pre-attack levels.



(a) No rate limiting.

(b) With rate limiting.

Figure 7.2: Per-second attacker throughput, victim node legitimate traffic throughput, and victim node power consumption during broadcast attack with and without rate limiting.[2]

[2]Copyright IEEE. Used through residual rights retained from [93].

Mica2 Results

To test the effectiveness of the adaptive rate limiting technique, Algorithm 1 was run with the parameters given in Table 7.1 under both T-MAC and B-MAC against a subtle broadcast attacker.

Table 7.1: CARL Parameters for Mica2 Simulations

Parameter	T-MAC value	B-MAC value	Description
$TH_{initiate\_RL}$	10	10	Threshold of malicious packets required to initiate rate limiting.
$TH_{increase\_RL}$	10	28	Threshold to increase rate limiting.
$TH_{decrease\_RL}$	7	3	Threshold to decrease rate limiting.
TH_{sleep}	0.90	0.90	Threshold of original sleep percentage required to not increase rate limiting.

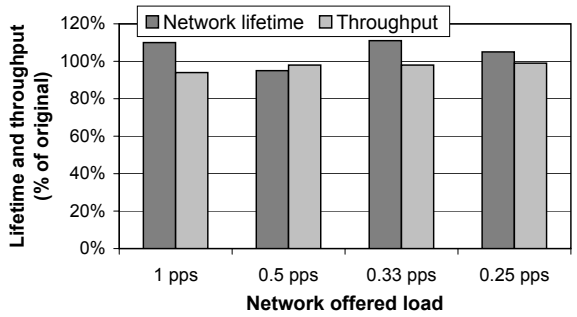
Note that in Algorithm 1, different thresholds are used to initiate rate limiting ($TH_{initiate\_RL}$), increase rate limiting ($TH_{increase\_RL}$), and decrease rate limiting ($TH_{decrease\_RL}$). Tests were conducted to determine thresholds that would be effective across offered loads and the thresholds in Table 7.1 were selected based on those tests. For these experiments, a 32-bit array was used to maintain packet history. A threshold of ten malicious packets was used to initiate rate limiting for both T-MAC and B-MAC. This threshold requires that almost one-third of the most recently received packets are malicious before rate limiting is initiated. T-MAC performs best at high rate-limiting levels. Section 7.2.5, shows how network lifetime and throughput are affected as rate-limiting levels increase for T-MAC and B-MAC. Those results show that in a T-MAC network, significant throughput degradation does not occur until the highest rate-limiting level, especially for low packet rates. A relatively low $TH_{increase\_RL}$ of 10 is used for T-MAC to allow rate limiting to increase to the highest levels for low offered-loads, while keeping the rate-limiting level slightly lower for higher offered loads to preserve throughput.

For B-MAC, throughput suffers dramatically if rate-limiting levels are increased such that the rate-limited poll interval is longer than the average time between legitimate packets. A high value or 28 for $TH_{increase\_RL}$ is, therefore, used to keep rate-limiting levels from increasing beyond

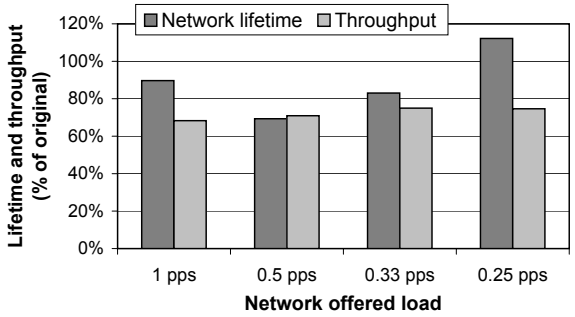
this point except under the highest rates of malicious traffic. A low value of $TH_{decrease\_RL}$ provides the best results for both protocols. Higher thresholds for decreasing the rate-limiting level causes the levels to fluctuate, resulting in an overall decrease in network lifetime. The low $TH_{decrease\_RL}$ values used in these experiments, 7 and 3 for T-MAC and B-MAC respectively, were shown to limit the amount of fluctuation in the rate-limiting level.

If rate limiting has been set to the maximum level below the network's interarrival rate but the ratio of bad packets to good packet is still high, throughput must be low, indicating an attack that is not allowing legitimate packets through the network. If the attacker is not allowing legitimate traffic, the logical response by WSN nodes is to increase the rate-limiting level so as to conserve energy. Once the attack is lifted and nodes receive only legitimate traffic or no traffic during polls, they will reduce their rate-limiting levels until they reach the original polling interval and the network will resume pre-attack operations.

Experiments were conducted with varying offered loads to determine how well the mechanism performs in a variety of traffic scenarios. Figure 7.3 depicts the percentage of network lifetime and throughput maintained across network offered loads using the automated rate-limiting mechanism for both T-MAC and B-MAC. For these experiments, the network was allowed to exchange data for three minutes, after which the subtle denial-of-sleep attack was initiated. The attack continues for the duration of the simulation. One of the design goals listed in in Section 3.1 was to maintain an average node lifetime of 90% or better while under attack as compared to a net-



(a) T-MAC rate limiting results.



(b) B-MAC rate limiting results.

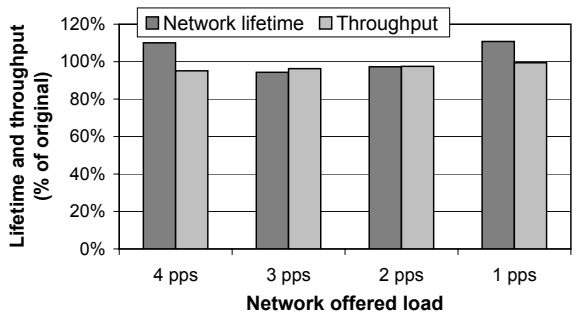
Figure 7.3: Percent of original network lifetime and throughput achieved using the automated CARL mechanism across offered loads on the Mica2 platform.

work that is not under attack. Under the conditions of the above experiments, this goal was easily achieved for T-MAC. This goal was not achieved for B-MAC for the 0.5 pps and 0.3 pps offered loads. In these cases, rate limiting achieves network lifetimes of 69% and 83% of the pre-attack lifetimes. It is important to note, however, that CARL parameters can be tuned to increase network lifetime or throughput based on expected packet rates in the deployed network and based on the priorities of the network designer. The thresholds used for these tests were chosen to preserve as much network throughput as possible, while still preserving network lifetime. These thresholds can easily be set to maximize network lifetime. Section 7.2.5, below, discusses the tradeoffs in network lifetime, throughput, and latency for the various rate-limiting levels in both T-MAC and B-MAC.

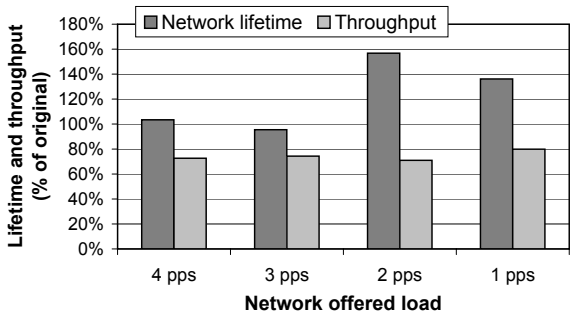
Tmote Sky Results

Figure 7.4 shows results of OPNET simulations using the Tmote Sky node model and the thresholds given in Table 7.2. Network lifetime and throughput results for the Tmote platform are consistent with those achieved using the Mica2.

The energy-saving benefit of CARL for the B-MAC protocol on the Tmote is much greater than that of the Mica2 platform, which can be observed by comparing Figure 7.3(b) and Figure 7.4(b). Mica2 network lifetime is up to 117% of pre-attack lifetime under CARL, while Tmote Sky lifetime is up to 156% of pre-attack lifetime. The higher cost of LPL polls on the Tmote



(a) T-MAC rate-limiting results.



(b) B-MAC rate-limiting results.

Figure 7.4: Percent of original network lifetime and throughput achieved using the automated CARL mechanism across offered loads on the Tmote Sky platform.

Table 7.2: CARL Parameters for Tmote Sky Simulations

Parameter	T-MAC value	B-MAC value	Description
$TH_{initiate\_RL}$	10	10	Threshold of malicious packets required to initiate rate limiting.
$TH_{increase\_RL}$	6	30	Threshold to increase rate limiting.
$TH_{decrease\_RL}$	4	4	Threshold to decrease rate limiting.
TH_{sleep}	0.90	0.90	Threshold of original sleep percentage required to not increase rate limiting.

accounts for this difference. Experimental data gathered by Polastre, et al. [8], and by Brownfield, et al. [69], were used to calculate the cost of a BMAC poll on the Mica2 and Tmote Sky, respectively. On the Mica2, with the CC1000 radio, the cost of a poll is 39.6 mW. On the Tmote, with the CC2420, the cost of a poll is 115.2 mW, almost a three-fold increase in poll cost. The Tmote platform, therefore, benefits more from the greatly reduced number of polls caused by rate limiting.

7.2.4 Statistical Accuracy

Section 3.4.2 discusses the method for determining the number of simulation replications required of each scenario to meet the desired statistical accuracy. In most cases, the desired accuracy, which was a 95% confidence that network lifetime results lie within ± 0.01 of the mean for the scenario under consideration, were met. This level of accuracy was not met in all cases, however. Tables 7.3 and 7.4 provide the size of the 95% confidence interval for network lifetime and throughput based on the series of simulations executed for each scenario, expressed as a percentage of the mean result for that scenario and performance metric. In the tables, columns labeled "No Atk" refer to scenarios in which routine network traffic is exchanged with no denial-of-sleep attack. Columns labeled "W Atk" are scenarios in which the network is subjected to a subtle unauthenticated broadcast denial-of-sleep attack. In all cases, the sizes of the confidence intervals were less than 3% of the mean result for the corresponding scenarios. The values above 1% are bolded in the tables.

Table 7.3: 95% Confidence Intervals for Mica2 Rate-Limiting Simulations

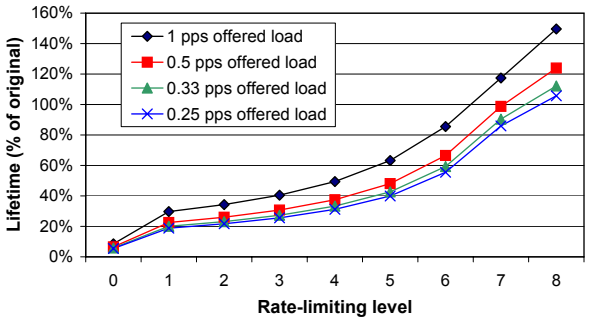
Lifetime								
	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk
T-MAC	0.06%	0.41%	0.04%	0.13%	0.04%	0.08%	0.04%	0.12%
B-MAC	0.23%	0.89%	0.14%	0.68%	0.88%	0.23%	0.02%	0.38%
Throughput								
	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk
T-MAC	0.11%	0.35%	0.11%	0.17%	0.14%	0.39%	0.15%	0.22%
B-MAC	0.03%	**1.15%**	0.01%	**1.25%**	0.04%	0.89%	0.02%	0.94%

Table 7.4: 95% Confidence Intervals for Tmote Sky Rate-Limiting Simulations

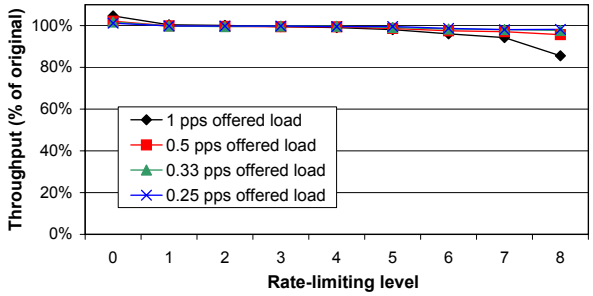
Lifetime								
	4 pps		3 pps		2 pps		1 pps	
	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk
T-MAC	0.09%	0.19%	0.09%	0.20%	0.08%	0.07%	0.07%	0.02%
B-MAC	0.49%	0.66%	0.28%	**1.74%**	0.16%	0.07%	0.10%	0.03%
Throughput								
	4 pps		3 pps		2 pps		1 pps	
	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk	No Atk	W Atk
T-MAC	0.27%	0.24%	0.32%	0.31%	0.41%	0.43%	0.57%	0.46%
B-MAC	0.07%	**2.98%**	0.04%	**2.54%**	0.02%	**2.04%**	0.03%	**1.56%**

7.2.5 Lifetime, Throughput, and Latency Tradeoffs

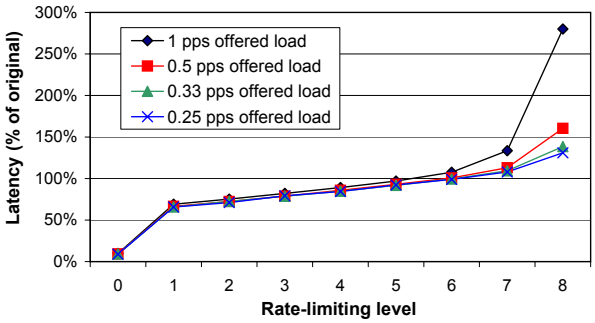
Inducing rate limiting on WSN nodes preserves network lifetime in the face of denial-of-sleep attack, however, these savings often come at the expense of lower network throughput and higher packet latency. Figure 7.5 shows the effect of increasing rate-limiting levels on network lifetime, network throughput, and average packet delay for the T-MAC protocol on the Mica2 platform. In these simulations, the maximum allowable rate-limiting levels were manually set to the desired level at the beginning of the simulation and the parameter values in Table 7.1 were used to initiate, increase, and decrease rate-limiting levels according to Algorithm 1. With no rate limiting in place (the far-left data points on the graphs in Figure 7.5), throughput is more than 100% of pre-attack throughput. This is because when nodes are under attack, they never transition to sleep mode, so



(a) Percent original network lifetime achieved.



(b) Percent original throughput achieved.



(c) Percent original delay experienced.

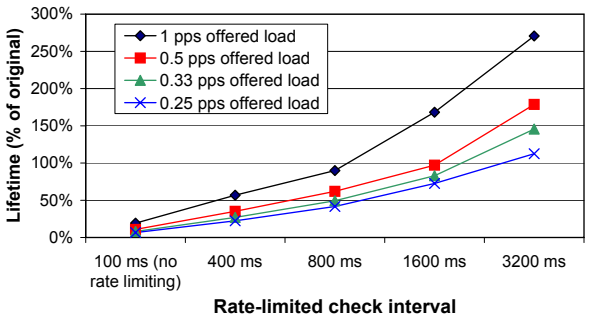
Figure 7.5: Percentage of original network lifetime, throughput,and delay on Mica2 platform as the T-MAC rate-liming level increases.

they do not have to wait until the active period at the beginning of the next frame to transmit a packet that arrives from the application layer. By not limiting the time during which packets can be transmitted, collisions are reduced, resulting in higher throughput. This also reduces network latency. Again, the attack keeps all nodes awake all of the time, so transmitters do not have to delay transmission until the beginning of the next frame. Network lifetime, however, is significantly degraded.

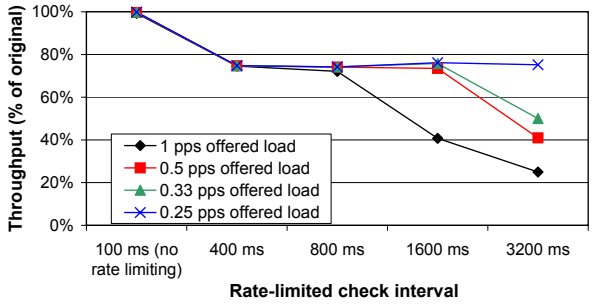
As the rate-limiting level is increased, network lifetime increases, and throughput eventually begins to drop. For the offered loads tested, throughput is not reduced below 99% until rate-limiting level five. Even at rate-limiting level eight, where no TTS resets are allowed, throughput remains at 84% or higher on the Mica2 platform. Packet latency also increases slowly as the rate-limiting level increases and latency does not reach "pre-attack" levels until rate-limiting levels five or six, depending on the network offered load. This is the same time that network lifetimes reach

100% of "pre-attack" levels. At lower rate-limiting levels, the network attack keeps nodes awake more than they normally would be, so average packet latency is lower than under non-attack conditions. It is clear from these results that as network offered loads increase, rate limiting improves network lifetime, but has a negative affect on both throughput and packet delay. This highlights the need to adjust rate-limiting levels dynamically based on legitimate packet rates. It is also important that the rate-limiting level be reduced quickly after an attack stops to minimize the amount of inappropriately filtered traffic.

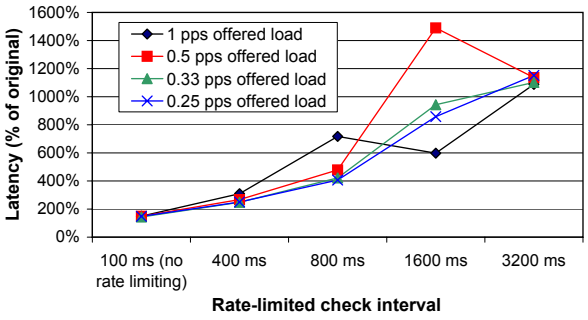
For rate limiting in B-MAC, the policy for increasing the rate limiting level is to double the check interval at each increase and halve the check interval at each decrease. Figure 7.6 shows the effect of increasing rate-limiting levels on network lifetime, throughput, and latency. In these simulations, check intervals are manually increased to the desired level at the beginning of the simulation.



(a) Percent original network lifetime achieved.



(b) Percent original throughput achieved.



(c) Percent original delay experienced.

Figure 7.6: Percentage of original network lifetime, throughput, and delay on the Mica2 platform as the rate-limited B-MAC check interval increases.

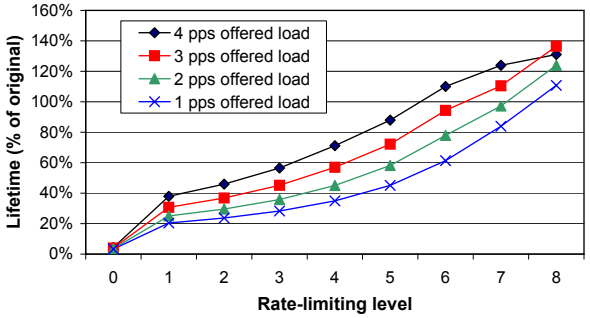
With no rate limiting in place (the far-left data points on the graphs in Figure 7.6), throughput remains 100%, but network lifetime is reduced to a fraction of the original for all offered loads. It should be noted from Figure 7.6 that throughput drops immediately to approximately 72% of the original network throughput for all offered loads when rate limiting is initiated. This is because for each packet sent, some percentage of nodes transition to sleep mode after receiving a malicious packet at the beginning of the check window. Once the check interval is increased beyond the network's offered load, throughput drops significantly. For example, at 1 pps, throughput is over 70% for check intervals of 400 ms and 800 ms, but drops to 40% when the check interval increases to 1,600 ms. If the network designer places a high priority on maintaining throughput, an 800-ms rate-limited check interval achieves this, while still maintaining over 90% of the original network lifetime. This is because there is a legitimate packet transmitted during almost every transmit window. The additional energy expended by transmitting each packet twice instead of once is largely made up for by the energy saved by polling the wireless channel every 800 ms instead of every 100 ms.

Packet latency increases dramatically when rate limiting is imposed on B-MAC networks, as indicated in Figure 7.6(c). This is due to the rate-limiting policy of doubling the check interval at each increase in rate-limiting level. Figure 7.6(c) seems to indicate that packet latency can decrease at high rate-limiting levels. This, however, is misleading because it does not reflect dropped packets. When the rate-limited check interval surpasses the network's average packet rate, packets are dropped in a first-in, first-out manner as nodes' transmit buffers overflow. Otherwise, over time, latency would grow without bound. As packets are dropped from the queue to allow newer packets to be transmitted, average packet latency does not grow as fast, and can actually decrease depending on the relationship between the network offered load and the rate-limited check interval. For example, at 1-pps offered load, the ovarall average time between legitimate packets transmitted in the network is 1000 ms. At this packet rate, Figure 7.6(c) shows latency increasing up to a rate-limited check interval of 800 ms. Recall that CARL only allows one legitimate packet to be sent per rate-limited poll. When the check interval is increased to 1600 ms, packets cannot be transmitted fast enough to satisfy the offered load of legitimate packets. Packets then begin to be dropped

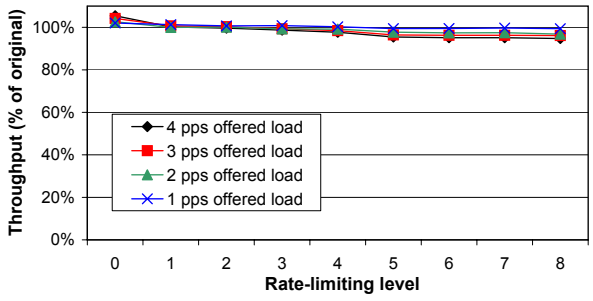
from packet queues and latency fluctuates. The same is true for the 0.5-pps offered load, for which the average time between packets is 2000 ms. Once the rate-limited check interval increases from 1600 ms to 3200 ms, which is longer than the average packet rate, latencies fluctuate.

