CHALLENGES AND SOLUTIONS FOR INTRUSION DETECTION IN WIRELESS MESH NETWORKS

A Dissertation

by

AMIN HASSANZADEH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Radu Stoleru
Committee Members,	Riccardo Bettati
	Anxiao (Andrew) Jiang
	I-Hong Hou
Head of Department,	Nancy Amato

May 2014

Major Subject: Computer Engineering

ABSTRACT

The problem of intrusion detection in wireless mesh networks (WMN) is challenging, primarily because of lack of single vantage points where traffic can be analyzed and the limited resources available to participating nodes. Although the problem has received some attention from the research community, little is known about the tradeoffs among different objectives, such as high network performance, low energy consumption, and high security effectiveness. In this research, we show how accurate intrusion detection can be achieved in such resource constrained environments. The major challenges that hinder the performance of intrusion detection systems (IDS) in WMN are resources (e.g., *energy, processing*, and *storage* capabilities) accompanied by the adhoc-dynamic communication flows.

In light of these challenges, we classify the proposed solutions into four classes: 1) Resourceless Traffic Aware (RL-TW) IDS, 2) Resourceless Traffic Agnostic (RL-TG) IDS, 3) Resourceful Traffic Agnostic (RF-TG) IDS, and 4) Resourceful Traffic Aware (RF-TW) IDS. To achieve a desirable level of intrusion detection in WMN, we propose a research program encompassing five thrusts. First we show how traffic-awareness helps IDS solutions achieving high detection rates in resource-constrained WMN. Next, we propose two RL-TG (i.e., cooperative and non-cooperative) IDS solutions that can optimally monitor the entire WMN traffic without relying on WMN traffic information. The third (RF-TG) and fourth (RF-TW) IDS solutions propose energy-efficient monitoring mechanisms for intrusion detection in battery-powered WMN for traffic-agnostic and traffic-aware scenarios, respectively. We then investigate the Attack and Fault Tolerance of our proposed solutions and finally enumerate potential improvements and future works for our proposed solutions.

To My Beloved and Supportive Family:

My Parents,

My Wife,

My Brothers and Sister.

ACKNOWLEDGEMENTS

I wish to thank my very supportive advisor Dr. Radu Stoleru, without whom this work would have been an unrealized dream. His methods of intuitive thinking, research methodology and outlook on life will be something I will treasure and learn from in the years to come. I would like to thank him for accepting me into the LENSS lab, which provided me with the necessary exposure to the field of wireless networks security.

I would like to thank Zhaoyan (Brooks) Xu, Ala Altaweel, Dr. Michalis Polychronakis and Dr. Guofei Gu for their collaborations in this research. I would also like to thank Dr. Riccardo Bettati, Dr. Anxiao (Andrew) Jiang and Dr. I-Hong Hou for being part of my committee and their positive and helpful feedback. I also wish to thank all the (former and current) members of the Laboratory for Embedded and Networked Sensor Systems (LENSS) for their support and companionship during the course of my study. This work would not have been possible without their constructive criticism.

I wish to thank my parents, wife, brothers and sister for their constant love, support and encouragement. Last, but not the least, I would like to acknowledge my compatriot friends for making my stay in College Station memorable and fun-filled.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

Wireless Mesh Networks (WMN) are self-managing networks that provide Internet, intranet, and other services to mobile and fixed clients using a multi-hop multi-path wireless infrastructure consisting of mesh nodes [2,4,40]. The number of deployments of cost-effective mesh networks is continuously increasing as they are suitable for many application domains such as disaster response [5,16–18,25,28,63,80], rural IT services [1,6,10,44,45], environmental monitoring [27,92,93] and many others, as surveyed in [2].

Because of the intrinsic sharing of the wireless medium and the emerging information security threats, security has become one of the most critical issues WMN deployments face today. Although *Intrusion prevention* methodologies, e.g., cryptography, are known as the first line of defense in wireless networking, this may not be enough in mission critical scenarios that require strictly secure communication. It has been argued [97] that regardless of the number of intrusion prevention strategies used in a network, some vulnerabilities can always be found to allow intruders passing the first line of defense. To address the issues of intrusion prevention, one of the most effective ideas proposed was to add layers of additional security tools, e,g., *intrusion detection systems* (IDS), that take appropriate actions when the network is perceived to be under attack.

Simply adopting IDS from wired networks is challenging because WMN lack:

- **Single Vantage Points** where traffic can be analyzed, which is typical in wired networks (e.g., a gateway or router in a corporate network).

- **Hardware Resources** available to wired networks, e,g., *processing power*, *storage* and practically unlimited *energy* for powering WMN devices.