The results given in Figure 7.6 indicate that check intervals that are significantly short of the network's offered load result in reduced network lifetime in the face of a broadcast attack with no throughput advantage. The B-MAC rate-limiting implementation, therefore, increases the check interval to the level closest to the network's offered load when rate limiting is initiated.

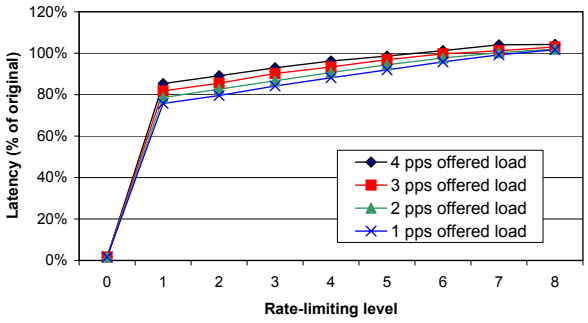
Figure 7.7 and Figure 7.8 show lifetime, throughput, and packet latency results for a similar set of experiments as those described above on the Tmote platform. These figures indicate that trends in each of these performance metrics are the same for the Tmote platform as they are for the Mica2.



(a) Percent original network lifetime achieved.

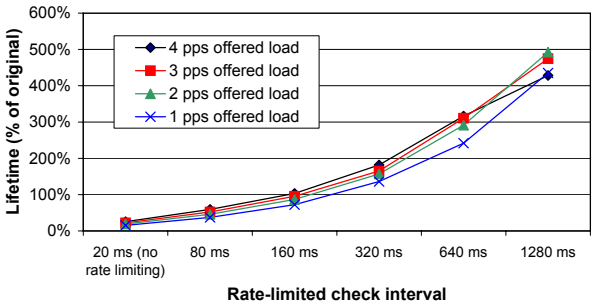


(b) Percent original throughput achieved.

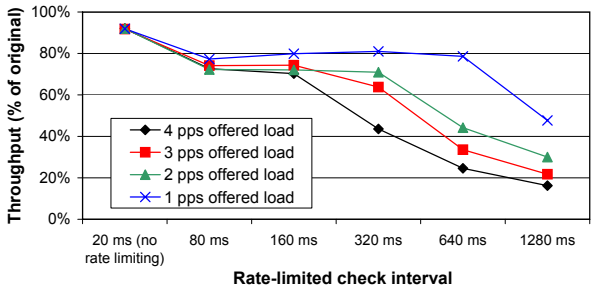


(c) Percent original delay experienced.

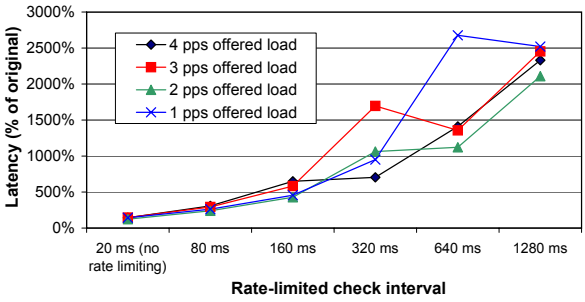
Figure 7.7: Percentage of original network lifetime, throughput, and delay on Tmote Sky platform as the T-MAC rate-limited check interval increases.



(a) Percent original network lifetime achieved.



(b) Percent original throughput achieved.



(c) Percent original delay experienced.

Figure 7.8: Percentage of original network lifetime, throughput, and delay on Tmote Sky platform as the B-MAC rate-limited check interval increases.

7.2.6 Implications of Packet Loss in CARL

In most cases, especially with the B-MAC protocol, imposing rate-limiting results in lost network traffic. This happens because nodes sometimes transition to sleep mode when the MAC protocol would otherwise have them awake to receive traffic from other network nodes. The extent to which packet loss impacts network functionality depends on the higher layer protocols and on the application running on the devices. In some cases, high rates of missed network-layer control packets could cause the network to become unstable or disconnected. An example is if a network attempts to recluster while in rate-limiting mode. If a high percentage of clusterhead volunteer messages or cluster join messages are missed, many nodes could be left out of the network. Some WSN applications, such as environmental monitoring or perimeter intrusion detection, are able to tolerate fairly high levels of packet loss. In many cases, even partial information from the network might be preferable to a network in which nodes quickly run out of energy due to denial-of-sleep

attacks. However, for applications that are not able to tolerate a high packet loss rate, the network may not serve the purpose for which it was deployed during rate-limiting.

In these cases, the benefits of rate-limiting must be carefully weighed against the potential negative impacts of lost traffic on the network. When using rate-limiting with the T-MAC protocol, reducing the maximum rate-limiting level still preserves significant energy while maintaining throughput levels that are close to pre-attack throughput. To avoid missed traffic during clustering, cross-layer techniques might be used to turn off rate-limiting during clustering and then restart it once the network has reclustered. At the application layer, redundant traffic or packet acknowledgments might be used to increase the likelihood of packet delivery. Another option is to have the rate-limiting mechanism detect potential attacks, but not take action to mitigate them. Instead, the rate-limiting mechanism could be modified to produce application-layer messages to notify a system administrator that a suspected attack is ongoing.

The previous paragraphs simply reported the extent to which traffic is lost in the face of rate-liming in various network configurations. Addressing the affect of packet loss in specific network scenarios is beyond the scope of this work and is not discussed here.

7.3 Rate Limiting Implementation

To test the effectiveness of the rate-limiting mechanism on a real WSN deployment, rate limiting was implemented for T-MAC and B-MAC on the Mica2 platform. For ease of portability between MAC protocols, rate limiting was developed as a TinyOS library module. The interface to the Caisson module is defined in the file `${TOSROOT}/tos/interfaces/Caisson.nc`[3]. This interface includes a set of *events*, that are signaled from the MAC protocol implementation, and a *command* that the Caisson module uses to indicate to the MAC protocol that a change to the rate-limiting level is required. The Caisson interface is given in Table 7.5.

The module code is located at `${TOSROOT}/tos/lib/Caisson/CaissonC.nc` and `CaissonM.nc`. By implementing rate limiting as a separate TinyOS module, most of the rate-

[3]The environment variable `${TOSROOT}` defines the root directory of the TinyOS codebase. On most systems, this root directory is located at `/tos/tinyos_1.x`.

Table 7.5: The Caisson Interface

Event name	Description
`signal_wakeup()`	Signaled by MAC layer when radio transitions from sleep mode to receive mode. This allows the Caisson module to track the percentage of time that the node spends in sleep mode.
`signal_sleep()`	Signaled when radio transitions to sleep mode.
`good_pkt_arrival()`	Signaled when a valid packet arrives from the physical layer. This triggers an update to the good/bad packet bit array.
`bad_pkt_arrival()`	Signaled when an invalid (unauthenticated or replayed) packet arrives from the physical layer, prompting an update to the bit array.
`empty_poll()`	Signaled when the radio awakens and returns to sleep mode without receiving a packet while rate limiting is activated. This is recorded in the bit array as a good packet.
`pkt_sent()`	Notifies Caisson module that a packet has been sent. This information is used by Caisson in tracking the network offered load.
`carl_level_updated()`	Notifies Caisson module that an update to the rate-limiting level, which is initiated by the `set_carl_level()` command (below) has been completed.

Command Name	Description
`set_carl_level(int level)`	This command notifies the MAC layer that the rate-limiting level should be set to *level*. This is determined using Algorithm 1 at every update to the bit array.

limiting code exists outside of the MAC protocols thus reducing the amount of required modification to the protocols. Algorithm 1 is completely contained in the Caisson module, using information passed to it via signals to the Caisson events listed in Table 7.5. The implementations of these events are also all contained in the Caisson module. For each event listed in the table, a one-line function call must be added to the MAC protocol code at the appropriate location to signal the event. The `set_carl_level()` command must be implemented in the MAC protocol based on the rate-limiting policy for each protocol. Just as with the anti-replay implementations, these modifications to T-MAC and B-MAC are all included using compiler directives so that applications can easily be compiled without rate limiting.

7.3.1 T-MAC Implementation and Results

After implementing the CARL mechanism in the Caisson TinyOS module, it was integrated into the current TinyOS version of T-MAC and tested on a network of five Mica2 WSN platforms. A test application on each node tracked sleep and throughput data and broadcast that data into the network such that the overall network offered load would average 1 pps. One of the nodes was connected to a computer with a serial cable and that node transmitted each packet to the computer as it was received so that overall network statistics could be collected.

A series of tests was conducted using this configuration with no attacker to collect baseline data on network lifetime and throughput. The CARL implementation was then tested against a node executing a subtle broadcast denial-of-sleep attack. For the network-attack tests, nodes were started and the network was allowed to stabilize for approximately two minutes before the attack was begun. When the attack was initiated, nodes quickly recognized the presence of the attack and over several seconds, increased to high rate-limiting levels. Based on the percentage of sleep time both with and without the attack, nodes were able to achieve sleep rates that, on average, would translate into approximately 98% of the pre-attack lifetime. This result is consistent with what would be expected based on the configuration of the test network and the simulation results described in Section 7.2.3. Throughputs, however, dropped to an average of less than 60% of the pre-attack level, which is significantly below the expected throughput under rate limiting in

T-MAC.

Based on data collected during these, and other, tests, it became apparent that this poor throughput was caused by the same physical-layer limitations in the Mica2 T-MAC implementation that were observed during the denial-of-sleep attack experiments conducted on T-MAC nodes, which are described in Section 4.4.2. To gather sleep and throughput data in an environment with better physical-layer properties, experiments were conducted using the Avrora [84] emulation environment, which allows PC-based emulation of the Mica2 WSN platform. In addition to the much improved physical layer, Avrora facilitates the collection of much more detailed sleep and throughput information. A series of experiments was conducted across network offered loads using Avrora. These experiments were executed on T-MAC using its default Mica2 configuration, which uses a 610 ms frame duration and an effective data rate of 19.2 kbps.

The percentages of original network lifetime and throughput achieved in the attack experiments are given in Figure 7.9. Due to the long running time of these emulations, these results are based on 20 replications of only 10 minutes of network time. The network lifetime results were as expected based on the OPNET simulation results. Throughputs, however, were 6% to 12% lower than expected depending on the network offered load and the rate-limiting level. This throughput reduction was caused by a much higher rate of packet collisions in the emulated network than in the OPNET simulations.

Collisions occur when a node that has a packet to transmit observes a clear channel during carrier-sense, but another node begins transmitting before the first node is able to switch its radio

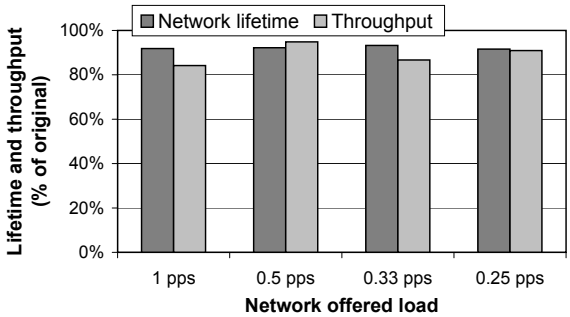


Figure 7.9: Percent of original network lifetime and throughput achieved using the automated CARL Mica2 implementation on a T-MAC network across offered loads.

from receive mode to transmit mode and begin transmitting. In the OPNET simulations, this delay is modeled as 450 μs, based on experimental results reported by Polastre, et al. [8]. In the Avrora emulations, there are cases of collisions resulting from packets that are sent as much as 3 ms apart. Since the packet durations for this experimental configuration are much longer than 3 ms, these packets collide. In the OPNET simulations, using a 450 μs receive-to-transmit delay, the packet loss rate due to collision is 1.5%. When the delay is increased to 3 ms, the loss rate due to collision increases to 11.5%. When an attacker begins transmitting a back-to-back stream of packets into the network, 3.5% of legitimate packets are lost to collisions with attack packets when the receive to transmit delay is 450 μs at the highest rate-limiting level. With a 3-ms delay, almost 19% of legitimate packets are lost to collisions at the maximum rate-limiting level. This increased rate of collisions in the Avrora emulation accounts for the decrease in preserved throughput for those experiments. The 3-ms transition is much higher than expected and is likely an artifact of the operation of the event queue in the Avrora emulator. The level of preserved throughput for rate limiting in TMAC on the Mica2 would likely lie somewhere between those observed in the Avrora emulations (Figure 7.9) and those observed in the OPNET simulations (Figure 7.3(a)).

Even under these less-than-ideal conditions, achieved throughput ranged from a minimum of 82% of unattacked network throughput to a maximum of 96% of unattacked network throughput depending on the offered load and the rate-limiting level. Network lifetimes were all 92% of pre-attack lifetimes or higher.

7.3.2 B-MAC Implementation and Results

Rate limiting was also integrated into the current version of the B-MAC protocol for the Mica2 platform. Again, a five-node network of Mica2s was used to test this implementation. As with T-MAC, network nodes were started and the network was allowed to stabilize over approximately two minutes, after which a subtle broadcast attack was initiated. A test application similar to the one used for the T-MAC tests collected sleep percentage and throughput data, which was compiled and reported by one of the devices via serial connection to a computer. When a broadcast-based denial-of-sleep attack was initiated, B-MAC nodes quickly recognized the attack and increased

rate-limiting levels accordingly. When the attack was stopped, nodes recognized the lack of malicious packet in the network and, over several seconds, resumed pre-attack sleep percentages and throughputs.

Figure 7.10 shows the results of these experiments in terms of percentage of original network lifetime and throughput preserved by nodes running the CARL mechanism. These results are based on ten replications of a network exchanging routine traffic with no attack at each offered load, and ten replications of the same network subjected to a subtle denial-of-sleep broadcast attack at each offered load. These results are consistent with those observed in the B-MAC simulations on the Mica2 depicted in Figure 7.3(b). The specific results per offered load are slightly different due to the different network model used in the implementation tests. The implementation tests use a baseline LPL polling interval of 135 ms, which is the same polling interval used in the denial-of-sleep attack tests discussed in Section 4.4.3. Throughput results for the 1-pps, 0.5-pps, and 0.25-pps experiments are lower than the throughputs achieved in simulation. This is because, in each case, the rate-limiting level selected by the CARL mechanism results in a polling interval that is slightly longer than the packet interarrival period. For example, at 0.25-pps, the network offered load is one packet every four seconds. The rate-limited polling interval, however, is $(32 \times 135\ ms)$ or $4.32\ ms$. As a result, the transmitted packet rate is 92.6% of the original (instead of 100%), which contributes to the throughput reduction.

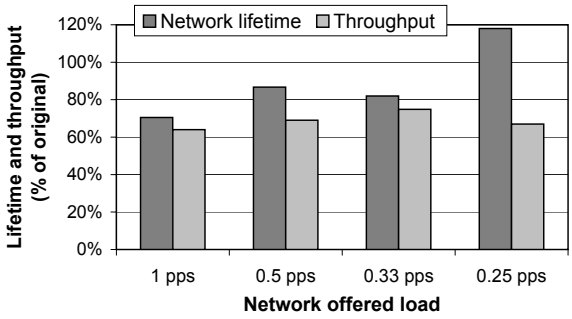


Figure 7.10: Percent of original network lifetime and throughput achieved using the automated CARL Mica2 implementation on a B-MAC network across offered loads.

7.3.3 Statistical Accuracy

To calculate the statistical accuracy of these experiments, 90% confidence windows were calculated based on a series of replications for each offered load under routine network traffic (No Atk) and while the networks were under attack (With Atk), using the technique discussed in Section 3.4.3. T-MAC results for both network lifetime and throughput are given in Tables 7.6 and 7.7. These results are based on 20 replications of the Avrora emulations discussed above. While these results do not approach the level of statistical accuracy of the simulation experiments, the confidence intervals are still reasonable. The size of the 90% confidence window for network lifetime ranges from 0.87% to 8.35% of the mean network lifetimes. Throughput confidence windows range from 2.55% to 4.56% of the mean.

Table 7.6: Statistical Accuracy of Mica2 T-MAC Network Lifetime Measurements (Network Lifetime Measured in Days)

	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No atk	With atk	No atk	With atk	No atk	With atk	No atk	With atk
Mean	49.86829	45.37523	50.83815	47.6424	52.5524	48.90003	53.64074	49.02543
Std dev.	4.749219	0.450276	2.982631	0.344775	2.557748	0.769373	2.645929	0.798782
90% conf.	47.78689	45.17789	49.53098	47.4913	51.43144	48.56285	52.48113	48.67536
window	51.94969	45.57256	52.14533	47.7935	53.67336	49.23722	54.80035	49.37551
Confidence window as % of mean	8.35%	0.87%	5.14%	0.63%	4.27%	1.38%	4.32%	1.43%

Table 7.7: Statistical Accuracy of Mica2 T-MAC Network Throughput Measurements (Throughput Measured in bps)

	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No atk	With atk	No atk	With atk	No atk	With atk	No atk	With atk
Mean	1656.597	1374.35	1165.401	738.7876	790.9404	665.6249	433.6391	394.3218
Std dev.	48.25308	70.9905	34.92171	38.46729	26.2298	31.61738	16.71218	19.8682
90% conf.	1635.45	1343.238	1150.096	721.9288	779.4449	651.7682	426.3148	385.6143
window	1677.745	1405.463	1180.706	755.6463	802.436	679.4816	440.9634	403.0292
Confidence window as % of mean	2.55%	4.53%	2.63%	4.56%	2.91%	4.16%	3.38%	4.42%

Network lifetime and throughput results for ten replications of the B-MAC experiments are given in Tables 7.8 and 7.9. The 90% confidence windows for network lifetime range from 2.05% to 6.64% of the mean lifetimes and throughput windows range from 2.19% to 11.69% of the mean values.

Table 7.8: Statistical Accuracy of Mica2 B-MAC Network Lifetime Measurements (Network Lifetime Measured in Days)

	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No atk	With atk	No atk	With atk	No atk	With atk	No atk	With atk
Mean	39.57342	32.09851	90.05644	96.73518	101.6863	115.8698	105.567	177.9669
Std dev.	1.599717	1.126738	1.734379	4.767054	1.678035	6.20569	2.256054	7.731754
90% conf. window	38.58192	31.40016	88.98148	93.78059	100.6462	112.0235	104.1687	173.1748
	40.56491	32.79686	91.1314	99.68978	102.7263	119.716	106.9653	182.759
Confidence window as % of mean	5.01%	4.35%	2.39%	6.11%	2.05%	6.64%	2.65%	5.39%

Table 7.9: Statistical Accuracy of Mica2 B-MAC Network Throughput Measurements (Throughput Measured in bps)

	1 pps		0.5 pps		0.33 pps		0.25 pps	
	No atk	With atk	No atk	With atk	No atk	With atk	No atk	With atk
Mean	1293.922	824.109	650.5333	448.0103	432.96	324.2295	428.5867	218.7275
Std dev.	28.39412	77.74257	18.76093	35.6957	7.644652	30.9757	13.44005	11.44987
90% conf. window	1276.323	775.9246	638.9054	425.8862	428.2219	305.0309	420.2566	211.6309
	1311.52	872.2935	662.1613	470.1343	437.6981	343.428	436.9167	225.8241
Confidence window as % of mean	2.72%	11.69%	3.57%	9.88%	2.19%	11.84%	3.89%	6.49%

7.3.4 Rate-limiting Implementation Overhead

CARL has no impact on transceiver transmit or receive time other than that resulting from the anti-replay mechanism, which is required to categorize replayed packets as malicious. The network lifetime overheads of adding replay counters to packets across WSN MAC protocols and platforms are given in Section 6.2.2.

The extra program code and data structures of the CARL implementation result in memory overhead. Table 7.10 shows the increase in Mica2 memory used when rate limiting is added to a test application. The overhead is higher for the B-MAC protocol because the mechanism by which rate limiting is achieved in B-MAC is more complicated than in T-MAC, so more code is added to the B-MAC protocol code. These overheads, when added to the CARP memory overheads given in Table 6.1, still meet the goal of keeping memory overhead at less than 5% of total Mica2 memory.

Table 7.10: Rate Limiting Memory Overhead

	Additional memory used (over application with anti-replay)	Percent additional Mica2 memory used
T-MAC	1,741 bytes	1.3%
B-MAC	4,348 bytes	3.2%

7.4 Summary

In this chapter, rate limiting was introduced as a technique for protecting network lifetime in the face of denial-of-sleep attacks. The rate-limiting mechanism is based on lightweight intrusion detection techniques tailored to the extreme resource constraints of WSN platforms. As shown here, these techniques can be used to maintain network lifetimes that are comparable to unattacked network lifetimes while still maintaining 64% or better throughput in the face of a subtle broadcast attack under the B-MAC protocol. Under T-MAC, throughput is much less severely affected and WSN lifetime is still preserved. When throughput cannot be maintained, this technique can be used to reduce energy consumption to an arbitrarily low level until a denial-of-sleep attack is lifted.

The next chapter describes the final component of the Caisson suite, the sensor anti-jamming engine (SAJE). This mechanism detects and mitigates jamming attacks against WSN deployments.

Chapter 8

Jam Detection and Mitigation

If there is no confidence in winning, one should retreat . . .,
reserving energy and resources for the next opportunity.
- Sun Tsu, *The Art of War*

Jamming can pose a serious threat to sensor networks. As discussed in Section 4.1, some WSN medium access control (MAC) protocols are vulnerable to jamming-based denial of sleep attacks. Furthermore, as described in Section 2.5.4, sensor platforms generally have simple transceivers that are not designed to use spread-spectrum techniques to protect against jamming. Other techniques, therefore, must be employed to mitigate this threat.

Even if mechanisms exist to maintain communication during an attack, jam detection is an important component of jamming mitigation. One of the most effective anti-jamming techniques for WSN, DEEJAM (discussed in Section 2.5.4), has an energy consumption overhead of up to 150% whether or not the network is under attack [54]. By limiting the use of such techniques to periods when the network is being jammed, network lifetime can be increased significantly. Furthermore, network operators can be notified of the jamming so that they can taken action against the jammer(s).

This chapter explores a proposed technique to detect jamming in sensor networks. It then describes its implementation for the Mica2 sensor platform as part of the Sensor Anti-Jamming Engine, or SAJE. The most basic action that can be taken to mitigate jamming is to cause nodes to

hibernate to conserve sensor energy supplies until the attack has abated. SAJE reacts to jamming by placing the node in hibernation for a random duration, then reawakening and testing to determine whether the jamming attack is still ongoing. Even short hibernation periods, averaging as little as 20 seconds each, can as much as double network lifetime over that of an unjammed network.

The rest of this chapter is organized as follows. Section 8.1 proposes a technique for effective jam detection. Section 8.2 gives details of the SAJE implementation on the Mica2 WSN platform, along with an analysis of its effectiveness and overheads. Section 8.3 examines the potential benefits of a SAJE's random hibernation technique to conserve sensor node energy in the face of jamming. Section 8.4 provides a chapter summary.

8.1 Techniques for Jam Detection in WSN

Because of the relative ease of jam-resistant communication using spread spectrum techniques, there is little research on the topic of detecting jamming attacks on wireless networks. Jam detection in WSN is necessary because, in general, the simplicity of sensor nodes' physical layer hardware and protocols do not support using spread spectrum to defeat jamming [14, 15, 25]. A precipitous drop-off in received packet rate is an indicator of possible jamming. It is also possible, however, that such a drop-off is the result of a reduction in transmitted packets due to node movement or other causes. In a static WSN, a reduction in received traffic might be caused by reduced transmit powers as node batteries are depleted over time. A reduction in received traffic alone, therefore, is not sufficient to indicate the presence of jamming.

8.1.1 Jamming Identification

In their paper on jamming in WSN, Xu et al. [52] identify a relationship between packet delivery ratios (PDR) and received signal strength indicator (RSSI) values in sensor network communications. RSSI values associated with received packets are easily obtained from most WSN platform transceivers, including the CC1000 transceiver on the Mica2. RSSI can also be sampled any time that the radio is active to determine background radio noise levels. Xu's mechanism for calculat-

ing PDR compares the number of packets that pass the CRC check to the total number of received packets. Using this technique,

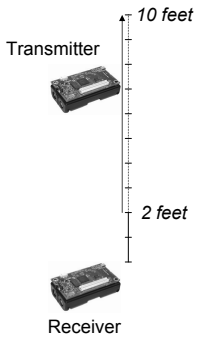
$$PDR = \frac{Pkts_{passed\_CRC}}{Pkts_{passed\_CRC} + Pkts_{failed\_CRC}}. \tag{8.1}$$

Another technique for calculating PDR, not considered by Xu, et al., relies on sequentially numbered packets to identify missing sequence numbers. Using this technique,

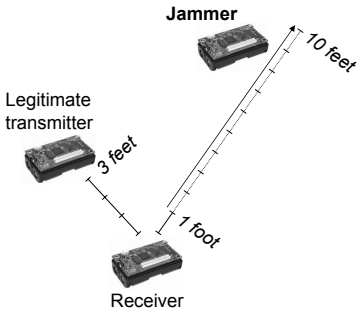
$$PDR = \frac{received\ packets}{total\ packets}, \tag{8.2}$$

where $received\ packets$ is the total number of packets successfully received during a certain period and $total\ packets$ is the difference between the sequence numbers of the first and last packets received during the same period. If the network includes anti-replay protection or packet counters for some other purpose, these counter values can be used for the PDR calculation. If the network does not use anti-replay protection and there is no application-layer requirement to add a sequence number to packets, adding sequence numbers would add data transmission overhead. While both of these techniques for calculating PDR are feasible for jam detection, there are tradeoffs in overhead and jam detection accuracy. These tradeoffs are analyzed in Section 8.2.2.

To confirm the results presented by Xu, et al., and to further explore the potential of using the relationship between PDR and RSSI to identify jamming attacks, experiments were performed to examine this relationship using Mica2 sensors. Figure 8.1 depicts the layout of these experiments. Packets were transmitted from one sensor to another at distances ranging from two feet to ten feet using a transmit power of -5 dBm, as shown in Figure 8.1(a). A similar set of experiments tested the effects of jamming on WSN communication. Each of the four types of jamming described in Section 2.5.4 were implemented in TinyOS [16] on the Mica2 platform. For the jamming tests, the legitimate transmitter was placed three feet from the receiver. The jammer's distance was varied from one foot to ten feet using a transmit power of -4 dBm, as shown in Figure 8.1(b). The jammer's transmit power is slightly higher than the legitimate transmitter's power setting. A real jammer would try to overwhelm legitimate nodes by transmitting at a higher power.



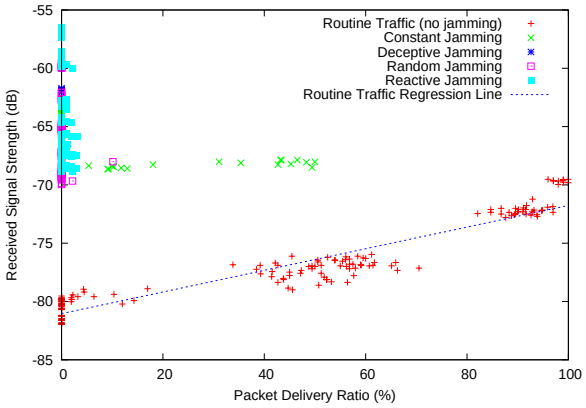
(a) Legitimate traffic experiments.



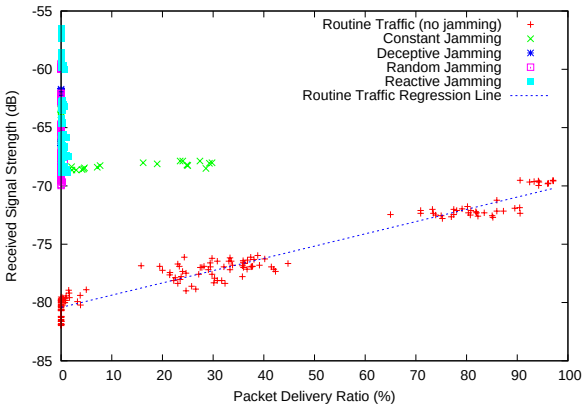
(b) Jamming experiments.

Figure 8.1: Node layout for initial PDR, RSSI experiments.

A series of ten experiments was conducted at each legitimate transmitter and jammer distance setting. Each experiment consisted of a series of packets transmitted by the legitimate transmitter. The receiver logged the packets and, at the end of each run, calculated the PDR using both of the methods described above. Each PDR was paired with the maximum RSSI value sampled during the experiment. Figure 8.2 shows the relationship between PDR and RSSI from these experiments using each of the above techniques for calculating PDR. As a sender's distance from the receiver increases, there is a decrease in PDR and RSSI associated with that sender. The plotted (PDR, RSSI) pairs during jamming attacks occupy the upper-left portion of the plot and are clearly distinct from legitimate traffic. It should be noted that the PDR values in in Figure 8.2(a) are gen-



(a) PDR calculated using CRC failures.



(b) PDR calculated using missing packets.

Figure 8.2: Relationship between RSSI and PDR in WSN communications.

erally higher than those in Figure 8.2(b). Packets for which the CRC check fails are counted in both cases. Packets that are not received at all, however, are not considered when PDR is solely dependent on packets which fail the CRC check.

8.1.2 An Algorithm for Jam Detection

Based on experiments similar to the ones described above, Xu et al. [52], describe a potential jam detection algorithm. First, PDRs and maximum RSSI values are collected for neighboring nodes during a collection period at the beginning of a network's deployment. At the end of the collection period, a statistical analysis is performed on the resulting data to define the relationship between PDR and RSSI. Once the collection phase is complete, the jam detection mechanism enters a steady-state period, during which it periodically checks for jamming using a two-phased process. During Phase I of jam detection, packets received and missed are tracked for neighboring nodes, much as they are during the collection phase. At the end of each jam check interval, PDRs for each neighbor are examined. If all PDRs are below a set threshold, the transceiver is polled for a series of RSSI measurements. The highest RSSI value is paired with the highest PDR obtained during Phase I. This (PDR, RSSI) pair is compared to the data taken during the collection period to see if it is consistent with the recorded data. If not, a jamming attack is suspected.

8.2 Jam Detector Implementation

To test the effectiveness of the above technique, it was implemented in TinyOS and tested in a B-MAC network. The sensor application with which the jam detection mechanism was tested is a simple environmental monitoring application in which each WSN node periodically takes readings from an attached sensor, in this case, a light sensor. Sensor readings are periodically packetized and broadcast to the other nodes in the network.

For the *collection period* of the jam detection algorithm, memory is set aside for a series of (PDR, RSSI) pairs, or *regression points*. The number of regression points is configurable and is set based on tradeoffs between memory overhead, desired duration of the collection period, and

the acceptable accuracy of jam detection. These tradeoffs are discussed in Section 8.2.2. As traffic is received, each node keeps track of packet statistics for its neighbors, along with the highest RSSI value for incoming traffic from each neighbor. At an interval dictated by the user, a PDR is calculated for each node from which traffic has been received during that period using either the bad-CRC or missed-packets method described above, and the resulting (PDR, RSSI) pair is added to the set of regression points. Once the regression point table is full, a simple linear regression is performed using the least-squares method to determine a slope and intercept for the line most closely matching the set of points. A simple linear regression is used to minimize the computational cost of performing the regression and of comparing future (PDR, RSSI) pairs to the regression in cases when Phase II of the steady-state jam detection period is required. The calculation of this regression marks the end of the collection period.

The *steady-state period* of the jam detection mechanism uses per-neighbor counters to track properly received and missed packets. After a period of time defined as the *jam check interval*, PDRs for each neighbor are calculated and compared to a threshold to determine whether Phase II, or RSSI sampling, is required. If the maximum PDR determined in Phase I is below a given threshold (set by default to 65%), Phase II is performed. During Phase II of jam checking, RSSI is sampled every 5 ms for 100 ms. The highest three resulting RSSI readings are averaged (to reduce the impact of outliers) and this average is paired with the highest PDR recorded in Phase I. This pair is compared to determine its proximity to the regression line computed during the collection period. If this point is within a specified range of the regression, no jamming is identified. If the point is outside this range, the node is considered to be jammed. Figure 8.3 is a flow diagram of the jam detection mechanism described above.

To test this implementation, several experiments were conducted using a four-node WSN deployed at ground level in a grassy field. After completion of the collection period of the jam detection mechanism, each of the four types of jammer was introduced into the network to ensure jamming was detected for each type. The jamming was detected in all cases. For this technique to be considered effective, there must be a low likelihood of false positives, or non-jamming events that are identified by network nodes as jamming. In the extreme case, a node might be temporarily

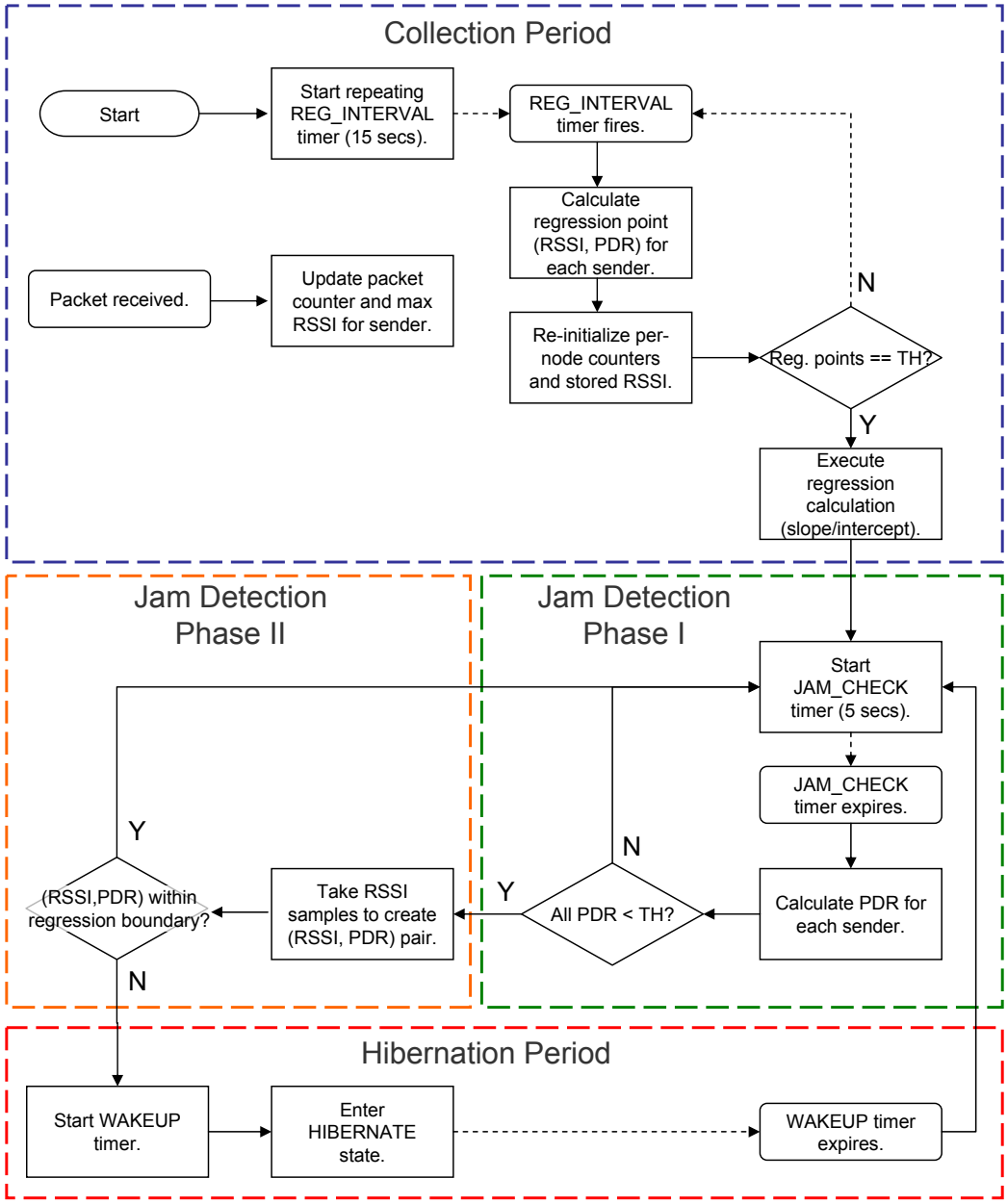


Figure 8.3: Flow diagram for SAJE jam detection mechanism.

physically blocked from other nodes in the network, causing PDRs to drop to 0%. In such cases, a false positive reading might occur if the regression intercept calculated during the collection period was such that its value, plus the width of the regression buffer (see Figure 8.4) is below the highest level of background noise. To check for false positives, all network nodes except one were turned off to artificially reduce PDRs to zero. This caused the remaining node to check for jamming once every jam check interval, but did not cause the node to believe it was jammed.

These experiments indicate that this technique is effective, however, a more rigorous analysis is required to determine the probability of either a false positive or false negative (failure to detect a jamming event) for a given scenario. There are two cases in which a false positive reading can occur. The first case is if a legitimate (PDR, RSSI) pair lies outside the regression buffer region and the highest neighbor PDR is below the PDR threshold. This would be mistakenly identified as jamming. Secondly, as described above, if the result of the regression is such that the regression intercept plus the regression buffer size is lower than the level of background noise, a false positive reading would occur at very low PDRs. A false negative can occur if the regression is such that when the buffer is added above the regression line, it overlaps the jamming region. See Figure 8.4 for a graphical representation of the background noise, jamming, and regression buffer regions. The jamming region occupies the upper left portion of Figure 8.4, which represents network con-

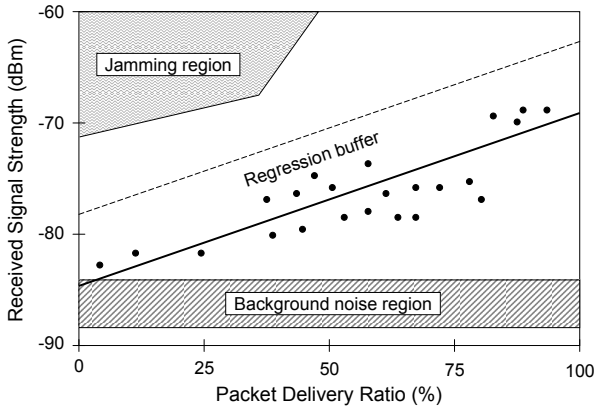


Figure 8.4: Example regression and region definitions. The solid line represents the results of linear regression performed during a jam detection collection period. (PDR, RSSI) pairs that are below the dashed line are assumed to be the result of legitimate traffic.

ditions in which PDRs are low, but RSSIs are high, which is the natural condition under which to assume jamming is present. The high RSSI readings are not necessarily caused by the jamming node. As long as the jammer's signal strength is high enough to corrupt packets that are being received by a particular node, PDR at that node will drop. The high RSSI readings may come from a legitimate node that is attempting to transmit, or even from a packet that was successfully received during the RSSI sampling period (Phase II of jam detection).

A series of experiments were conducted to evaluate the probability of each of the above potential causes of false positive and false negative readings. Using the outdoor network described above, a series of 50 iterations of the jam detection collection period were executed using the parameters given in Table 8.1. One of the nodes was connected via a serial cable to a computer to collect the PDR and RSSI for each regression point and the slope and intercept values for the linear regressions calculated on the device. This resulted in a total of 1000 regression points and 50 regression results.