Due to the lack of concentration points in WMN where network traffic can be analyzed, research community has proposed decentralized monitoring mechanisms [42, 62, 69, 75] for intrusion detection in WMN. Decentralized (also known as distributed) solutions have been investigated mainly in the context of MANET and sensor networks [21,47,67,84,85,98]. There, IDS were completely decentralized, and an intrusion detection agent was placed on each node [97]. These solutions were very inefficient since nodes in the network would execute intrusion detection in a redundant manner (e.g., a multi-hop stream was analyzed multiple times) thus consuming both hardware resources (that could be allocated to other network functions) and energy. Additionally, the research [42, 58] has recently shown that the number of attacks detectable by an intrusion detection agent placed on each node is limited by the amount of resources available on the node (resulting in high false negative rates). Therefore, a significant number of intrusion detection mechanisms proposed for WMN have only considered a specific type of service/attack in a particular WMN application (e.g., Wormhole and Grayhole attacks) and proposed a detection technique/rule for it with respect to resource limitations [22, 29, 48, 60, 62, 69, 73, 74, 90]. The other few efforts in intrusion detection for WMN, regardless of attack types, aim at finding an *optimal monitoring mechanism* (e.g., IDS node placement or IDS rule assignment) that is practical based on WMN characteristics [30, 42, 58, 75]. Our motivating scenario for this research is the second group of IDS solutions and their design and implementation challenges.

Among the research efforts on the optimal monitoring mechanisms for intrusion detection in WMN, OpenLIDS [42] proposes a *Lightweight* detection engine that im-

poses less computational load than off-the-shelf IDS (e.g., Snort [78] and Bro [65]) when executing on WMN nodes. This solutions is a distributed solution that requires every WMN node to run the proposed lightweight IDS to monitor the entire network. However, when compared to off-the-shelf IDS, OpenLIDS has even higher false negative rates because fewer IDS functions are implemented in the detection engine to conserve more processing and storage resources. Hence, we believe with ever increasing security threats against such networks, lightweight IDS solutions are not suitable for intrusion detection in WMN.

The identified inefficiencies have triggered significant research on *optimal monitoring* for intrusion detection in WMN [19, 30, 46, 75, 76, 83]. The optimal monitoring has been typically solved by selecting a few nodes (called *monitoring nodes*) that execute the same set of IDS rules/functions but each of them is responsible for a distinct part of the network. The research has shown that these solutions are only suitable for *resourceful* mesh networks in which WMN nodes have sufficient resources for executing a complete set of IDS functions. Otherwise, the IDS will suffer from high false negative rates because the hardware resources does not allow node to perform a full IDS [42, 58]. We believe that state-of-the-art solutions proposed for optimal monitoring in WMN do not consider:

1. *Detecting link-based attacks* since they are formulated to cover WMN nodes which is shown [37] to leave some communication links uncovered and consequently some link-based attacks undetected.

2. *Energy efficiency* in battery-powered WMN [7, 16, 57] where executing IDS tools imposes higher energy consumption rates to WMN devices [37].

These two inefficiencies in optimal monitoring solutions are our motivating scenarios in some of research thrusts in this dissertation.

Recently, in order to decrease the number of nodes responsible for monitoring network traffic and consequently to reduce the total resource consumption for intrusion detection purposes, it was proposed that knowledge about network traffic (i.e., *traffic-awareness* [59]) be used for optimal monitoring for intrusion detection [30, 70]. The traffic awareness is particularly helpful in networks with significant constraints on hardware resources (designated herein as *resourceless*). Some WMN may fall in this category and can benefit from such solutions while other WMN have wireless nodes with more hardware capabilities (designated herein as resourceful), that can be dedicated to performing full IDS functions [16, 75, 92]. In this research, we show that *traffic-aware* IDS solutions proposed for wired network [70] are infeasible for resource-constrained WMN, however, inspired by the idea proposed in [70], we show how the idea can be modified and practically used in real-world resource-constrained WMN. We hypothesize that traffic awareness can also be helpful for resourceful WMN. More precisely, we are motivated by the facts that traffic-awareness can be applied to:

1. *Resourceless WMN* where nodes are not able to perform a complete set of IDS functions, but distributing IDS functions efficiently along traffic paths will increase intrusion detection rates while ensuring that nodes are not overloaded by IDS function [39].

2. *Resourceful WMN* where number of monitoring nodes, being able to perform full IDS, decreases because only few traffic paths (consequently few communication links) have to be monitored [31].

Applying traffic-aware mechanisms to the state-of-the-art IDS solutions in WMN is another motivating scenario in this research.