The first step in calculating the probability of a false positive was to calculate the probability of a valid (PDR, RSSI) pair lying outside the regression buffer. To calculate this probability, the distances from the regression line for each of the 1000 regression points was calculated and a probability distribution of this distance was created using an automated tool called EasyFit [94]. The best-fit probability density function (PDF) was a Weibull distribution with parameters $\alpha = 1.2659$, $\beta = 1.6399$, and $\gamma = -0.00673$. Based on the corresponding cummulative distribution function (CDF), the probability of a point lying more than 7 dBm above the regression line was 0.00095. Next, the probability that a regression intercept might be sufficiently low that it, plus the buffer,

Table 8.1: Jam Detection Scenario Configuration

Parameter	Value
Regression points	20
Regression interval	20 seconds
PDR calculation technique	Missed packets
Regression buffer width	7 dBm
Combined network offered load	6 pps
Packet size	36 bytes

was below the noise floor was determined. Previous experiments indicated that the range of background noise in the test environment was -84 to -96 dBm. The 50 regression intercepts calculated during the experiments described above were analyzed using EasyFit, resulting in a normal distribution with parameters $\sigma = 1.4802$ and $\mu = -85.1$ dBm. The regression intercept would have to be lower than -91 dBm for the buffer region to intercept below the noise floor. The probability of such a low regression result was 0.000034. Finally, the probability of a false negative, or the failure to detect a jamming event, can be calculated using the same CDF of regression intercepts. In this case, the regression intercept would have to be -80 dBm or higher for the regression buffer to cross into the jamming region at zero PDR for the jamming model used in these experiments. This probability was 0.000285. In this experimental scenario, the overall probability of either a false positive or a false negative are well under one-quarter of 1%.

8.2.1 Jam Detection Overhead

In this section, the cost of jam detection in terms of energy consumption and memory overhead is examined.

Energy Consumption Overhead

Energy overhead is incurred when the radio is in the receive or transmit state when it could otherwise be in sleep mode. This only happens during Phase II of the steady-state period, when RSSI sampling is required because all PDRs are below a predetermined threshold. During the collection phase, PDR and RSSI data is collected based on legitimately received packets and there is no additional active radio time. The PDRs calculated during Phase I of the steady-state period also incur no lifetime overhead for the same reason.

The overhead of RSSI sampling in Phase II of jam detection is dependent on the WSN platform and the MAC protocol used in the network. This analysis assumes a 20-node single-hop cluster of Mica2s using B-MAC with a cluster-wide offered load of one packet per second (pps) of broadcast traffic. It exposes worst-case overheads because it assumes that the RSSI check is performed at the end of every jam check interval, which is unlikely. In fact, the technique of

comparing PDR values to a threshold is done specifically to avoid the need to sample RSSI values unless an attack is suspected due to a significant drop in observed PDRs.

The energy cost of RSSI sampling is based on its duration, which was set to 100 ms for these experiments on the Mica2 platform. This sampling duration is based on the time required to transmit a packet using the CC1000 radio. A default maximum-sized B-MAC packet is 41 bytes and a maximum-sized packet in S-MAC or T-MAC is 250 bytes [7, 29]. These packets take 19 ms and 104 ms to transmit, respectively. Since the goal of a jamming attack is to minimize throughput, even a random jammer, which intersperses jamming events with idle periods, should set the size of the idle periods such that a legitimate packet is unlikely to "slip through." An idle period of 100 ms or longer is likely to allow many packets through, especially B-MAC packets and reasonably-sized S-MAC and T-MAC packets. A 100-ms RSSI collection period is considered sufficient, therefore, to detect most effective jamming events. Recall that even if the RSSI check does not coincide with a jammer's active period on one check, if PDRs remain below threshold, the jammer is likely to be detected during a subsequent jam check.

For the S-MAC protocol on the Mica2, the default duty period is 130 ms per frame. The 100-ms RSSI check should be timed to coincide with the duty period in case a jammer is attempting an energy-efficient attack by jamming only during this predictable period. Since the default S-MAC active period is longer than the 100-ms RSSI sampling period, there is no additional radio overhead. For the T-MAC protocol, the RSSI check should also coincide with the beginning of a frame so that it overlaps with the duty period. For the T-MAC protocol, a moderate-sized packet of 64 payload bytes is assumed (a longer average packet size would decrease the overhead). At one packet per frame, each node in a one-hop neighborhood is awake for the average duration of the T-MAC packet backoff (7 ms), plus the time required to transmit or receive one packet (30 ms), plus the duration of the adaptive timeout (54 ms). This leaves an extra 9 ms of required radio active time for each jam detect interval. For a jam check interval of 10 seconds, this overhead is approximately 0.8%. Figure 8.5 shows how this overhead decreases as the jam check interval is increased.

Since B-MAC does not schedule a cluster-wide active period, it is not possible to make

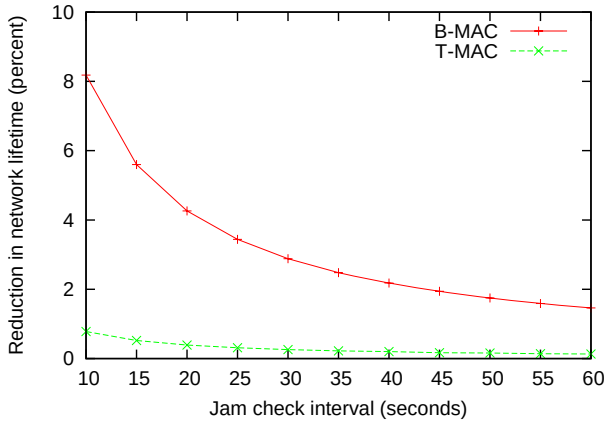


Figure 8.5: Worst-case jam detection energy-consumption overhead for CC1000 platform.

RSSI check periods exactly coincide with node awake time. This results in higher overheads for B-MAC. At a traffic rate of 1 pps, B-MAC receivers are awake for an average of 102 ms per second, or 10.2% of the time. As a result, an overall average of 10.2% of each RSSI check period will overlap with transmitted or received packets. The rest of the RSSI check period is charged as additional receive cost to the nodes in the network. Figure 8.5 shows the overhead of jam checking across a range of jam check intervals. Using a 10-second jam check interval, the network lifetime overhead is 8.2% for B-MAC. Again, this is the worst case overhead assuming that PDRs are always below threshold, requiring RSSI sampling at the end of every jam check interval. It is much more likely that the requirement to do RSSI sampling would be extremely infrequent, especially if thresholds are set properly. In the case that the PDR threshold is breached at some high percentage yet no jamming is detected, a logical response would be to adjust the threshold or the jam check interval to reduce energy overhead. When the jam check interval is increased to 30 seconds, B-MAC network lifetime us decreased by less than 3% if an RSSI check is required every time.

For the energy consumption overhead analysis on the Tmote Sky platform, a 20-node single-hop network is again assumed. The effective data rate of the Tmote's CC2420 transceiver is significantly higher than the Mica2's CC1000 (250 kbps versus 19.2 kbps). The simulated traffic rate for this analysis is an offered load of two broadcast packets per 500-ms frame. For this analy-

sis, the RSSI sampling duration for the Tmote Sky is 10 ms. Again, this is based on the transmission duration of IEEE 802.15.4-compliant packets. It takes 4.1 ms to transmit a maximum-sized 128-byte ZigBee packet at the CC2420's 250-kbps data rate. An idle period between jamming events for a random jammer is, therefore, unlikely to be set to something longer than 10 ms, so a 10-ms RSSI sampling duration will identify most effective jamming scenarios.

For S-MAC nodes, the 10% duty cycle has nodes awake for the first 50 ms of every frame. A 10-ms RSSI sampling period fits easily into this radio active time. For the T-MAC configuration tested, nodes are active for an average of the first 19 ms of each frame. Again, the 10-ms sampling period fits into this active period. Under B-MAC, the sampling period cannot be timed to coincide with network-wide active periods, so the sampling period is again charged as radio receive time. Under G-MAC, the sampling period is timed to coincide with the network-wide awake period at the beginning of the contention-free portion of each frame when nodes are awake to receive the GTIM and any broadcast packets. For this network model, the network is awake for at least 4.9 ms at the beginning of the contention-free period and the remaining 5.1 ms of the RSSI sampling period is charged as radio receive time. Worst-case jam detection costs for B-MAC and G-MAC are shown in Figure 8.6.

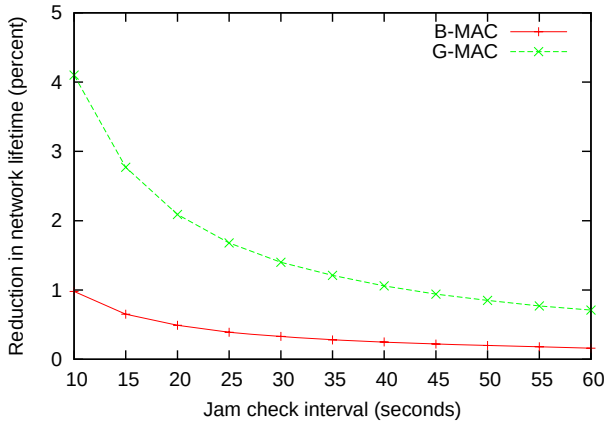


Figure 8.6: Worst-case jam detection energy-consumption overhead for CC2420 platform.

Memory Overhead

The jam detection implementation described above uses a table of four-byte entries for the (PDR, RSSI) pairs gathered during the collection period for regression analysis. For 20 regression points, this table totals 80 bytes, or slightly less than 2% of a Mica2's memory and less than 1% of a Tmote's memory. A 50-entry regression table uses 200 bytes, which approaches 5% of a Mica2's memory. During the collection period, 2 bytes of data are used to track received and missed packet counts for neighboring nodes so that PDRs for regression points can be calculated. All neighbors need not be tracked during the collection phase. The experiments described above collected data from four neighbors to produce consistent regressions. Setting aside memory for up to 10 nodes, or 20 bytes, is likely sufficient for use in real sensor network deployments. During the steady-state period, missed and received packets for all neighbors must be tracked. If a node has more than 10 neighbors, the memory set aside for the (PDR,RSSI) pairs used in regression analysis can be reused for this purpose, allowing for an additional 40 neighbors (if a 20-entry regression table is used). The total amount of memory required for a 20-point regression table and the neighbor received and missed packet data is 100 bytes. The equals 2.4% of the RAM on a Mica2 sensor and 1% of the RAM on a Tmote.

There is memory overhead associated with the extra program code, as well as with the data structures described above. Table 8.2 shows the increase in Mica2 memory used when the SAJE implementation is added to a test application. This overhead assumes a maximum of 40 neighbors and a 20-entry regression table.

Table 8.2: Jam Detection Memory Overhead

Additional memory used (over test application alone)	Percent additional Mica2 memory used
4,761 bytes	3.5%

8.2.2 Tradeoffs in Jam Detection Parameters

There are a variety of parameters that can be adjusted to modify the behavior of the jam-detection mechanism. The configurable parameters and the effects of changing these parameters are discussed here.

Number of Regression Points

Experiments have shown that increasing the number of regression points has the effect of increasing the accuracy of the regression analysis, resulting in a reduced likelihood of a poor regression that could lead to false positives or false negatives. Each additional regression point, however, requires 32-bits of allocated storage space on each mote. This is problematic if node memory is at a premium. Furthermore, the duration of the collection period increases linearly with an increase in the number of regression points.

PDR Calculation Technique

Section 8.1 introduces two techniques for calculating packet delivery ratios. Using missed packets to calculate PDRs captures more information because it considers packets that are never received, not just those that fail a CRC check. This generally leads to a wider range of PDRs and led to slightly more accurate jam detection results in experiments. Using missed packets to calculate PDRs requires that sequence numbers be added to packets if such sequence numbers are not in use. Under the B-MAC protocol, the overhead of adding a 1-byte counter to every packet is approximately a 1.1% reduction in network lifetime under the network model considered here. A detailed analysis of the overheads of adding packet counters to network traffic across WSN platforms and MAC protocols can be found in Section 6.2.2.

Regression Interval

A longer period of time between the calculation of PDRs during the collection period allows for more data to be considered. Experiments show that more accurate data tends to improve the con-

sistency of regression results. The duration of the collection period depends on the the number of neighbors, the number of regression points, and the duration of the regression interval. In a five-node cluster with a cluster offered load of 6 pps and using 50 regression points and a 30-second regression period, the duration of the collection period is approximately 8 minutes. A more reasonable 1 pps offered load would require the regression period to be multiplied 6-fold to achieve comparable results, increasing the duration of the collection period to approximately 48 minutes. Increasing the regression interval duration results in a linear increase in the duration of the collection period.

Jam Check Interval

A short jam check interval detects jamming attacks more quickly, allowing mitigation mechanisms to be triggered sooner. However, it also results in more required RSSI sampling periods over time. If PDR thresholds are not set carefully, this could result in a decrease in network lifetime as discussed in Section 8.2.1.

PDR Threshold

This threshold is used to determine whether RSSI sampling must be done at the end of a jam check interval. A low threshold value results in fewer RSSI sampling instances. This decreases the potential cost of jam detection, but may result in missed jam detection opportunities, especially in random jamming scenarios in which an adversary wishes to reduce, but not stop, WSN communication. This research assumes that target networks will operate at high PDRs and uses a default threshold of 65%. This threshold can be reduced as appropriate for networks that can tolerate low PDRs. The PDR threshold might even be set dynamically based on the neighboring node PDRs observed during the collection period.

RSSI Sampling Duration

A longer RSSI sampling duration results in an increase in jam detection costs, especially if the RSSI sampling must happen often, but is also more likely to detect jamming events. Based on

the analysis of effective jamming scenarios in Section 8.2.1, the 100-ms sampling duration for the Mica2 is a balanced tradeoff between jam detection accuracy and overhead.

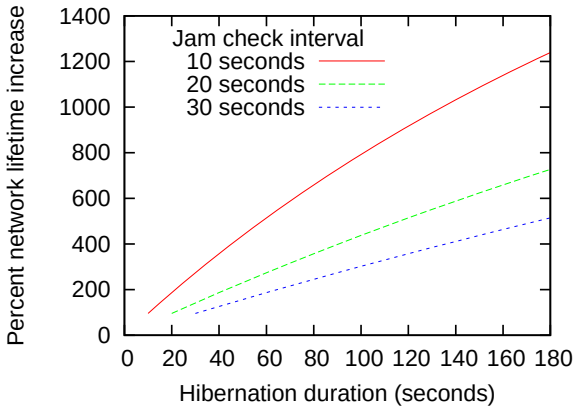
8.3 Jamming Mitigation

SAJE reacts very simply to jamming, using a technique called *random hibernation*. This technique places jammed nodes in a sleep (or hibernation) state to conserve energy in the face of an attack that might not only result in denial-of-service, but may also constitute a denial-of-sleep attack. The implementation described here has nodes hibernating for a duration chosen randomly from a uniform distribution $u(15000, 25000)$ ms. The hibernation duration is chosen randomly with high granularity to prevent an attacker from predicting the network-wide hibernation duration and using it to mount an energy-efficient jamming attack.

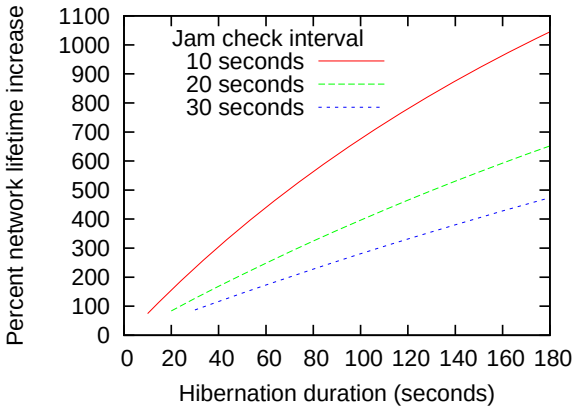
Figure 8.7 depicts the node lifetime increase for various average hibernation durations and jam check intervals across MAC protocols for the Mica2 platform. With a 10-second jam check interval and an average hibernation period of 3 minutes, an increase in network lifetime of over 1000% is possible in a network running any of the protocols analyzed here. Even a relatively short hibernation duration can result in a significant lifetime increase. An average hibernation duration of 20 seconds results in a better than two-fold increase in network lifetime for a 10-second jam check interval. The results of the same analysis on the Tmote Sky platform are given in Figure 8.8.

While a longer hibernation duration has the potential to dramatically increase sensor network lifetime, a shorter duration would result in less network disruption in the face of an intermittent jammer. Furthermore, hibernation constitutes a self-imposed denial-of-service attack and a shorter average hibernation duration allows the network to resume operation more quickly when the jamming attack has ended. Since network lifetime savings beyond 100% of the expected network lifetime would normally be unnecessary, an average hibernation duration that is only slightly longer than the jam check interval would usually provide the best trade-off between energy savings and rapid network recovery.

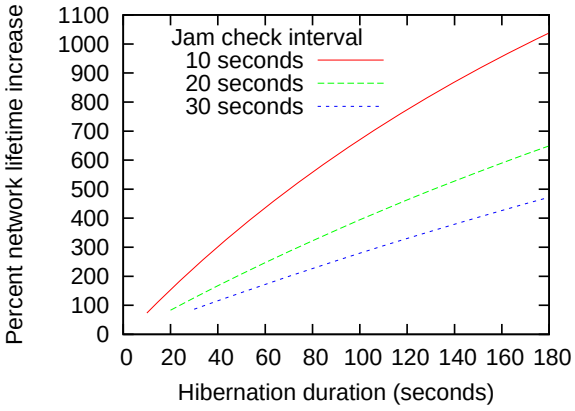
A 30-second jam check interval provides a reasonably fast jam detection with a worst-case



(a) S-MAC.



(b) T-MAC.



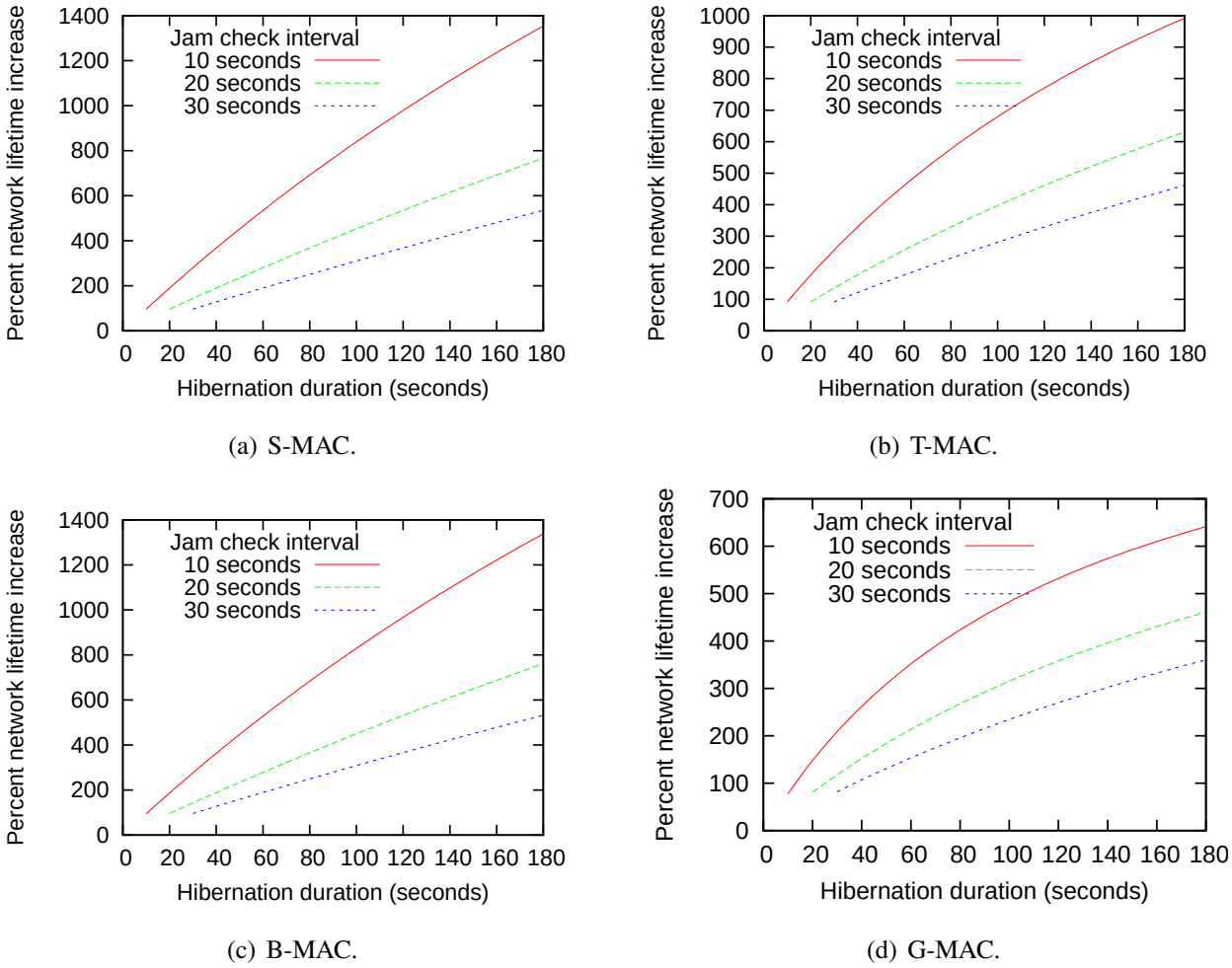
(c) B-MAC.