Research has shown [14,16,64] that even static WMN topology and routing paths are subject to change due to: a) *link-quality* variations, especially in outdoor deployments, caused by weather, noise and other radio signals, etc.; b) *mobility of clients* and their requested services that result in changes of WMN routing paths; c) *node failure* (e.g., running out of power) or node replacement (e.g., administrative reasons) during network lifetime. Thus, traffic-awareness might be a strong assumption for many WMN applications where traffic paths change very often and consequently degrade the performance of the IDS solution. Therefore, the traffic knowledge has to be very accurate and up-to-date in traffic-aware solutions, which is not always feasible. In fact, a *traffic-agnostic* IDS solution that monitors all communication links instead of only few paths is more reliable and also applicable to all types of WMN, at the price, however, of putting IDS load on all WMN nodes and consuming more resources for intrusion detection.

As a traffic-agnostic solution for resource-constrained networks, *cooperative IDS* have been investigated mainly in the context of ad hoc and sensor networks [41, 49, 50, 52, 54, 62, 81, 84]. In these solutions every node is assigned a few IDS functions to detect attacks based on local observation. A cooperative IDS engine is then employed for detecting more attacks, based on neighbor information [62,94]. Cooperative mechanisms incur high communication overhead, caused by message exchange required for intrusion detection, and high detection latency since some of decisions are made only after receiving other nodes' reports. Therefore, although cooperative IDS have proven viable for low-traffic networks, e.g., sensor networks, they are not practical (i.e., degrades the network performance and delays intrusion response) in most of WMN applications [1,2,23,59]. Although cooperative IDS has received some attention from the research community, little is known about the tradeoffs among different objectives, such as high detection rate, low resource consumption and detec-

tion latency. Motivated by the fact that cooperative IDS are not practical solutions for most of WMN applications, another research thrust in this dissertation concentrates on proposing *non-cooperative IDS* where each node, depending on its available resources, is assigned a subset of IDS functions, i.e., a customized IDS configuration, and investigates the entire network traffic on the set of communication links it can monitor (i.e., in its coverage area). This solution does not require message exchange in order to make intrusion detection decision and eliminates communication overhead and detection delay.

Finally, although intrusion detection mechanism in WMN have received considerable focus, little attention has been paid to attacks-and-failures against/of IDS nodes. Undoubtedly, when an IDS node is compromised or faulty, it is unable to participate in intrusion detection process, thus, the intrusion detection rate will decrease and some malicious activities will remain undetected (i.e., high false negative rates). Therefore, as the last research truth in this dissertation, we investigates the attack-and-fault tolerance of IDS solutions we propose.

1.2 Dissertation Statement

Intrusion detection in WMN is challenging because a set of requirements and constraints that need to be considered when proposing an optimal monitoring mechanism for IDS. Since the network characteristics vary from one to another WMN application, an IDS designed for a particular WMN has to consider all WMN characteristics:

- processing power and storage of the nodes
- energy constraints in WMN
- networking services provided by the WMN

- administrative knowledge (e.g., user traffic patterns)
- vulnerabilities and potential threats

Taking into consideration that each of these characteristics has a significant impact on both regular networking functionality and intrusion detection performance, it is our belief that the all IDS design and implementation requirements related to the characteristics above have to be met in optimal monitoring mechanisms proposed for intrusion detection in WMN.

Our thesis is that the following goals can be achieved by an intrusion detection system in WMN:

- **Practical:** An intrusion detection mechanism designed for a specific WMN application should consider the amount of available *resources* and also the importance of networking *services* currently available in the WMN. It is shown in recent research that most of the proposed IDS solutions cannot be practically employed by mesh networks due to resource limitations (e.g., mesh node runs out of memory and crashes after running IDS). Additionally, a desired IDS always considers all active services and their potential threats instead of focusing on few specific attacks (i.e., leaved many types of attacks undetected resulting in high false negative rates). Finally, since WMN is known as a cost-effective easy-to-deploy networking solution, a practical IDS solution for WMN must also be an easy-to-deploy mechanism (e.g., no need for extra hardware).
- **Efficient:** Similar to other processes, intrusion detection process impose *memory and CPU loads* to WMN nodes. In addition, research has shown that some WMN hardware consume more electrical current *(energy)* as CPU load increases. Thus, an efficient intrusion detection imposes minimum amount of

memory and CPU loads (consequently minimum energy consumption) to WMN nodes.

- **Accurate:** The most important evaluation metrics for intrusion detection systems are detection rates and false alarm rates. When designing an IDS for a WMN application, the *attacker model* and *detection rules* must be accurately defined and evaluated to achieve maximum detection rate and reduce false alarms.

- **Scalable:** Some IDS mechanisms perform complex optimization algorithms to solve the optimal monitoring problem and optimal IDS rule assignment. Due to limited amount of processing resources on WMN nodes, these solutions require a central and computationally powerful node (e.g., base station) to execute *complex algorithms*. In addition, in order to find optimal solutions for a given network, the central nodes needs to collect nodes' information. The communication overhead imposed by the message exchange between WMN nodes and the central node increases as network size increases. Hence, it is very important to propose solutions that impose less communication overheads or can be implemented in a *distributed* mode.