Figure 8.7: Percent network lifetime increase for jam detection and hibernation across jam check intervals for CC1000 platform.

overhead of 3% or less. This value, combined with an average hibernation duration of 35 seconds results in a network lifetime of at least 100% across MAC protocols and platforms in the face of a jamming attack while allowing quick recovery after the jamming attack has ceased.

8.4 Summary

This chapter has described the development, implementation, and testing of a technique to detect and react to jamming in WSN. Since spread spectrum techniques cannot be used to protect against jamming in WSN using most current platforms, other mitigation mechanisms are required. The



(a) S-MAC.

(b) T-MAC.

(c) B-MAC.

(d) G-MAC.

Figure 8.8: Percent network lifetime increase for jam detection and hibernation across jam check intervals for CC2420 platform.

jam detection mechanism described herein accurately detects jamming, with an overall probability of properly detecting jamming without false positives of 99.75% in the test scenario. Potential techniques for mitigating jamming once detected vary in effectiveness and overhead. SAJE's random hibernation mechanism allows nodes to conserve valuable energy resources, dramatically increasing network lifetime. This technique also allows nodes to resume normal behavior once the jamming attack has ceased.

Chapter 9

Conclusion

What is the good of experience, if you do not reflect?

- Fredrick the Great

This chapter presents a summary of the research described in this dissertation and highlights significant contributions of this work. It also discusses limitations in this research and possible directions for future work in the area of sensor network denial-of-sleep prevention and mitigation. Finally, it offers concluding remarks.

9.1 Summary of Research

This research explored the threat of denial-of-sleep in wireless sensor networks. Denial-of-sleep is a specific form of resource consumption attack that exploits the energy constraints of WSN platforms. By keeping WSN radios in receive mode when they should be off to conserve energy, WSN lifetime can be reduced from several months to only a few days. Furthermore, with detailed knowledge of WSN MAC protocols, effective attacks can be mounted against secured networks, even without penetrating encryption or authentication protocols. Finally, many of these attacks are difficult to detect because they can be launched in such a way that legitimate network traffic is not affected until nodes quit transmitting due to exhausted energy supplies.

The various WSN MAC protocols have different vulnerabilities, each of which must be

addressed for comprehensive denial-of-sleep prevention. Some MAC protocols, such as S-MAC and G-MAC, are vulnerable to counterfeit or replayed control traffic that can improperly cause the MAC protocol to keep sensor devices' radios in receive mode when they should be sleeping. Protocols that have nodes remain awake as long as there is traffic in the network, such as T-MAC and B-MAC, are susceptible to unauthenticated broadcast-based denial-of-sleep attacks. Even though malicious broadcast packets are dropped because they cannot be authenticated, radios remain awake to receive these packets, thus wasting energy. The energy-draining effect of this attack increases as the volume of unauthenticated traffic increases. Finally, some protocols are vulnerable to jamming-based denial-of-sleep attack. B-MAC nodes, for example, will remain awake permanently listening to preamble bytes transmitted by a deceptive jammer.

Each of these vulnerabilities requires a different prevention or mitigation technique. First, a WSN must be protected from unauthenticated packets using one of several existing packet authentication techniques tailored to the resource constraints of WSN nodes. Clustered Anti-replay Protection (CARP) provides anti-replay support with low overhead in large-scale WSN deployments. Clustered Adaptive Rate Limiting (CARL) detects potential denial-of-sleep attacks using a lightweight intrusion detection mechanism that identifies and reacts to potential denial-of-sleep attacks. By limiting the amount of time that WSN radios can remain active, CARL prevents these attacks from having the desired effect of quickly draining sensor node batteries. Finally, the Sensor Anti-jamming Engine (SAJE) detects various forms of jamming by correlating the network's packet delivery ratio (PDR) and received signal strength indicator (RSSI) values, and then causes affected nodes to hibernate to conserve energy during the jamming attack. Together, these components (CARP, CARL, and SAJE) comprise the Clustered Anti Sleep-deprivation for Sensor Networks, or Caisson, suite of denial-of-sleep prevention tools. These mechanisms are selectable at application compile time so that WSNs can be protected from threats specific to the MAC protocol used in the network.

This research classified denial-of-sleep attacks based on the level of protocol knowledge that the attacker has and the degree to which network encryption and authentication protocols can be penetrated by the attacker. It then used analytical modeling and attack implementations to

determine the impact of each attack on the various protocols and to determine the efficiency with which they can be carried out.

Each of the denial-of-sleep mitigation techniques was explored using analytical modeling to determine its overhead. Simulation was used to verify the effectiveness of the Caisson components and to verify the network lifetime overheads. Finally, the Caisson components were implemented on the Mica2 WSN platform for the S-MAC, T-MAC, and B-MAC protocols. These implementations show that the techniques are effective in maintaining network lifetime in the face of a variety of denial-of-sleep attacks. Furthermore, these implementations demonstrate that significant throughput can be maintained when the attack is one that allows legitimate network traffic so as to reduce the possibility of detection. Finally, the implementations show that the memory overheads are well within the limitation of a typical WSN platform, the Mica2.

9.2 Contributions

Don't fight a battle if you don't gain anything by winning.
- Field Marshall Erwin Rommel

This research makes two significant contributions to WSN research. First, it thoroughly analyzes and validates the effects of denial-of-sleep attacks on WSN MAC protocols and identifies the characteristics of these protocols that make them vulnerable to malicious attack aimed at draining energy on sensor devices. Second, the Caisson suite provides a low-overhead, platform-independent set of mechanisms to mitigate denial-of-sleep attacks in WSNs. Specific contributions to sensor network research include the following.

1. A thorough categorization and analysis of heretofore unexplored resource consumption attacks on sensor network energy supplies that target characteristics common among contention-based WSN MAC protocols.

2. The validation of denial-of-sleep attacks on WSN through the implementation of attacks on a physical platform, specifically, the Mica2 sensor platform.

3. The development of Caisson, a suite of mechanisms for mitigating the effects of denial-of-sleep attacks in WSN.

   (a) The design and implementation of the first known scalable, platform-independent, lightweight anti-replay mechanism for WSNs. This technique virtually eliminates the threat of replay-based denial-of-sleep attacks with energy consumption overheads of 0% to 4.6%, depending on the platform and MAC protocol.

   (b) The design and implementation of a module for detecting broadcast-based denial-of-sleep attacks and an adaptive rate limiting algorithm to mitigate the effects of these attacks. In Chapter 7, this technique is shown to maintain 71% or more of the expected network lifetime while maintaining 64% to 96% of network throughput in the face of broadcast-based denial of sleep attacks. If configured to maximize lifetime without regard for throughput, 100% or more of sensor network lifetime can be preserved.

   (c) The design and implementation of a jamming detection and hibernation mechanism to reduce the effects of jamming attacks on sensor network lifetimes. Chapter 8 shows that this mechanism accurately detects a variety of different types of jamming attacks with false positive rates of 0.25% or less.

4. A detailed analysis of the effectiveness and overheads imposed by the Caisson suite across a range of network configurations.

   (a) An analysis of the lifetimes of unprotected networks versus those of networks using Caisson, both in the face of denial-of-sleep attacks and when attacks are not being launched.

   (b) An analysis of sensor node memory required by each of the Caisson components.

9.3 Limitations and Future Research Directions

This research has concentrated on MAC layer denial-of-sleep vulnerabilities and defenses. As a result of this focus on the MAC layer, the analyses of the effects of these attacks, and of the benefits and costs of countermeasures, have been made using one-hop neighborhoods of sensor nodes.

Since the MAC layer arbitrates access to the wireless channel for direct device-to-device communications, limiting the analysis to a single neighborhood is logical and helped keep the problem tractable. However, it does not provide insight into how denial-of-sleep attacks might affect nodes that are not directly affected by the malicious attacker. Some attacks, such as unauthenticated broadcast, would likely have minimal energy-consumption impact outside the nodes that are in direct radio reception range of the attacker. Other attacks, such as control packet replay attacks, could have a more far-reaching effect on the network. It would be interesting to analyze how these MAC-layer attacks affect a multihop network.

Another limitation of this research is the relatively small scale of the testing conducted with the actual Mica2 sensor devices, driven by time, manpower, and equipment constraints. The area in which more devices might provide more interesting results are the jam detection tests. More devices would provide a wider range of (RSSI,PDR) data points, which would be expected to improve the accuracy of the jam detection mechanism. The tests on the effectiveness of the anti-replay and rate-limiting mechanisms are sufficient to determine their effectiveness and overheads.

There is clearly potential for future denial-of-sleep research at higher layers in the WSN protocol stack. As WSN research matures, a handful of multi-hop transport protocols, data aggregation algorithms, and WSN applications are becoming prevalent in the research community. Some of these protocols may be vulnerable to attacks similar to those explored in this work. Attacks that target higher-layer protocols could escape detection by the Caisson mechanisms. For example, a particular network-layer protocol might be vulnerable to an attack that triggers a flood of routing messages, which could reduce network lifetime. These messages, if initiated at the network layer of a victim node, would carry correct MAC-layer authentication and replay counters and, therefore, would not be detected by Caisson. It should be noted, however, that the MAC-layer protections that prevent nodes from accepting unauthenticated and replayed traffic effectively prevent the use of counterfeit or replayed packets when targeting higher layers in the protocol stack, so some other attack vector would have to be devised. Similar attacks could also be launched at the application layer. One example of an application-layer attack that does not rely on malicious packets is an *actuation attack*, in which attacking nodes are placed in close proximity to legitimate

nodes and take action to physically trigger sensors on victim nodes [95]. Examples of actuator attacks include movement to trigger motion sensors, artificial light to trigger light sensors, or the release of chemicals to trigger chemical agent sensors. Such attacks could cause victim nodes to remain awake, repeated reporting sensor readings. Depending on the application and the MAC and network protocols used in the network, this attack might also keep neighboring nodes awake to receive these reports and they might keep nodes on the path to a base station awake routing the traffic.

9.4 Concluding Thoughts

This dissertation has shown, through modeling and implementation, the susceptibility of modern WSN MAC protocols to devastating denial-of-sleep attacks. A detailed analysis of denial-of-sleep vulnerabilities, along with a description of attacks that target these vulnerabilities, makes evident the ease with which attacks can be launched against these protocols. Encrypting and authenticating network traffic is not sufficient to protect networks from denial-of-sleep attacks. While necessary, anti-replay protection is also not sufficient to protect against many attacks.

This work has also described techniques for protecting current and future WSN MAC protocols from attack. These protections include anti-replay support (CARP), adaptive rate-limiting (CARL), and jamming mitigation (SAJE). These techniques, if applied appropriately based on the MAC protocol used in a WSN, will mitigate the effects of denial-of-sleep attacks while maintaining reasonable throughput, depending on the type of attack launched. Furthermore, the Caisson mechanisms do not reduce network throughput during routine network operation. Energy consumption and memory overheads are minimized. The full Caisson suite reduces sensor network lifetime by less than 5% under routine network conditions. For some MAC protocols and platforms, this overhead is much less. Caisson consumes less than 7% of the combined program memory and RAM on a Mica2, one of the most resource-constrained sensor platforms.

To date, this research has been published or has been accepted for publication in the form of two journal articles and four conference papers. The analysis of MAC protocol vulnerabili-

ties and effects of denial-of-sleep attacks on MAC protocols was published in the *Proceedings of the Seventh Annual IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop* [30]. This analysis, along with the description of denial-of-sleep attack implementations and an analysis of their efficiency will be published in an upcoming issue of *IEEE Transactions on Vehicular Technology* [24]. The initial description of the CARP mechanism was published in the *Proceedings of the Eighth Annual IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop* [92]. The CARL mechanism and its implementation for the B-MAC protocol was published in the *Proceedings of the 2007 IEEE/AFCEA Military Communications Conference (MILCOM)* [93]. The T-MAC implementation of the CARL mechanism, along with a description of the OPNET process models developed for this research, was published in *OPNETWORK 2007* [96]. Finally, a survey on denial-of-service attacks in WSN, along with an introduction to the denial-of-sleep attack, was published in the January-March 2008 issue of *IEEE Pervasive Computing Magazine* [97]. Additional publications are expected.

This research re-emphasizes the importance of considering security early in the network protocol development process. Without this, vulnerabilities inherent in these network protocols, and other software, will increasingly become targets for malicious attack. As summed up in the quote by GEN Omar Bradley preceding the introduction to Chapter 4: "If we continue to develop our technology without wisdom or prudence, our servant may prove to be our executioner."

Bibliography

[1] T. Narten, "Internet routing," in *Symposium on Communications, Architectures, and Protocols*, pp. 271–282, Aug. 1989.

[2] LAN MAN Standards Committee of the IEEE Computer Society, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. New York, NY: IEEE STD 802.3 - 2005 edition, 2005.

[3] ——, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*. New York, NY: IEEE Std 802.11 - 1999 edition, 1999.

[4] W. Ye, J. Heidemann, and D. Estrin, "A flexible and reliable radio communications stack on motes," USC/ISI, Technical Report ISE-TR-565, 2002. [Online]. Available: http://www.isi.edu/

[5] R. Jain, *The Art of Computer Systems Performance Analysis*. New York: John Wiley and Sons, 1991.

[6] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1567–1576, June 2002.

[7] T. VanDam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *First ACM International Conference on Embedded Networked Sensor Systems*, pp. 171–180, Nov. 2003.

[8] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Second ACM International Conference on Embedded Networked Sensor Systems*, pp. 95–107, Nov. 2004.

[9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Second ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 88–97, Sept. 2002.

[10] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 71–80, May 2003.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications*, vol. 3, no. 8, pp. 102–114, Aug. 2002.

[12] "Smart Dust," Accessed Aug. 2006. [Online]. Available: http://www-bsac.eecs.berkeley.edu/

[13] Texas Instruments, "SmartRF CC2420 datasheet (Rev. B)," Accessed Aug. 2007. [Online]. Available: http://focus.ti.com/

[14] LAN MAN Standards Committee of the IEEE Computer Society, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification for Low-rate Wireless Personal Area Networks (LR-WPANs)*. New York, NY: IEEE Std 802.15.1 - 2002 edition, 2002.

[15] Texas Instruments, "SmartRF CC1000 datasheet (Rev. A)," Accessed Aug. 2007. [Online]. Available: http://focus.ti.com/

[16] "TinyOS community forum," Accessed Aug. 2007. [Online]. Available: http://www.tinyos.net/

[17] C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A dynamic operating system for sensor nodes," in *Third International Conference on Mobile Systems, Applications and Services*, pp. 163–176, June 2005.

[18] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *Mobile Networks and Applications (MONET) Journal, Special Issue on Wireless Sensor Networks*, vol. 10, no. 4, pp. 563–579, Aug. 2005.

[19] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, Nov. 2004.

[20] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systesm," in *ACM Conference on Programming Language Design and Implementation*, pp. 1–11, June 2003.

[21] CrossBow Corporation, "Mica2 datasheet," Accessed May 2006. [Online]. Available: http://www.xbow.com/

[22] Moteiv Corporation, "Tmote sky datasheet: Low power wireless sensor module," Accessed Feb. 2006. [Online]. Available: http://www.moteiv.com/

[23] J. Kurose and K. Ross, *Computer Networking: A Top-down Approach Featuring the Internet*, 3rd ed. Boston, MA: Addison-Wesley, 2005.

[24] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of denial of sleep attacks on wireless sensor network MAC protocols," to appear in *IEEE Transactions on Vehicular Technology*.

[25] LAN MAN Standards Committee of the IEEE Computer Society, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification for Low-rate Wireless Personal Area Networks (LR-WPANs)*. New York, NY: IEEE Std 802.15.4 - 2003 edition, 2003.

[26] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Fifth Symposium on Operating System Design and Implementation*, pp. 147–163, Dec. 2002.

[27] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16–27, Oct. 2000.

[28] M. Brownfield, N. Davis, and A. Fayez, "Wireless sensor network radio power management," in *OPNETWORK 2005*, Aug. 2005.

[29] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[30] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of denial of sleep attacks on wireless sensor network MAC protocols," in *Seventh Annual IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop*, pp. 297–304, June 2006.

[31] M. Brownfield, K. Mehrjoo, A. Fayez, and N. Davis, "Wireless sensor network energy-adaptive MAC protocol," in *IEEE Consumer Communications and Networking Conference*, pp. 778–782, Jan. 2006.

[32] "SourceForge.net," Accessed June 2006. [Online]. Available: http://sourceforge.net/

[33] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Hawaii International Conference on System Sciences*, pp. 8020–8029, Jan. 2000.

[34] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, Dec. 2004.

[35] H. Chan and A. Perrig, "ACE: An emergent algorithm for highly uniform cluster formation," in *First European Workshop on Sensor Networks*, pp. 154–171, Jan. 2004.

[36] K. Sun, P. Peng, P. Ning, and C. Want, "Secure distributed cluster formation in wireless sensor networks," in *22nd Annual Computer Security Applications Conference*, pp. 131–140, Dec. 2006.

[37] G. V. Crosby, N. Pissinou, and J. Gadze, "A framework for trust-based cluster head election in wireless sensor networks," in *Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pp. 13–22, April 2006.

[38] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 113–127, May 2003.

[39] W. Stallings, *Cryptography and Network Security: Principles and Practices*, 3rd ed. Upper Saddle River, New Jersey: Prentice Hall, 2003.

[40] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second International Conference on Embedded Networked Sensor Systems*, pp. 162–175, Nov. 2004.

[41] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 55, pp. 521–534, Sep. 2002.

[42] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," in *Eighth Annual Symposium on Network and Distributed System Security*, pp. 35–46, Feb. 2001.

[43] National Institute of Standards and Technology, "Skipjack and KEA algorithm specifications, version 2.0," May 1998. [Online]. Available: http://csrc.nist.gov

[44] Cylink Corporation, "Nomination of SAFER+ as candidate algorithm for AES," June 1998. [Online]. Available: http://www.princeton.edu/

[45] S. Kent and R. Atkinson, *Security Architecture for the Internet Protocol*. IETF RFC 2401, Nov. 1998.

[46] M. G. Gouda, Y. Choi, and A. Arora, "Antireplay protocols for sensor networks," Accessed Aug. 2004. [Online]. Available: http://www.cse.ohio-state.edu/

[47] Y. Xiao, S. Sethi, H.-H. Chen, and B. Sun, "Security services and enhancements in the IEEE 802.15.4 wireless sensor networks," in *IEEE Global Telecommunications Conference*, pp. 1796–1800, Dec. 2005.

[48] N. Sastry and D. Wagner, "Security considerations for ieee 802.15.4 networks," in *ACM Workshop on Wireless Security*, pp. 32–42, Oct. 2004.

[49] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. New York, NY: John Wiley and Sons, 2001.

[50] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler, "Keyless jam resistance," in *Seventh IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop*, pp. 143–150, June 2007.

[51] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, Oct. 2002.

[52] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Eleventh Annual International Conference on Mobile Computing and Networking*, pp. 46–57, May 2005.

[53] A. D. Wood, J. A. Stankovic, and S. H. Son, "JAM: A jammed-area mapping service for sensor networks," in *24th IEEE Real-Time Systems Symposium*, pp. 286–297, Dec. 2003.

[54] A. D. Wood, J. A. Stankovic, and G. Zhou, "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15.4-based wireless networks," in *Fourth Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communictions and Networks*, pp. 60–69, June 2007.

[55] R. Roman, J. Zhou, and J. Lopez, "Applying intrusion detection systems to wireless sensor networks," in *IEEE Consumer Communications and Networking Conference*, pp. 640–644, Jan. 2006.