- **Reliable:** A severe and challenging attack against a WMN network could be an *attack against IDS nodes* or an attack in which the attacker is already aware of the intrusion detection mechanism. Unlike the considerable effort in designing IDS solutions for WMN from the research community, little effort has been dedicated to the attack-and-fault tolerance of IDS themselves.

Table 1.1: Taxonomy for Traffic and Resource Aware Intrusion Detection in WMN

		Hardware Resources	
		Resourceless	**Resourceful**
Traffic Awareness	**Traffic Agnostic**	[33, 34], RAPID [36]	EEMON [31, 37]
	Traffic Aware	PRIDE [38, 39]	TRAIN [31]

1.3 Main Contributions

In light of the aforementioned objectives, the contributions of this dissertation are the following (as listed in Table 1.1):

- **PRIDE: Traffic Aware and Resourceless IDS** [38, 39]: The fact that WMN are resource constrained poses significant challenges for intrusion detection. The main idea in PRIDE is to use the knowledge a security administrator has about the WMN traffic to distribute IDS functions more efficiently. More precisely, a security administrator, knowing the routing paths of the traffic in the WMN, would employ a traffic-aware framework that optimally places IDS functions on the nodes along the routing paths. The information about the busiest and most frequently used paths in the WMN is obtained from routing algorithms (e.g., OLSR) and network monitoring tools (e.g., tcpdump). PRIDE has no detection latency in making the intrusion detection decision. In this solution, *each node along a routing path, runs a distinct and customized IDS*. This *customized* IDS (technically a subset of IDS functions) allows resource conservation. The combination of *distinct* IDS along the path allows for a complete set of IDS functions to be applied to the entire network traffic.

- **RAPID: Traffic Agnostic and Resourceless IDS** [36]: This research thrust is motivated by the fact that in many WMN applications traffic paths

change very often, which consequently degrades the performance of traffic-aware IDS solutions. For example, routing paths in large scale WMN that provide networking services for mobile clients are subject to change due to client mobility. Additionally, WMN topology, especially in outdoor deployments, may change due to node failures or drastic link-quality changes. Hence, the traffic knowledge has to be very accurate and up-to-date in traffic-aware solutions, which is not always feasible. In RAPID, we propose a traffic-agnostic intrusion detection mechanism for resource-constrained WMN that monitors all communication links, instead of only few paths. Each node, depending on its available resources, is assigned a subset of IDS functions and investigates the entire network traffic on the set of communication *links* it can monitor (i.e., in its coverage area). This customized IDS allows resource conservation on resource-constrained WMN nodes and also increases the probability of monitoring a WMN link with multiple distinct IDS functions activated on all WMN nodes that can monitor the link. It is worth mentioning that for a given network size, the complexity of traffic-agnostic solution is larger than traffic-aware solution as it needs to find optimal IDS function distribution for all nodes.

- **Cooperative IDS: An Optimal Monitoring Mechanism for Cooperative IDS** [33, 34]: When WMN nodes are extremely resource-constrained and the network density is low, RAPID intrusion detection rate degrades because nodes can only execute few IDS functions and communication links are poorly covered. Cooperative solutions propose information exchange among WMN nodes (e.g., their local observation) in order to achieve higher detection rates. This solution is practical for low traffic WMN where the communication overhead caused by intrusion detection message exchange does not influence the

network performance. In this research thrust, we propose a cooperative solution in which a base station which has knowledge about network (e.g., node resources, locations, etc.) and security requirements (e.g., maximum permissible *delay* in reporting an event, minimum network *coverage*), computes the optimal distribution of roles specific to cooperative IDS. Our proposed solution allows execution of sophisticated algorithms that *optimize multiple objectives* related to network performance and security effectiveness.

- **EEMON: Energy Efficient Traffic Agnostic and Resourceful IDS** [31, 37]: This research considers battery-powered WMN where resourceful nodes are able to perform full IDS configurations. Thus, the optimal monitoring mechanism for intrusion detection is to select a few monitoring nodes that can monitor the entire network traffic. However, despite the attention energy efficient operation in WMN has received there is no provision in the 802.11s standard for power saving mode operation. This led to the absence of mesh node hardware that operates in a power saving mode. Given the urgent need for energy saving, most of the solutions propose duty-cycling which has adverse effects on the IDS operation where monitoring nodes are required to be on/awake at all times. Consequently, the research challenge/problem we address in this work is how to reconcile energy efficient operation, which requires nodes to be asleep as much as possible, with an effective intrusion detection, which requires nodes to be awake, to monitor traffic.

- **TRAIN: Traffic Aware and Resourceful IDS** [31]: As the last class of IDS we investigate in this research, TRAIN proposes a traffic-aware monitoring mechanism for battery-powered mesh networks where nodes are resourceful and able to execute full IDS. In fact, TRAIN studies the effect of traffic-awareness

on EEMON in which, instead of monitoring all communication links, the monitoring mechanism has to monitor only a few links in the WMN (i.e., those located on traffic paths).

- **AFT-IDS: Attack and Fault Tolerant IDS**: Although the design and implementation of specific intrusion detection mechanisms have received considerable attention, little effort has been dedicated to the attack-and-fault tolerance of IDS themselves. In this research thrust, we investigate the attack-and-fault tolerance of our all intrusion detection systems we have designed and implemented for wireless mesh networks (as listed in Table 1.1). We first survey a series of administrative mechanisms for attack-and-fault tolerant (AFT) IDS design and proposes a classification for all AFT mechanisms and then concentrates on *preventive* solutions. These solutions use redundant IDS nodes to maintain high IDS availability ratio after IDS compromise/failure times. Finally, we propose redesigned IDS solutions that are attack and fault tolerant and then show that these mechanisms, at the price of higher resource consumption, increase the attack/fault tolerance level.

1.4 Organization

This dissertation is organized in nine sections. The current section motivates our work and states the contributions of the research. In Section 2 we review state-of-the-art solutions and investigate their practicality and effectiveness in different WMN applications. Section 3 presents the system and security (attacker) models considered in the entire research for different intrusion detection mechanisms. From Section 4 to Section 7, we introduce the four classes of IDS proposed for different WMN applications. In Section 4, we present PRIDE, a traffic-aware and resourceless intrusion detection proposed for resource-constrained WMN. Section 5 introduces RAPID, a

traffic-agnostic and resourceless intrusion detection inspired by the fact that PRIDE performance degrades when WMN topology varies frequently and traffic-awareness is not a realistic assumption. We also present a cooperative IDS for extremely resource-constrained WMN where a multi-objective optimization algorithm is used to produce an optimal cooperation model for intrusion detection mechanism. In Section 6, we present EEMON, an energy-efficient intrusion detection proposed for battery-powered WMN that benefits from memory-rich nodes. EEMON is a traffic-agnostic and resourceful IDS. Section 7 studies the effect of traffic-awareness on resourceful class of IDS, e.g., EEMON. Section 8 investigates the attack and fault tolerance of all of our intrusion detection mechanisms and proposes attack-and-fault tolerant (AFT) design for each of our proposed IDS.

Finally, in Section 9, we summarize this research, discuss the possibilities of applying the proposed intrusion detection systems to other networking areas, discuss future works and conclude the dissertation.

2. STATE OF THE ART

In this section, we first present background material on wireless mesh network and their applications. Next, we present related work in the area of intrusion detection systems for different wireless networks (e.g., mesh, sensor and mobile ad hoc networks) and emphasize the practicality and effectiveness of each solution. It has been more than a decade from the time first intrusion detection system for wireless ad hoc networks [97] was published. Since then, a significant amount of research papers have been presented in the area of intrusion detection in different infrastructure-less networks. We mainly focus on the solutions proposed for optimal monitoring for intrusion detection of these networks and do not cover the efforts in proposing intrusion detection engines (e.g., detection rules) for a specific type of attack in wireless networks. Additionally, we survey a series of administrative mechanisms for attack-and-fault tolerant (AFT) IDS design in different networking areas and show how they can be applied to AFT-design IDS in WMN. Aside from related work on intrusion detection, we also review energy efficient mechanisms and evolutionary algorithms that have been previously applied to wireless mesh network as we will use them as our optimization tools in designing different class of IDS in this research.

2.1 Background

As the first step towards providing dynamic and cost effective network services in environments with no network infrastructure, wireless mesh networks are becoming more popular. As an instance of real WMN implementations the rural wireless mesh network project in Zambia [6] provides telephony and internet access in some remote physical areas. Moreover, the lack of cellular network in disaster areas has convinced researchers [5, 16] to propose mesh networks as a cost-efficient and easy-

to-deploy solution in order to provide networking services in disaster situations. In addition to these applications, mesh networks have been deployed in academic and research centres as test-bed for developing and evaluating networking protocols for WMN [4, 87].

As wireless mesh networks become a popular choice for offering wireless services, security challenges grow in importance [9]. The problem of intrusion detection in wireless mesh networks has received some attention from the research community. Some existing solutions address specific attacks (e.g., Man-in-the-Middle and Wormhole Attacks [29], Selective forwarding [73] and Grayhole attack [74], selfish routing [90] and scheduling in WMN [48]). Other solutions are general IDS solutions for mesh networks, which consider memory, processing [42, 58], and energy [37] constraints. In [58], a set of technical challenges associated with IDS solutions in mesh networks are presented. The authors provide interesting evaluation results on the CPU utilization of a Netgear WG302 router and propose an initial design of a modular IDS but do not evaluate their solution. Performance evaluations of the Netgear WG302 mesh router, when running off-the-shelf IDS have also been reported [42].