[56] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *First ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, pp. 16–23, Oct. 2005.

[57] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Sixth Annual International Conference on Mobile Computing and Networking*, pp. 255–265, Aug. 2000.

[58] M. Brownfield, Y. Gupta, and N. Davis, "Wireless sensor network denial of sleep attack," in *Sixth Annual IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop*, pp. 356–364, June 2005.

[59] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Seventh International Workshop on Security Protocols*, pp. 172–194, April 1999.

[60] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Second IEEE International Conference on Pervasive Computing and Communication*, pp. 309–318, March 2004.

[61] D. C. Nash, T. L. Martin, D. S. Ha, and M. S. Hsiao, "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *Third International Conference on Pervasive Computing and Communications Workshops*, pp. 141–145, March 2005.

[62] T. K. Buennemeyer, F. Munshi, R. C. Marchany, and J. G. Tront, "Battery-sensing intrusion protection for wireless handheld computers using a dynamic threshold calculation algorithm for attack detection," in *Hawaii International Conference on System Sciences*, p. 163b, Jan. 2007.

[63] R. Racic, D. Ma, and H. Chen, "Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery," in *Second International Conference on Security and Privacy in Communication Networks*, pp. 1–10, Aug. 2006.

[64] V. Livingston, "Two billion GSM customers worldwide," June 2006. [Online]. Available: http://www.prnewswire.com/

[65] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in TinyOS," in *First Symposium on Networked System Design and Implementation*, pp. 1–14, March 2004.

[66] D. Manjunath and R. C. Hansdah, "A review of current operating systems for wireless sensor networks," in *22nd International Conference on Computers and Their Applications*, pp. 387–394, March 2007.

[67] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Second ACM Conference on Embedded Networked Sensor Systems*, pp. 214–226, Nov. 2004.

[68] A. Park, K. Perumalla, V. Protopopescu, M. Shankar, F. DeNap, and B. Gorman, "On evaluation needs of real-life sensor network deployments," in *Second European Modeling and Simulation Symposium*, Oct. 2006.

[69] M. Brownfield, "Energy-efficient wireless sensor network MAC protocol," Ph.D. dissertation, Virginia Polytechnic Institute and State University, March 2006.

[70] OPNET Technologies Inc., "OPNET Modeler," Accessed Aug. 2006. [Online]. Available: http://www.opnet.com/

[71] C. Ulmer, "SensorSimII," Accessed Nov. 2006. [Online]. Available: http://www.craigulmer.com/research/sensorsimii/

[72] L. F. Perrone and D. M. Nicol, "A scalable simulator for TinyOS applications," in *Winter Simulation Conference*, pp. 679 – 687, Dec. 2002.

[73] S. Park, A. Savvides, and M. B. Srivastava, "Sensorsim: A simulation framework for sensor networks," in *Third ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 104 – 111, Aug. 2000.

[74] "The network simulator - ns-2," Accessed Nov. 2006. [Online]. Available: http://www.isi.edu/nsnam/ns/

[75] E. Egea-Lopez, F. Ponce-Marin, and J. Vales-Alonso, "OBIWAN: wireless sensor networks with omnet++," in *13th IEEE Mediterranean Electrotechnical Conference*, pp. 777 – 780, May 2006.

[76] T. W. Carley, "Sidh: A wireless sensor network simulator," University of Maryland, Technical Report TR 2005-88, 2005.

[77] A. Sobeih, J. C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen, H.-Y. Tyan, and H. Lim, "J-Sim: A simulation and emulation environment forwireless sensor networks," *IEEE Wireless Communications*, vol. 13, no. 4, pp. 104 – 119, Aug. 2006.

[78] G. Chen, J. Branch, M. J. Pflug, L. Zhu, and B. K. Szymanski, "SENSE: A sensor network simulator," in *Advances In Pervasive Computing And Networking*, B. K. Szymanski, Ed. New York, NY: Springer-Verlag Telos, 2005, pp. 249 – 266.

[79] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," in *37th Annual Simulation Symposium*, pp. 221 – 228, April 2004.

[80] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "VisualSense: Visual modeling for wireless and sensor network systems," University of California, Berkeley, CA, Technical Memorandum UCB/ERL M05/25, July 2005.

[81] R. Barr, Z. J. Haas, and R. van Renesse, "Scalable wireless ad hoc network simulation," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*. Boca Raton, FL: CRC Press, 2005, pp. 279 – 311.

[82] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," in *First ACM Conference on Embedded Networked Sensor Systems*, pp. 126 – 137, Nov. 2003.

[83] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir, "ATEMU: A fine-grained sensor network simulator," in *First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 145 – 152, Oct. 2004.

[84] "Avrora - the AVR simulation and analysis framework," Accessed Aug. 2006. [Online]. Available: http://compilers.cs.ucla.edu/avrora/

[85] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, "EmStar: An environment for developing wireless embedded systems software," in *USENIX Annual Technical Conference*, pp. 283 – 296, July 2004.

[86] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. Boston, MA: McGraw Hill, 2000.

[87] P. D. Welch, "On the problem of the initial transient in steady-state simulation," IBM Watson Research Center, Yorktown Heights, NY, Technical Report, 1981.

[88] Y. W. Law, P. Hartel, J. denHartog, and P. Havinga, "Link-layer jamming attacks on S-MAC," in *Second European Workshop on Wireless Sensor Networks*, pp. 217–225, Feb. 2005.

[89] R. Negi and A. Perrig, "Jamming analysis of MAC protocols," Carnegie Mellon University, Technical Report, 2003.

[90] Y. W. Law, L. vanHoesel, J. Doumen, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols," in *Third ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 76–88, Nov. 2005.

[91] "Eyes - sensors and networking," Accessed Feb. 2008. [Online]. Available: http://www.eyes.eu.org/

[92] D. Raymond, R. Marchany, and S. Midkiff, "Scalable, cluster-based anti-replay protection for wireless sensor networks," in *Eighth Annual IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop*, pp. 127–134, June 2007.

[93] D. Raymond and S. Midkiff, "Clustered adaptive rate limiting: Defeating denial-of-sleep attacks in wireless sensor networks," in *AFCEA/IEEE Military Communications Conference*, pp. 1–7, Oct. 2007.

[94] MathWave Technologies, "EasyFit, v. 3.2," Accessed Sep. 2007. [Online]. Available: http://www.mathwave.com

[95] A. Czarlinska and D. Kundur, "Distributed actuation attacks in wireless sensor networks: Implications and countermeasures," in *Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pp. 3–12, April 2006.

[96] D. Raymond and S. Midkiff, "Analyzing wireless sensor network denial-of-sleep attacks and defenses using OPNET Modeler," in *OPNETWORK 2007*, Aug. 2007.

[97] ——, "Denial-of-service in wireless sensor networks: Attacks and defenses," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 74–81, Jan. 2008.

List of Acronyms

ACE Algorithm for Cluster Establishment

ACK Acknowledgment packet

ACL Access Control List

AES Advanced Encryption Standard

AM Active Messaging

AP Access Point

AR Active Ratio

ATIM Announcement Traffic Indication Message

B-MAC Berkley-Medium Access Control protocol

B-SIPS Battery-sensing Intrusion Protection System

CARL Clustered Adaptive Rate Limiting

CARP Clustered Anti-replay Protection

CBC Cipher Block Chaining

CDF Cumulative Distribution Function

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CSMA/CD Carrier Sense Multiple Access with Collision Detection

CTR Counter block cipher mode of operation

CTS Clear-to-send message

CVS Concurrent Versioning System

DCF Distributed Coordination Function

DEEJAM Defeating Energy-Efficient Jamming protocol

DES Data Encryption Standard

DIFS Distributed Interframe Space

DSSS Direct-Sequence Spread Spectrum

ECC Error-Correcting Code

EMU Energy Monitoring Unit

FDM Frequency Division Multiplexing

FHSS Frequency Hopping Spread Spectrum

FRTS Future request-to-send message

FSM Finite State Machine

FTP File Transfer Protocol

G-MAC Gateway-Medium Access Control protocol

GPRS General Packet Radio Service

GSM Global System for Mobile Communication

GTIM Gateway Traffic Indication Message

HB Higher is Better performance metric utility class

HEED Hybrid, Energy-efficient, Distributed Clustering protocol

HTTP Hypertext Transfer Protocol

IDS Intrusion Detection System

IP Internet Protocol

IPSec Internet Protocol Security suite

IV Initialization Vector

JAM Jammed Area Mapping protocol

KB Kilobyte

LAN Local Area Network

LB Lower is Better performance metric utility class

LEACH Low-energy Adaptive Clustering Hierarchy

LPL Low Power Listening

MAC Medium Access Control

MB Megabyte

MD Message Digest

NB Nominal is Best performance metric utility class

NBC Nuclear, Biological, and Chemical

OS Operating System

OSI Open Systems Interconnect

PCF Point Coordination Function

PDF Probability Density Function

PDP Packet Delivery Protocol

PDR Packet Delivery Ratio

pps packets per second

RAM Random Access Memory

RBS Reference Broadcast System

RSA Rivest, Shamir, and Adleman asymmetric cryptography protocol

RSSI Received Signal Strength Indicator

RTS Request-to-send message

SAFER+ Secure And Fast Encryption Routing, Plus

SAJE Sensor Anti-jamming Engine

SIFS Short Interframe Space

SINR Signal to Interference-plus-noise Ratio

S-MAC Sensor-Medium Access Control protocol

SMTP Simple Mail Transfer Protocol

SMS Short Messaging Service

SNEP Secure Network Encryption Protocol

SPINS Security Protocols for Networked Sensors

SYNC Synchronization packet

TA Adaptive Timeout

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TDMA Time Division Multiple Access

TESLA Timed, Efficient, Streaming, Loss-tolerant Authentication protocol

TIM Traffic Indication Message

T-MAC Timeout-Medium Access Control protocol

TOSSIM TinyOS Simulator

TTS Time to sleep

UDP User Datagram Protocol

USC/ISI University of Southern California's Information Sciences Institute

WPAN Wireless Personal Area Network

WSN Wireless Sensor Network

Appendix A

Wireless Sensor Platforms

Table A.1: Typical Wireless Sensor Network Platforms [66]

Platform	RAM	Code Memory	Transceiver	Frequency
BTnode3	64 KB	128 KB	Zeevo-BT/CC1000	2.4 GHz/868 MHz
Cricket	4 KB	128 KB	CC1000	433 MHz
ECR	2 KB	60 KB	TR1001	868 MHz
ESB	2 KB	60 KB	TR1001	868 MHz
EYES	4 KB	60 KB	TR1001	868 MHz
intelmote	64 KB	512 KB	Zeevo-BT	2.4 GHz
intelmote2	256 KB	32 MB	CC2420	2.4 GHz
MANTIS nymph	4 KB	128 KB	CC1000	315/433/868/915 MHz
mica	4 KB	128 KB	TR1000	433/916 MHz
mica2	4 KB	128 KB	CC1000	315/433/916 MHz
mica2Dot	4 KB	128 KB	CC1000	315/433/916 MHz
MICAz	4 KB	128 KB	CC2420	2.4 GHz
TelosA	2 KB	60 KB	CC2420	2.4 GHz
TelosB	10 KB	48 KB	CC2420	2.4 GHz
TinyNode 584	10 KB	48 KB	Xemics XE1205	868 MHz
Tmote Sky	10 KB	48 KB	CC2420	2.4 GHz
Smart-Its mote	4 KB	128 KB	Ericsson-BT/TR1001	2.4 GHz/868 MHz
Sun SPOT	512 KB (combined)		CC2420	2.4 GHz
XYZ mote	32 KB	256 KB	CC2420	2.4 GHz
ZebraNet Hardware	2 KB	60 KB	9XStream	902-928 MHz

Appendix B

OPNET Process Models

This appendix provides details on the OPNET process models developed as part of this research. Section B.1 describes the MAC protocol process models and Section B.2 provides details on the functioning of the energy consumption process model introduced in Section 3.5.4.

B.1 MAC Protocol Process Models

OPNET process models were developed for the S-MAC, T-MAC, and B-MAC WSN protocols as part of this research to develop and test techniques for defeating or mitigating denial-of-sleep attacks.

B.1.1 S-MAC Protocol Model

A description of the basic operation of the S-MAC protocol is given in Section 2.3.3. The FSM diagram for the S-MAC protocol process model is shown in Figure B.1. S-MAC state is initialized in the INIT state. Upon transition to the WAIT_INIT_SYNC state, a random timer is started to trigger node power-up. This random startup is to simulate nodes in a Mica2 network starting up over a 10-second period. Immediately after power-up, each node places its radio in receive mode and waits for a SYNC packet broadcast, which will dictate the frame timing for the virtual cluster of nodes within that node's one-hop neighborhood. If a node does not receive a SYNC packet

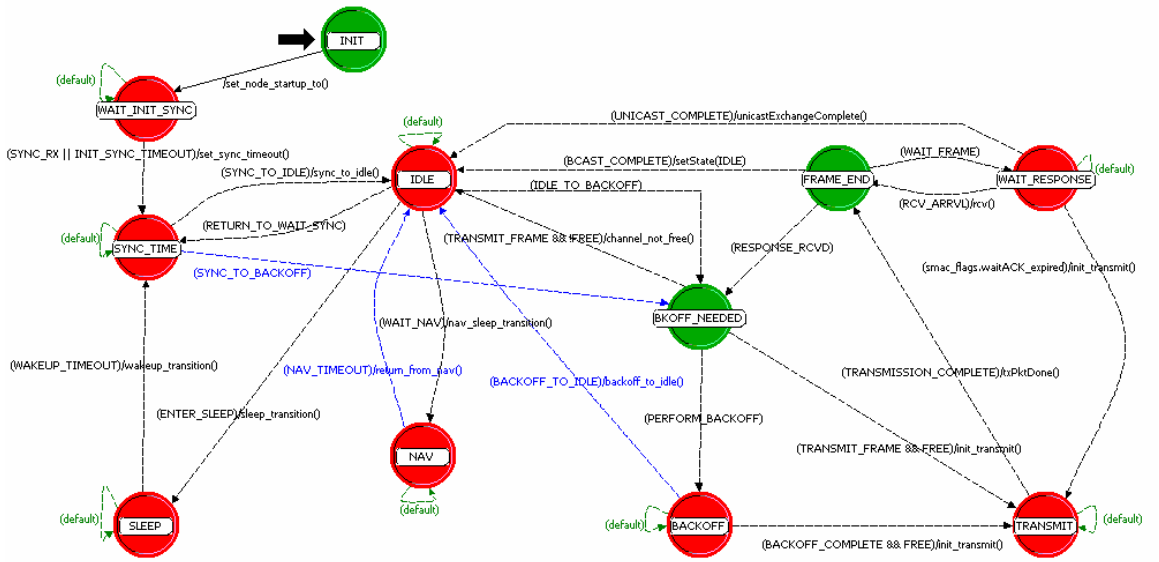


Figure B.1: S-MAC protocol process model.

within a certain period (the default is ten frames, or 13 seconds) it transmits its own SYNC packet, in which it advertises when it will next enter sleep mode [29]. The S-MAC frame duration and the duration of the active and sleep periods are set at compile time. When other nodes in a one-hop cluster receive a SYNC packet, they adjust their sleep schedules according to the value of the SYNC packet's $sleepTime$ value so that their schedule will coincide with that of the node that sent the SYNC. This ensures that the sleep and wake periods of the one-hop cluster of nodes is synchronized so they can exchange packets during the active period. Nodes periodically exchange SYNC packets to maintain this frame synchronization. Once nodes receive this initial SYNC, they enter the IDLE state for the remainder of their active periods. Just as in the TinyOS S-MAC implementation, a countdown timer with a 1-ms granularity triggers the transition from the IDLE state to SLEEP.

The most obvious difference between the WSN MAC FSMs, including S-MAC, and those for other CSMA MAC protocols is the SLEEP state. WSN MACs use this state to conserve energy by switching off sensor nodes' transceivers. The OPNET receiver module (labeled RX in the Mica2 node model shown in Figure 3.3) cannot be turned off like the actual Mica2 transceiver, so any packets received from the physical layer via the RX module while in the SLEEP state are

simply discarded. Upon transition to the SLEEP state, each node sets a timer-based interrupt to wake the node at the start of the next frame. A WSN node cannot transmit if the transceiver is in the SLEEP state. If the application layer sends a packet to the S-MAC process, the packet is buffered until the next active period. If the node already has a packet buffered, the incoming packet is rejected and must be either discarded or queued at the application layer. If a packet arrives from the application layer while a node is in SLEEP mode, the node will not immediately awaken and transmit because the other devices in the node's virtual cluster are asleep and would not receive the packet.

Upon wakeup, S-MAC nodes transition to the SYNC_TIME state. In this state, nodes that are scheduled to send a SYNC packet compete for the wireless channel to transmit the SYNC. A timer is used to trigger the transition from the SYNC_TIME state to the IDLE state, where routine network traffic is exchanged. Upon transition to IDLE, a node with a broadcast packet to transmit will transition to the BKOFF_NEEDED state to randomly select a backoff within S-MAC's 31-ms contention window, schedule an interrupt, and then transition to the BACKOFF state. If a packet is received while a node is in the BACKOFF state, the node transitions back to IDLE to process the packet, and then to the BKOFF_NEEDED state to select another backoff. When the random backoff timer expires, the node transitions to the TRANSMIT state to send the packet. Once transmission is complete, the node transitions through the FRAME_END state and back to IDLE.

If a node is sending a unicast packet instead of a broadcast packet, the above sequence is slightly different to account for the RTS/CTS/DATA/ACK sequence used for unicast transmissions. The sending node creates an RTS packet before transitioning to the BKOFF_NEEDED state and selecting a backoff. Once the RTS has been transmitted, the sending node transitions to the WAIT_RESPONSE state to wait for the CTS. The node then transmits the DATA packet and waits in the WAIT_RESPONSE state for the ACK. The backoff duration for CTS, unicast DATA, and ACK packets is 5 ms, which is the S-MAC short interframe space (SIFS) duration. A timeout prevents nodes from becoming stuck in the WAIT_RESPONSE state and triggers packet retransmission if an appropriate response is not received.

If two nodes have initiated a unicast exchange using an RTS, CTS handshake before the

sleep timer expires, the nodes will complete the unicast exchange before entering the SLEEP state. When a node overhears an RTS packet destined for another node, it waits for the CTS reply and then transitions to the NAV state for the duration of the exchange, which is indicated by a field in the RTS and CTS packets. Nodes can sleep during the NAV period if the time required to transition to sleep mode and back is shorter than the duration of the subsequent data packet. When the NAV has expired, nodes return to the IDLE state to await another packet. If a node's sleep timer expires while the node is in the NAV state, the node immediately transitions to the SLEEP state upon arrival at the IDLE state.

B.1.2 T-MAC Protocol Model

The state diagram for the T-MAC protocol process model is shown in Figure 3.4. The T-MAC protocol model behaves in a manner similar to the S-MAC model. The TinyOS T-MAC protocol implementation is, in fact, based largely on the S-MAC implementation. The primary difference between S-MAC and T-MAC is the adaptive timeout, which causes T-MAC nodes to transition to the SLEEP state when there is no more traffic in the network, instead of waiting for a timer as S-MAC does [7]. A description of T-MAC protocol functionality and the adaptive timeout mechanism is given in Section 2.3.3. Once in the IDLE state, each node sets its TTS timer to the duration of a T-MAC adaptive timeout. TTS is counted down at a granularity of one millisecond in the T-MAC event loop. In the OPNET process model, the expiration of the TTS countdown timer is triggered with a self-interrupt. Each transmitted or received packet causes TTS to be reset to the TA value by canceling and resetting the TTS interrupt. If the TTS timer expires while a node is in the IDLE state, the node sets a wakeup timer for the beginning of the next frame and enters SLEEP mode. Incoming packets are ignored until the wakeup timer has expired.

The remainder of the T-MAC states provide the same functionality as the same states in the S-MAC protocol process model.