2.2 Decentralized Intrusion Detection Systems

Adopting traditional intrusion detection mechanisms from wired networks is not practical because: a) WMN lack single vantage points (e.g., gateways in wired networks) where network traffic can be inspected; b) WMN hardware has *limited resources* (e.g., CPU and RAM) to run resource-demanding intrusion detection systems (IDS). The lack of concentration points where network traffic can be analyzed has been investigated mainly in the context of MANET and sensor networks. There, IDS were completely decentralized, and an intrusion detection agent was placed on each node [97]. These solutions were very inefficient since nodes in the network would

execute intrusion detection in a redundant manner (e.g., a multi-hop stream was analyzed multiple times) thus consuming both hardware resources (that could be allocated to other network functions) and energy.

A recent work [42] investigates challenges in applying off-the-shelf IDS (Snort [78] and Bro [65]) on mesh devices and proposes a lightweight (i.e., customized) IDS for WMN. The proposed lightweight IDS requires less memory and decreases the packet drop rate, when compared to off-the-shelf IDS. These achievements, however, are at the price of detecting fewer types of network attacks (smaller detection coverage and higher false negative rates), since most IDS functions are not implemented. In addition, although deploying same lightweight intrusion detection agent on every single WMN node may decrease the IDS loads on the nodes, it is still suffer from inefficient redundant monitoring for multi-hop traffic.

2.3 Optimal Monitoring Mechanisms

The identified inefficiencies in section 2.2 have triggered significant research on *optimal monitoring* for intrusion detection. In a *optimal monitoring* solution, a minimum subset of nodes are strategically selected to perform intrusion detection to *monitor* the entire network. More recently, several methods have been proposed for selecting nodes that run intrusion detection functions. While these nodes are primarily selected based on connectivity in wired networks, in resource constrained wireless networks the selection criteria is much more complicated. The proposed methods fall largely in two categories: *distributed* algorithms and *centralized* algorithms. Nodes selected by either distributed or centralized methods are referenced in different papers as: monitoring nodes, cooperators, aggregators, or cluster heads.

In a *distributed* solution each node decides individually which neighbor is the best monitor for it. Some distributed solutions use the concept of local elections and vot-

ing for monitoring node selection (and the cluster of nodes each monitoring is responsible for) [13, 15, 54, 89]. Nodes may either use, as selection criteria, their neighbors' capability, e.g., residual battery charge [47, 79] or degree of connectivity [46]. Other solutions employ simple random selection as an election protocol [41, 95]. The problem with this type of algorithms is that the best possible clustering is only optimal at the local level due to lack of global information. *Centralized* solutions guarantee that the decision will be made based on more complete information about the nodes and the network. For this, a powerful central node is required to run more complicated algorithms. There are several centralized algorithms for selecting the location of monitoring nodes that run in polynomial time [19, 75, 83]. These algorithms, however, consider only node coverage as an optimization objective. Taking into account other issues, e.g., amount of information collected by the nodes, total power consumption, and delay in detection process, may significantly affect the role assignment. The tradeoffs that centralized and distributed solutions have, and how these tradeoffs affect network performance and intrusion detection rate will be explored in this research. Centralized algorithms produce near optimal solutions since more information is used by the selection algorithm. Distributed algorithms, with lower time and message complexities, produce locally optimal solutions.

Recently, an optimized solution for selecting monitoring nodes in a multi-radio multi-channel wireless mesh network is proposed [76]. The problem is formulated as an ILP and solved with rounding techniques. We will propose an energy efficient monitoring technique for intrusion detection in battery powered WMN. Since the number of IDS functions running on the *monitoring node* is limited by the amount of available resources, these solutions, in resource-constrained WMN, only detect a limited number of attacks even though all communication links and network traffic are monitored. Since the number of services provided by WMN is expected to increase

(e.g., delay tolerant services [16], VoIP services [1]), fewer resources will be available for intrusion detection. Consequently, the intrusion detection rate is expected to degrade if the WMN nodes are not resourceful. In order to achieve higher detection rates, some solutions [75,76,83] assume that monitoring nodes are *resourceful* devices and are able to perform a complete IDS configuration. Thus, a monitoring node in a resourceful class of IDS solutions investigates its local data with all IDS functions, i.e., no need for cooperation. Most of the state-of-art monitoring node solutions [42, 75, 76, 83] assign the same set of IDS functions to monitoring nodes (where each monitoring node is responsible for a distinct part of the network).

2.4 Cooperative IDS

When considering resource-constrained WMN where nodes are not able to perform full IDS, optimal monitoring solutions cause high false negative rates as most of IDS functions have to be deactivated due to memory limitations. Another class of IDS solutions proposed for resource-constrained networks relies on cooperation mechanisms between resourceless nodes. *Cooperative IDS* solutions (e.g., hierarchical [33,54,69,84], group-based [49,50,52], or zone-based [81], or neighbor-assisted [67, 94] cooperation) distribute IDS agents [97] on wireless nodes. IDS agents consist of a local detection engine (to detect attacks based on local observation) and also a cooperative detection engine (to detect attacks based on local data and neighbors' reports [62, 94]).

The main reasons for using cooperative detection engine are: 1) to achieve higher detection rates through nodes collaboration [33, 34, 41, 81]; and 2) to reduce the IDS load on the resource constrained ad hoc nodes [42, 58] by distributing IDS functions to multiple nodes. Cooperative mechanisms, however, incur high communication overhead since nodes have to exchange their local observations with others run-

ning different IDS functions. Moreover, waiting to receive neighbors' data and then making the decision imposes high latency in cooperative intrusion detection systems. Therefore, although cooperative IDS have proven viable for low-traffic networks, e.g., sensor networks, they are not practical (i.e., degrades the network performance and delays intrusion response) in WMN with significant traffic [1, 2, 23, 38, 59] caused by mesh clients and external hosts.

2.5 Traffic Awareness in Intrusion Detection

The fact that neither monitoring node solutions nor cooperative IDS techniques can practically solve the intrusion detection problem in resource-constraint WMN motivates researchers to propose non-cooperative IDS solutions for these WMN networks. As a traffic-aware approach to overcome resource constraints and increase the detection rate, TRAM [30] uses mesh routers to monitor multi-channel WMN. Another research, in wired networks [70], proposed a scheme where each node along a network path executes a full Bro IDS. To save resources (*processing and memory that would be allocated based on traffic*), each node investigates only a portion of the network traffic. Although we will show that this method cannot be directly applied to resource-constrained WMN since it assumes that each node performs all IDS functions, we use the traffic-aware IDS rule distribution in the RL-TW class of IDS.

Traffic-aware solutions considers static resource-constrained WMN where network topology does not change often (compared to other ad hoc networks). In fact, they assumes that network information periodically collected by central node reflects the most recent network topology. However, research has shown [14, 16, 64] that even static WMN topology and routing paths are subject to change due to link-quality variations, mobility of clients and their requested services , and node failure (e.g., running out of power) during the network lifetime. Hence, traffic awareness might

be a strong assumption for many WMN applications. Motivated by this fact, traffic-agnostic IDS solutions are of paramount importance for many WMN applications.

2.6 Attack and Fault Tolerant IDS

Although intrusion detection mechanism in WMN have received considerable focus, little attention has been paid to attacks-and-failures against/of IDS nodes. Undoubtedly, when an IDS node is compromised or faulty, it is unable to participate in intrusion detection process, thus, the intrusion detection rate will decrease and some malicious activities will remain undetected (i.e., high false negative rates). The IDS attack/failure problem has received some attention in other computer networking areas [11, 55, 56, 61, 96]. Some of the proposed solutions use redundant/backup nodes [55] to increase the network/service availability after node compromise/failure while others concentrate on camouflaging mechanisms [61, 96] to make monitoring/IDS nodes localization [11] very hard for the attacker. Furthermore, few other solutions propose fast and efficient fault detection mechanisms to detect compromised or faulty nodes [56] and recover the network from that situation [55, 61].

2.7 Optimization Techniques in WMN

This research proposes five different intrusion detection mechanisms that all of them aim at providing optimal monitoring mechanisms -each for a specific WMN application (e.g., battery-powered, resource-constrained, etc.) We have used different optimization tools (e.g., integer linear programming, evolutionary algorithms, and combinatorial optimization) throughout this dissertation. Moreover, the energy-efficient monitoring solution presented in Section 6 in motivated by other research in battery-powered WMN. Thus, this section reviews energy-efficient and evolutionary algorithms previously used by research community.

Energy-efficient intrusion detection in wireless networks has received some atten-

tion [47, 71, 82]. As in the case of other efforts in power-constrained WMN, some power-aware algorithms have been proposed for solar-powered WMN [7, 24, 57]. Reducing the load on battery was proposed for giving battery recovery time [24, 57]. An on/off controller was proposed theoretically for battery recovery [66]. The idea of reducing the battery load is proposed to give the battery a recovery time to prolong its total lifetime. This idea led us to apply a duty-cycling method to WMN nodes (similar to the on/off controller proposed in [24]) to recover the battery residual charge [66].