B.1.3 B-MAC Protocol Model

The state diagram for the B-MAC protocol process model is shown in Figure B.2. The B-MAC protocol behaves differently than S-MAC and T-MAC. Recall from Section 2.3.3 that B-MAC nodes do not form virtual clusters. After initializing B-MAC variables and data structures in the INIT state, nodes transition to the SLEEP state and begin low-power listening (LPL) at a rate that is dictated at compile time [8]. After the pre-determined LPL poll interval, B-MAC nodes transition from the SLEEP state, through the WAKEUP state, to the POLL state, where they sample the wireless medium for preamble bytes. The preamble duration is longer than the polling interval to ensure that all of the transmitter's neighbors have the opportunity to conduct a poll during the preamble. If a node in the POLL state observes a preamble, the node transitions to the RX state to receive the incoming packet. If the received packet was a unicast packet for that node and if packet acknowledgments are activated, the receiver transitions through the CALC_BACKOFF state directly to the TX state to transmit the ACK packet (there is no backoff for unicast ACKs in B-MAC). Once the ACK packet has been transmitted, the node transitions through TX_COMPLETE back to the SLEEP state.

When a packet arrives at the MAC layer for transmission while a node is in the SLEEP state,

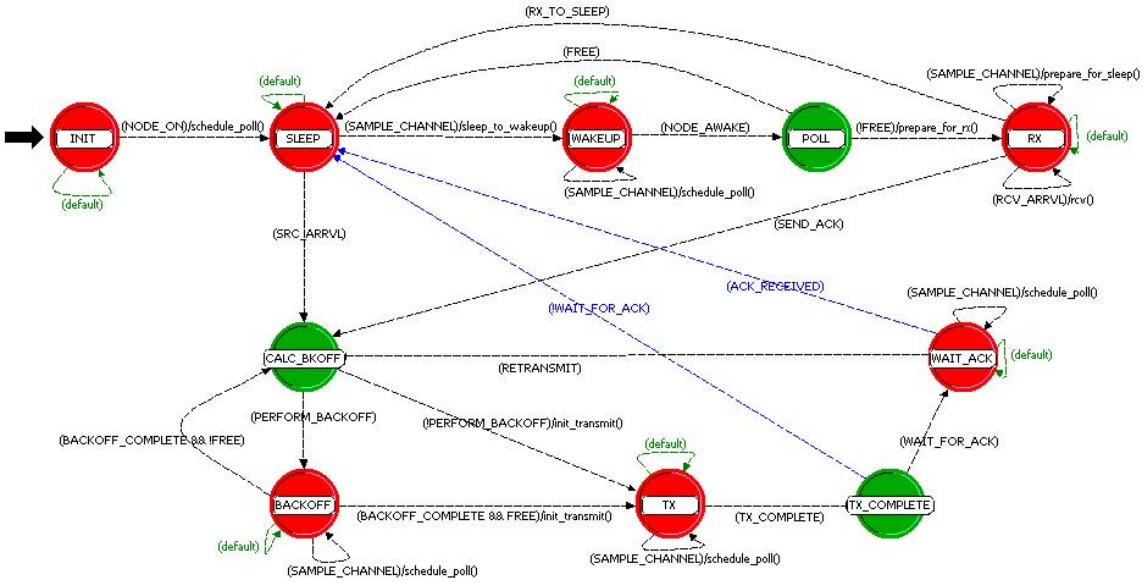


Figure B.2: B-MAC protocol process model.

the node immediately transitions to the CALC_BACKOFF state to set a random backoff within B-MAC's 15-ms contention window and then transitions to the BACKOFF state. If a node observes another node transmitting while in BACKOFF, the node receives the packet, then transitions back through the CALC_BACKOFF state to set a new backoff. Once the backoff timer has expired, the node transitions to the TX state and sends the packet. If the outgoing packet was unicast and acknowledgments are activated, the sending node transitions through the TX_COMPLETE state to the WAIT_ACK state to wait for the acknowledgment. If an ACK packet is not received within the specified time period (the S-MAC default is 7.5 ms), the node returns to the SLEEP state. Failure to receive an ACK is reported as a failed transmission to the application layer. If the outgoing packet was a broadcast packet, or if acknowledgments are not active, the transmitting node transitions from TX state through TX_COMPLETE state and then back to SLEEP.

B.2 Energy Consumption Model

Figure B.3 shows the FSM for the energy consumption process model, labeled "battery" in the WSN node model shown in Figure 3.3. In the INIT state, data structures and variables for tracking sensor node energy consumption are initialized. A statistics wire connects the MAC protocol process model, which dictates the state of the transceiver, to the battery process model. The energy consumption process transitions between the SLEEP, POLL, RX, and TX states based on signals on the statistics wire. The time that a node enters a particular state is logged and when the node transitions out of that state, the node is charged based on the amount of time spent and the energy consumption level for that state. The energy cost of transitions between states, which are listed in Table 2.1, are charged to the node upon transition from one state to another, and the duration of the state change is not counted in the energy consumption for the new state.

When the simulation ends, an END_SIM signal is passed to the energy consumption process model and the model transitions to the RECORD_STATS state. In this state, statistics are compiled to include the amount of time spent in each state, the total energy consumed during the simulation, and average the per-second energy consumption. This data is written to a file for later analysis.

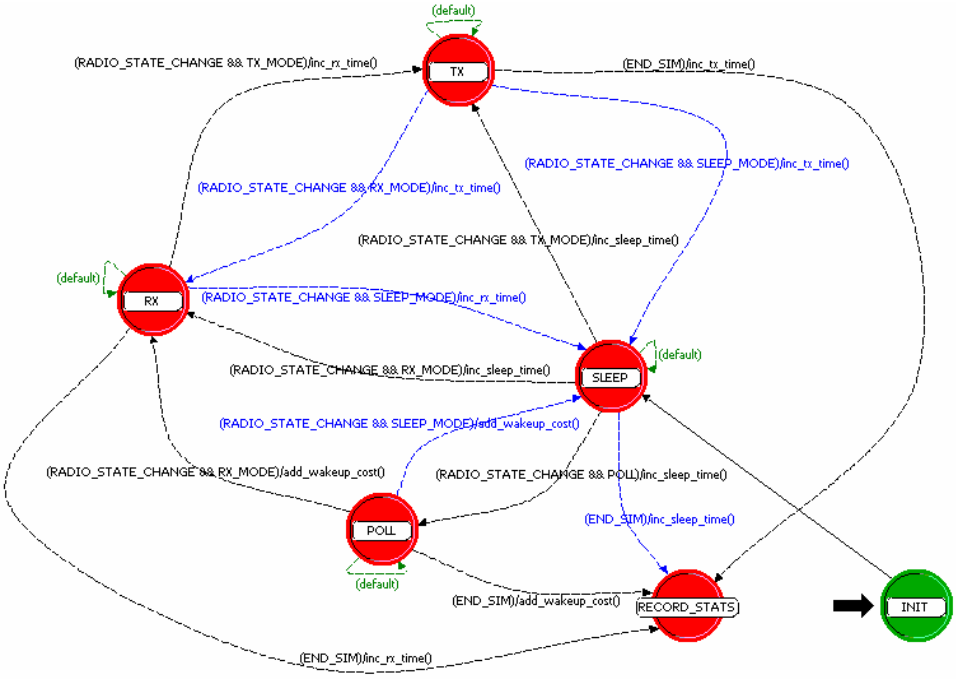


Figure B.3: Energy consumption process model.

Appendix C

Rate Limiting Threshold Analysis

Section 7.2 gives a brief description of the effect of the various threshold values on the functioning of the CARL mechanism. Thresholds for experiments described in Chapter 7 were selected based on simulations that were conducted to determine how various threshold values, particularly thresholds for increasing and decreasing rate-limiting levels, affect network lifetime and throughput. Simulations were then executed to determine how lifetime, throughput, and average packet delay are impacted by these threshold values. This appendix describes the results of those experiments on T-MAC and B-MAC in a simulated Mica2 network across offered loads.

C.1 T-MAC Threshold Analysis

Figures C.1, C.2, C.3, and C.4 show how network lifetime, throughput, and average packet delay are affected by manipulating $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ across offered loads in a T-MAC network. These graphs are based on five replications of 90 minutes of network time, with statistics collection beginning after 30 minutes. Network parameters for these simulations are the same as those described in Section 7.2.1. In each scenario the network is victim to the subtle unauthenticated broadcast attack described in Section 7.2.2. The graph in the upper-left of each figure gives

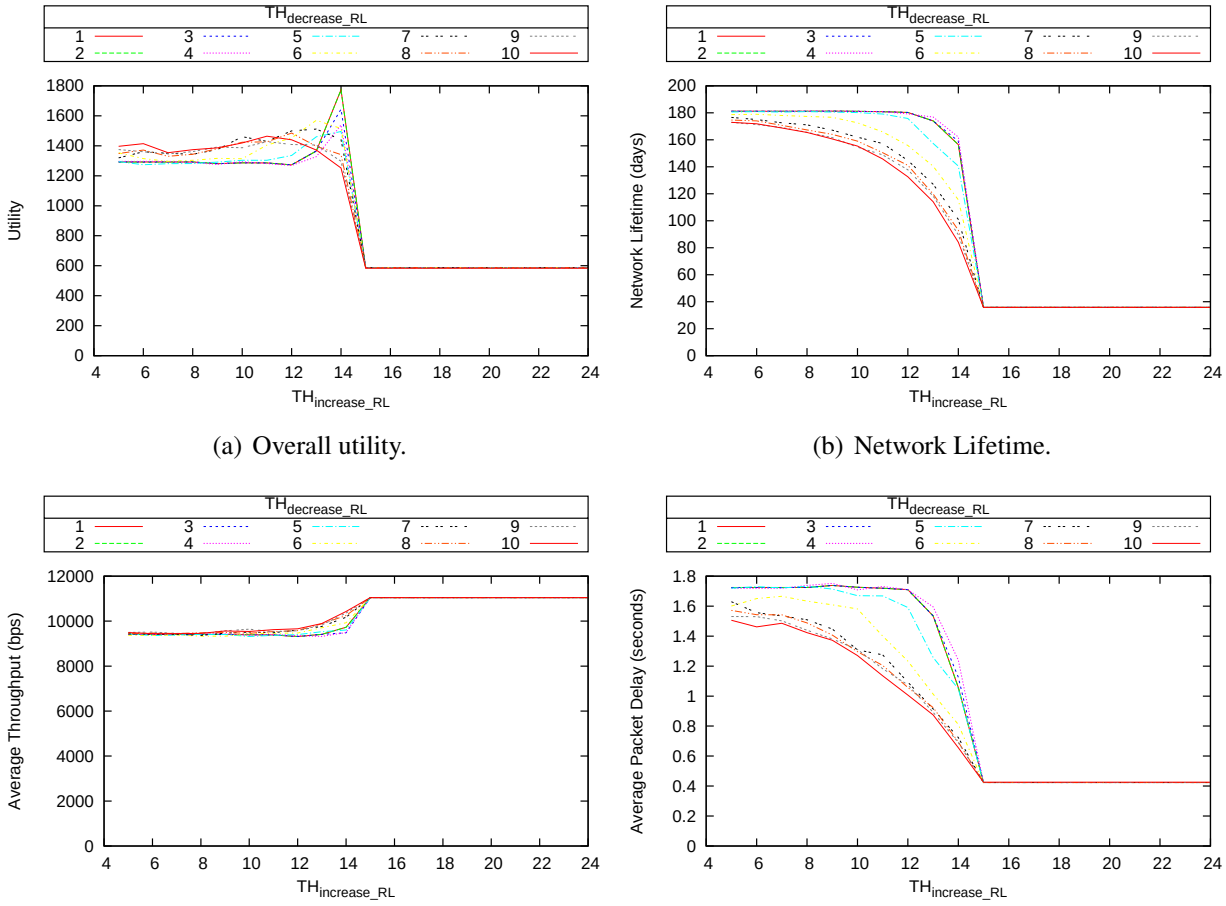
an overall utility value for each TH combination. Utilities are calculated as follows.

$$Utility = \frac{Lifetime^{w_{Lifetime}} \times Throughput^{w_{Throughput}}}{Latency^{w_{Latency}}}. \quad \text{(C.1)}$$

Weights for network lifetime ($w_{Lifetime}$), throughput ($w_{Throughput}$) and packet delay ($w_{Latency}$) were assigned based on the relative importance of each metric. In these figures, the weights are 1.5, 1.5, and 1 for lifetime, throughput, and delay, respectively.

These figures show that network lifetimes drop sharply for all offered loads as the value of $TH_{increase\_RL}$ increases beyond a value of 14. This is because thresholds beyond 14 are insufficient to increase rate-limiting levels beyond the minimum value for the unauthenticated broadcast attack used in these experiments. Packet delay is also reduced when $TH_{increase\_RL}$ values increase beyond 14 for the same reason. As shown in Figure C.1, at 1 pps, average throughput increases when $TH_{increase\_RL}$ increases beyond 14 because at the packet rates tested, throughput is not reduced at low rate limiting levels. This can be observed in Figure 7.5, where the impact of increasing rate-limiting levels on lifetime, throughput, and delay for the T-MAC protocol are examined. At lower packet rates, throughput is only slightly affected as the value of $TH_{increase\_RL}$ is increased because at these offered loads, throughput remains high even at high rate-limiting levels.

The values of $TH_{decrease\_RL}$ have a less dramatic impact on all performance metrics. These graphs show, however, that lower values of $TH_{decrease\_RL}$ (between 1 and 4), produce the best overall utilities across network offered loads. Higher thresholds for decreasing the rate-limiting level makes it easier for the level to be decreased. Decreasing the rate-limiting level causes a slight improvement in network throughput and packet delay, but it causes a more significant reduction in network lifetime, causing overall utility to suffer.



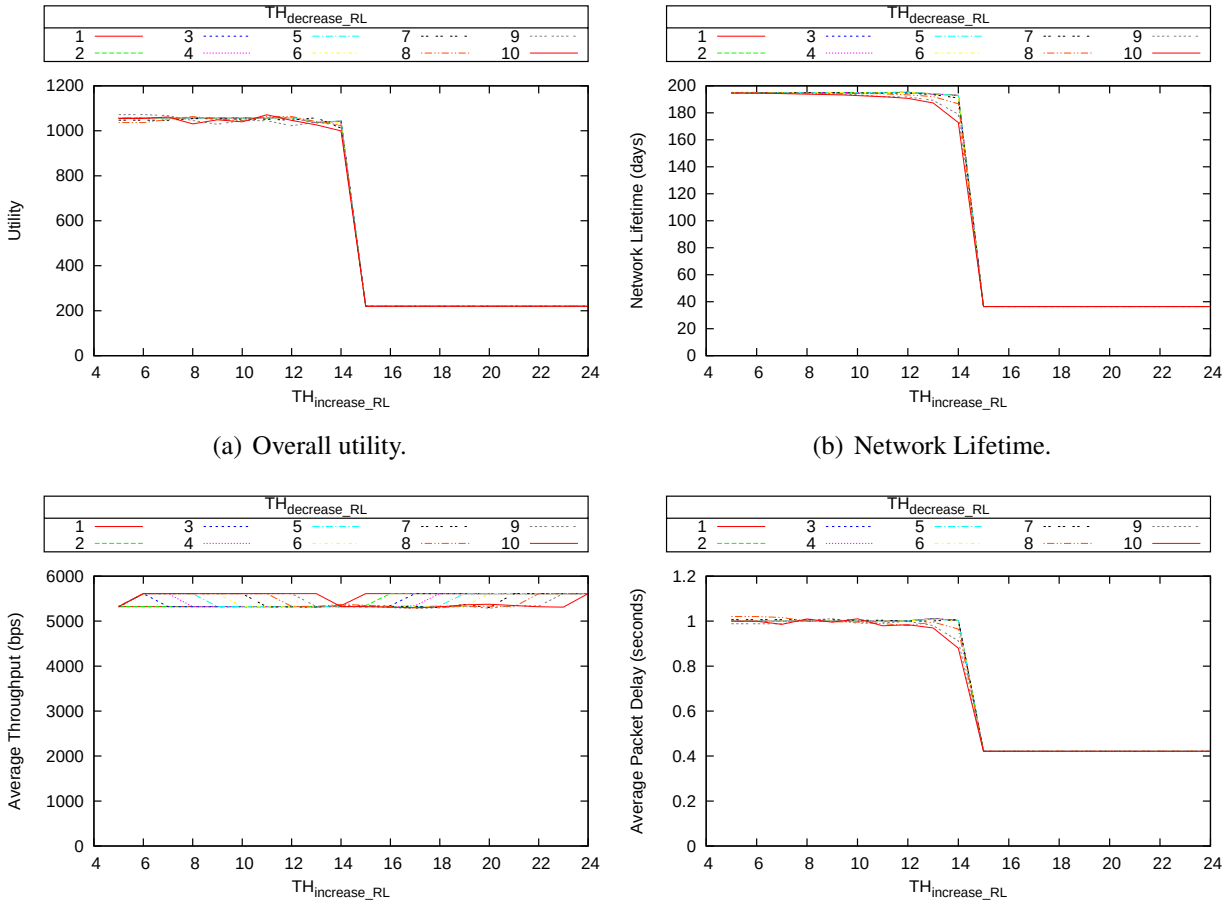
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.1: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for T-MAC protocol at 1 pps on the Mica2 platform.



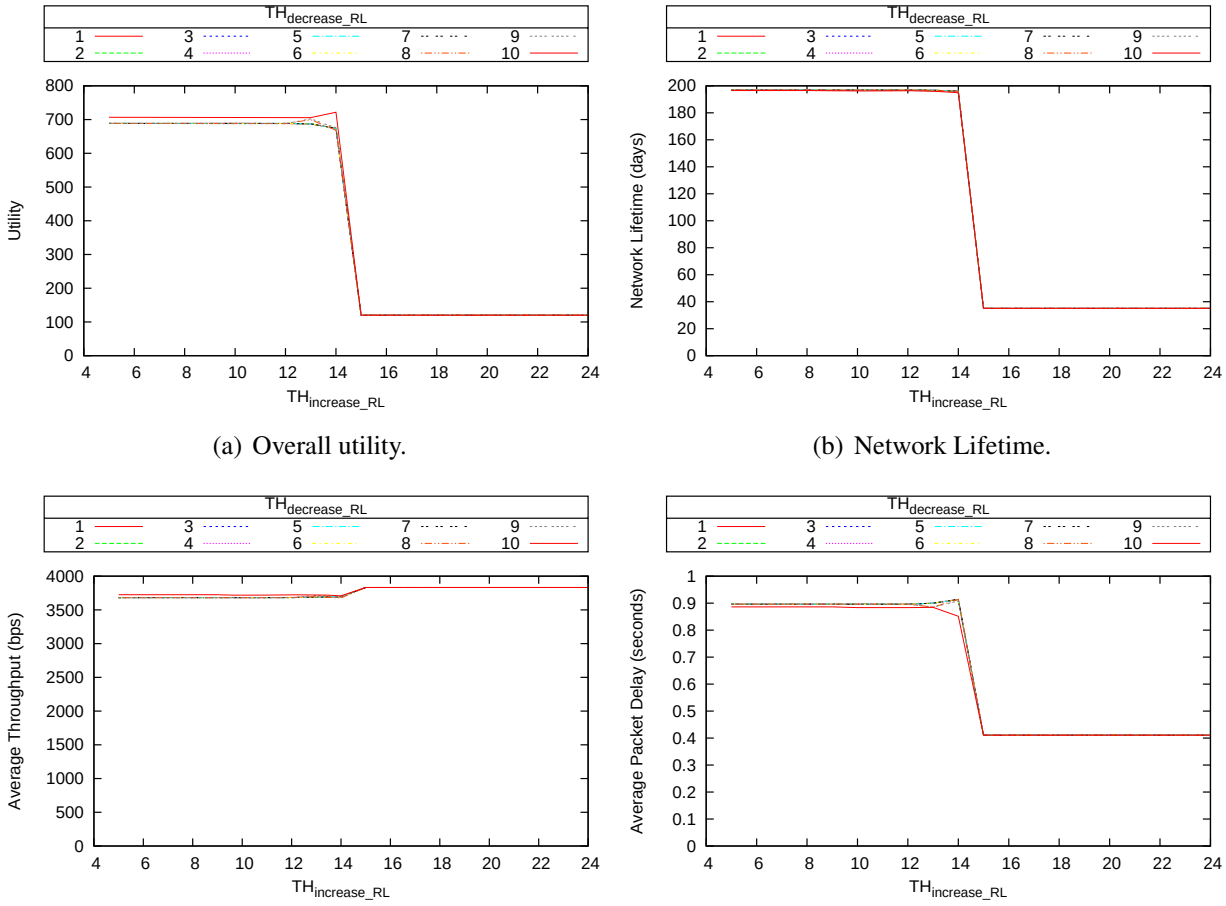
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.2: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for T-MAC protocol at 0.5 pps on the Mica2 platform.



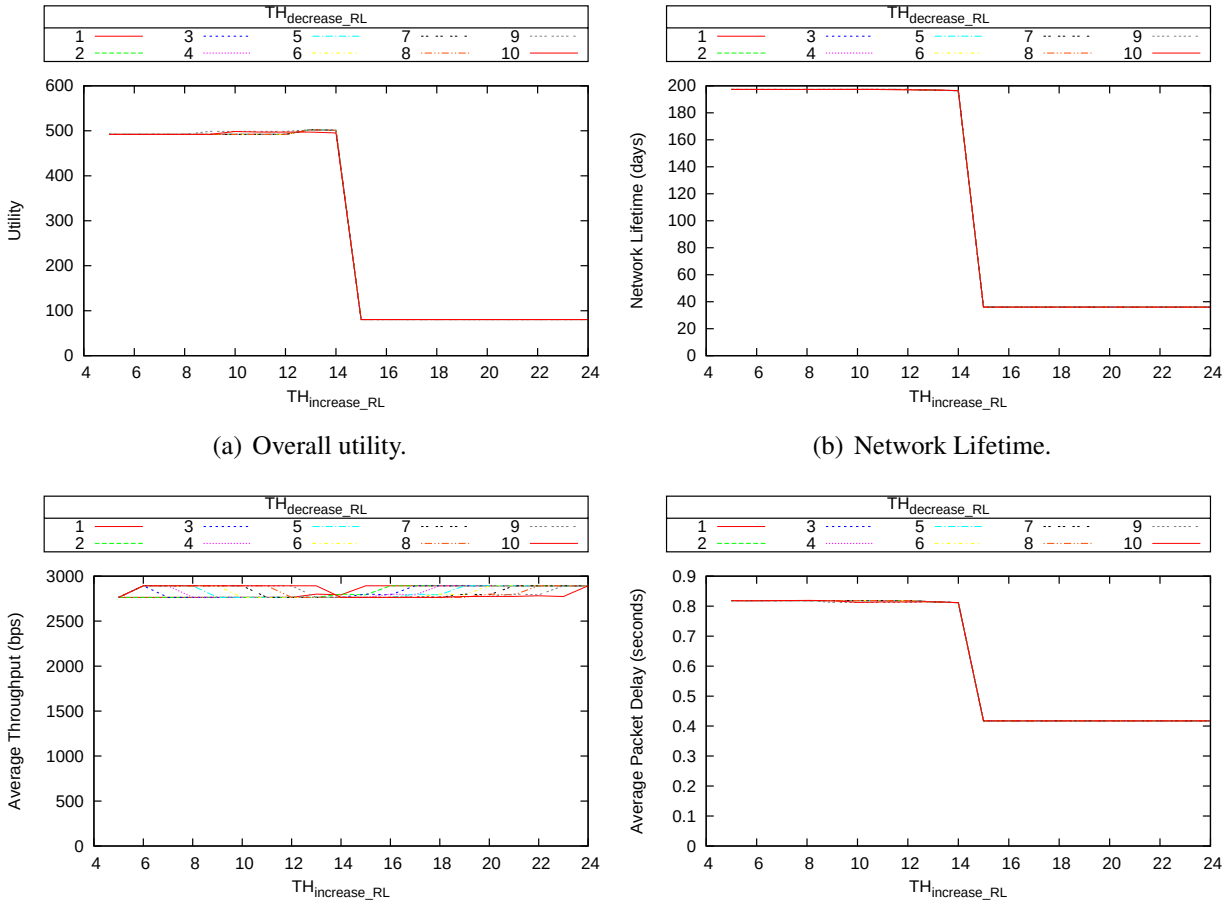
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.3: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for T-MAC protocol at 0.33 pps on the Mica2 platform.



(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

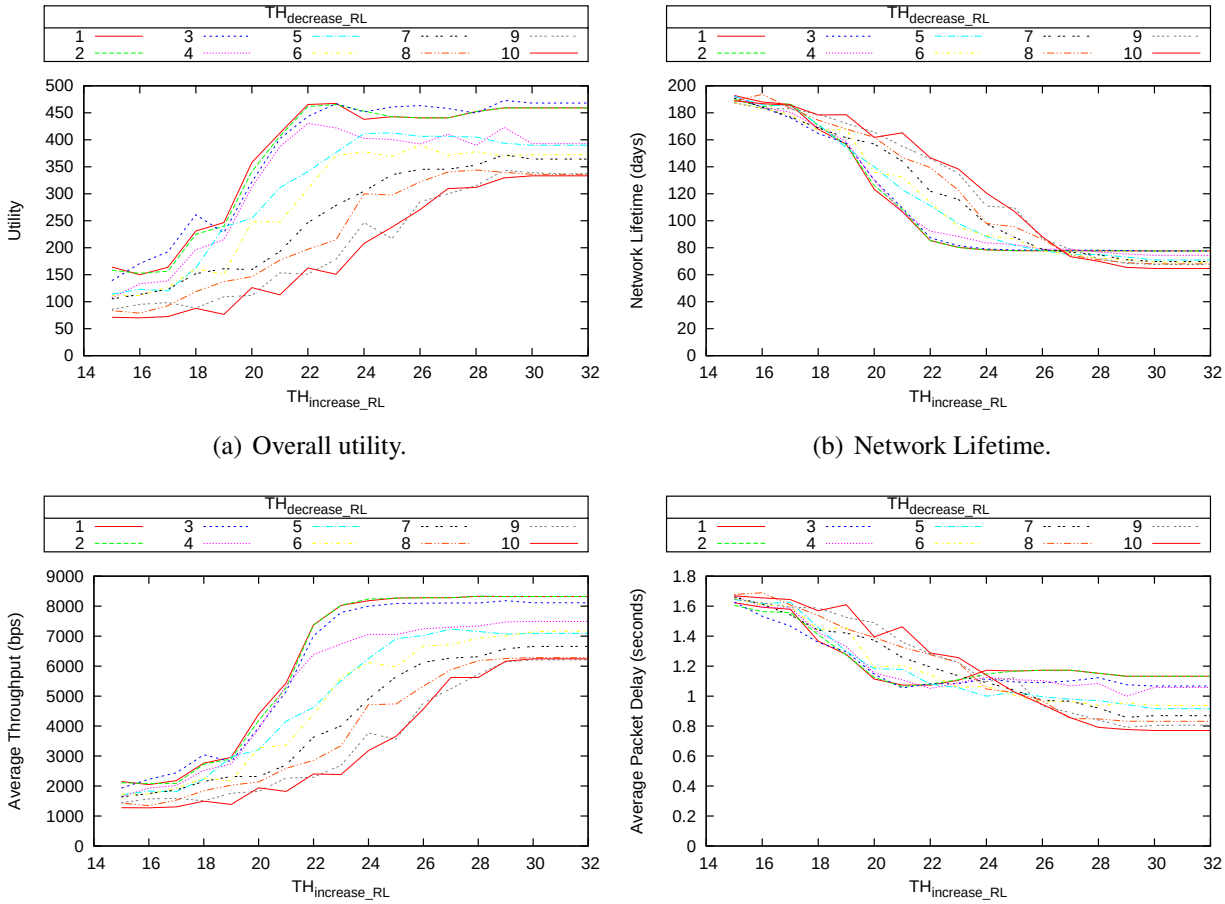
Figure C.4: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for T-MAC protocol at 0.25 pps on the Mica2 platform.

C.2 B-MAC Threshold Analysis

Figures C.5, C.6, C.7, and C.8 show how network lifetime, throughput, and average packet delay are affected by manipulating $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ across offered loads in a B-MAC network during a subtle unauthenticated broadcast attack.

As with the T-MAC results, there are obvious trends across offered loads. As the value of $TH_{increase\_RL}$ is increased, network lifetimes decrease because a higher ratio of malicious packets is required to increase the rate-limiting level beyond the level set when rate-limiting was initiated. Recall from Section 7.1.3 that when rate limiting is initiated in B-MAC, the rate-limiting level is set such that the rate-limited polling interval is closest to the average packet rate in the network. This policy is used because lower rate-limiting levels produce a severe reduction in network lifetime with little or no throughput advantage, as shown in Section 7.2.5. Lower values for $TH_{increase\_RL}$ cause rate-limiting levels to increase beyond the point at which throughput is preserved. The advantage to this is the increased network lifetime at these low thresholds. High values of $TH_{increase\_RL}$ prevent this increase in rate-limiting level across offered loads, thus providing a combination of higher throughput and lower packet delay, however with less-than-optimal network lifetime. Overall utilities across offered loads favor high values for $TH_{increase\_RL}$.

Just as with T-MAC, lower values of $TH_{decrease\_RL}$ result in less fluctuation in rate-limiting levels, causing longer network lifetime, higher throughput, and lower packet delay when the value of $TH_{decrease\_RL}$ is high (25 and above) at high packet rates (1 pps and 0.5 pps). At lower packet rates, lower values of $TH_{decrease\_RL}$ provide the best results in all cases. These results suggest that it is important to consider the expected network offered load and, if offered loads will be high, to select thresholds for increasing and decreasing rate-limiting levels such that they complement each other.



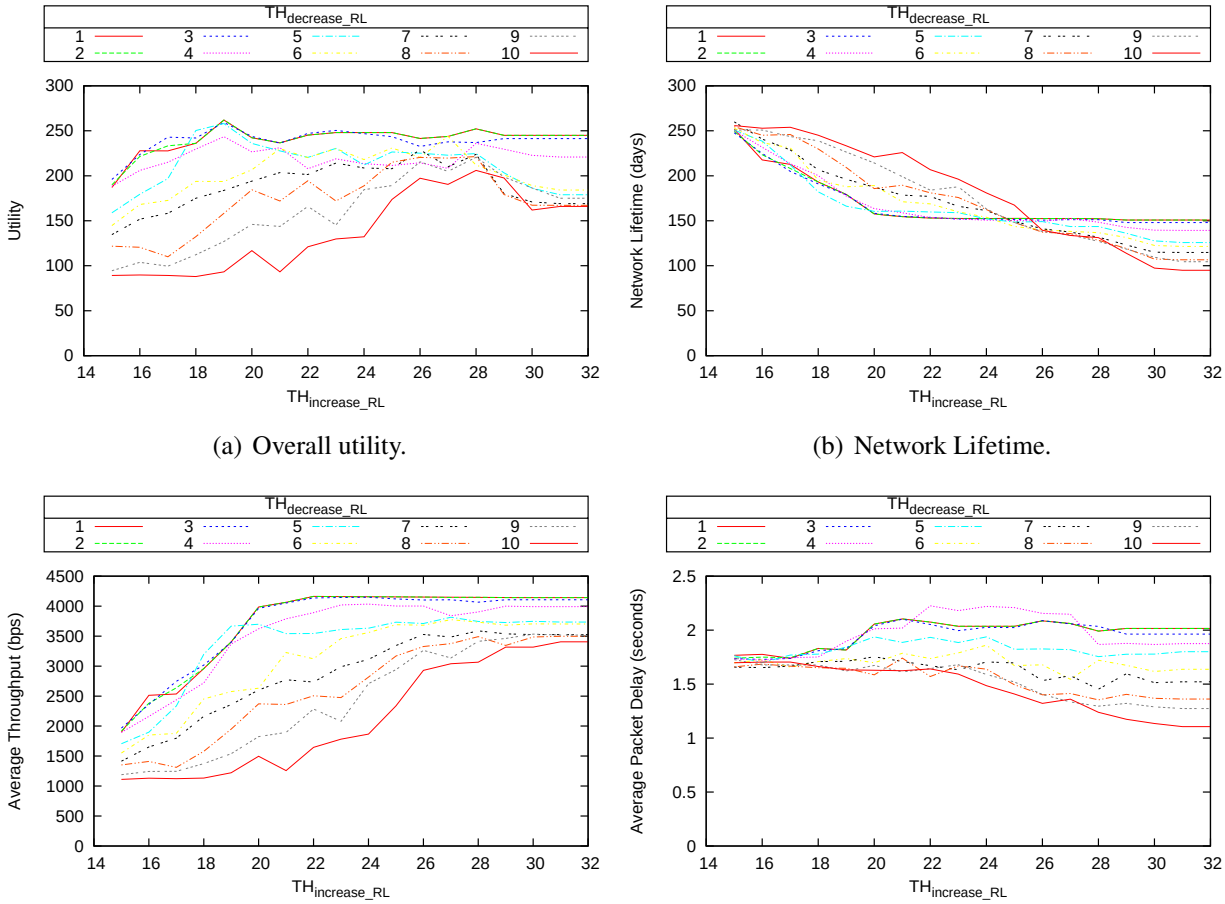
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.5: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for B-MAC protocol at 1 pps on the Mica2 platform.



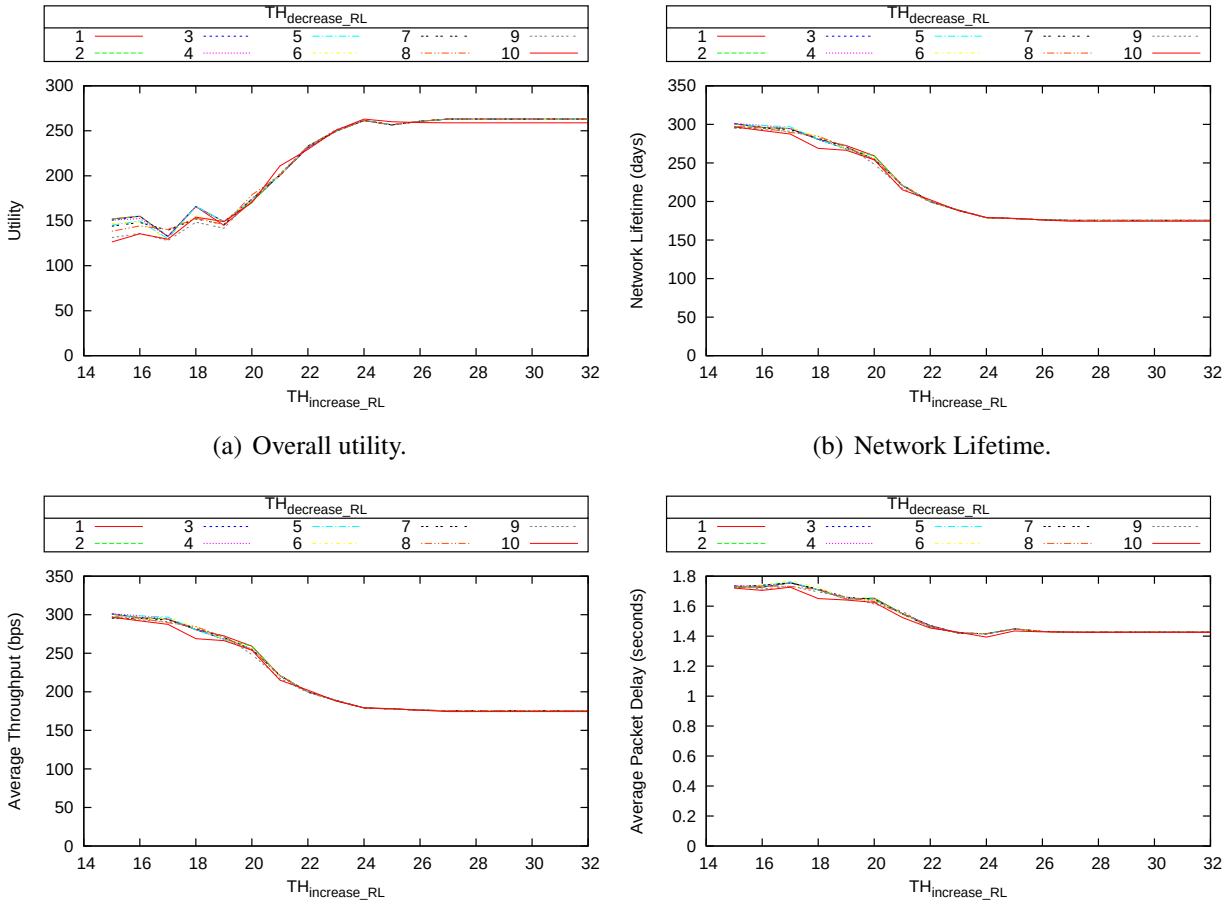
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.6: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for B-MAC protocol at 0.5 pps on the Mica2 platform.



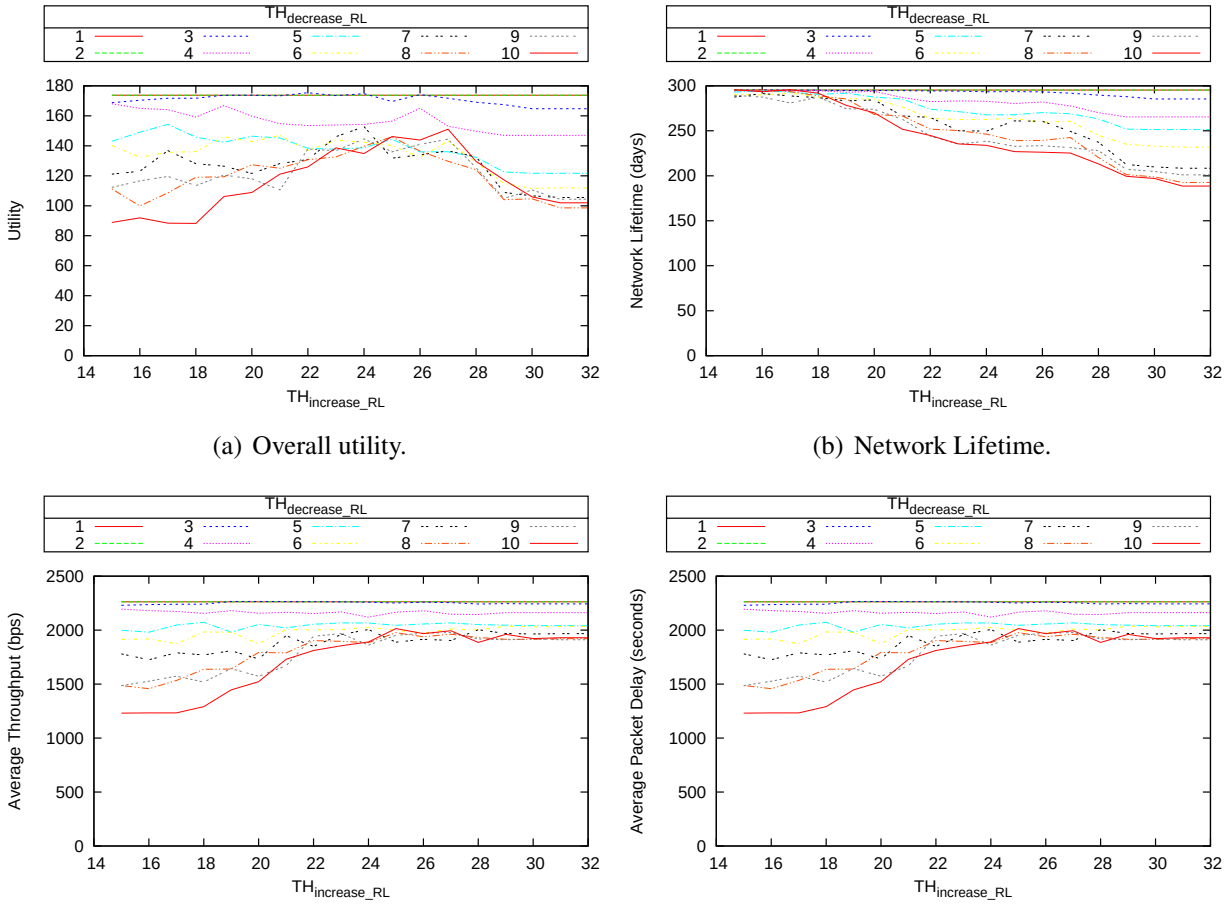
(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.7: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for B-MAC protocol at 0.33 pps on the Mica2 platform.



(a) Overall utility.

(b) Network Lifetime.

(c) Network Throughput.

(d) Packet Delay.

Figure C.8: Overall utility, network lifetime, throughput, and delay by values of $TH_{increase\_RL}$ and $TH_{decrease\_RL}$ for B-MAC protocol at 0.25 pps on the Mica2 platform.