Our proposed cooperative IDS solution, presented in Section 5, is influenced by recent developments in applying evolutionary algorithms to intrusion detection and cluster formation in wireless networks. In [72] a grammatical evolution is proposed for creating an intrusion detection engine in wireless nodes. The authors have extended their work in [71] to make optimal tradeoffs between the detection rate of intrusion detection programs and the amount of power they consume. [43] suggests using a genetic algorithm (GA) to form intelligent, energy-efficient clusters in wireless sensor networks (WSN). A similar approach for data aggregation in wireless sensor network is presented in [3]. The authors propose a genetic algorithm to find an optimal grid-based routing with minimum energy dissipation and latency in WSN. Other efforts in data aggregation and event detection in wireless sensor network employ neural network to find more efficient aggregation paths and to predict sensor outputs accurately. In [68] the authors employ a neural network function to predict the changes in sensor outputs. Such a signal change detection, as authors claim, could improve data compression and consequently enhance the network reliability and security. In order to eliminate redundant data and improve the aggregation accuracy, [86] proposes a neural network-based method for data aggregation in wireless sensor network. The authors show how distributing input layer and hidden layer neurons to cluster mem-

bers and cluster heads, respectively, would improve data aggregation and reduce the energy consumed for event reporting to the base station.

3. SYSTEM AND SECURITY MODELS

This section presents the system and security (attacker) models for all solutions proposed in this research and highlight their similarities and differences. Depending on the WMN application we consider, the system model, the amount of available resources, traffic patterns, client accessibility, etc. would change. Moreover, the attacker model in each class of IDS solutions depends on the WMN application and also the system model. This section uses "mesh router" and "node" interchangeably and refers to a "WMN client" as "client."

3.1 System Model

The WMN system we consider in this research is as specified by the IEEE 802.11s WLAN Mesh Standard [40]. The system, as shown in Figure 3.1, consists of: i) *mesh access points (MAP)* that connect WMN clients to the mesh network and external hosts (i.e., Internet); ii) a *wireless mesh backbone* consisting of relay nodes, also known as *mesh points (MP)*; and iii) *gateways* that connect the mesh network (internal hosts) to the Internet (external hosts). The network traffic, as shown in Figure 3.1, is either between WMN clients and external hosts (i.e., *external traffic*) or between two internal hosts (clients or local servers) inside the mesh network (i.e., *internal traffic*). The system proposed for PRIDE [38] and RAPID [36] considers resourceless AC-powered nodes. EEMON [37] and TRAIN [31], on the other hand, assume resourceful battery-powered nodes. All of these systems, however, assume that the WMN is connected to the Internet through more powerful gateway routers that do not have energy constraints (AC powered). Some nodes in the WMN operate in a duty-cycled manner to save energy.

Each router in a WMN has some local information (e.g., its communication load

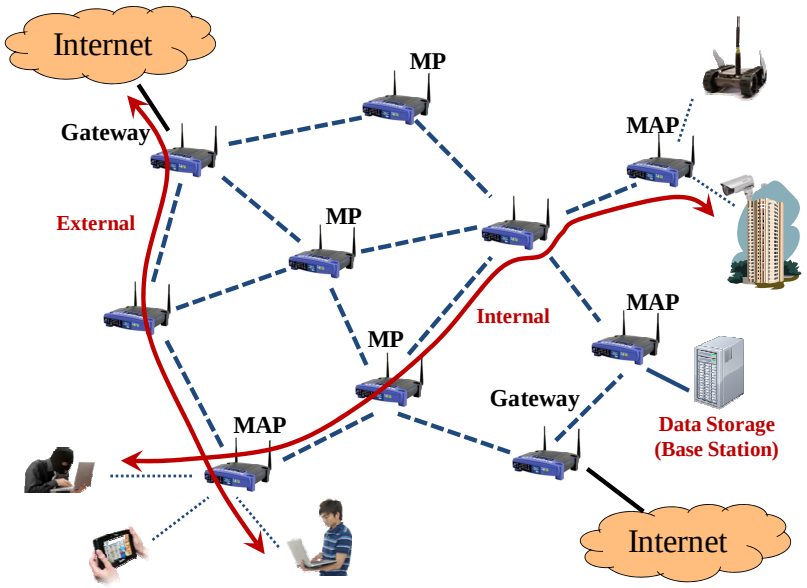


Figure 3.1: A typical mesh network with internal and external traffic flows.

and its residual energy, processing/memory loads, connectivity and traffic information) and periodically sends it, via a middleware and secure communication links, to the base station. For both EEMON and TRAIN, used in battery-powered WMN applications, each node sends its communication load and residual energy to the base station. In TRAIN (i.e., a traffic-aware solutions), however, traffic information is also sent to the base station. WMN nodes in PRIDE and RAPID send their processing/memory loads and connectivity, however, the traffic information also has to be sent to the base station in PRIDE. Based on the collected information, the base station assigns intrusion detection tasks to the nodes. The role of each node (i.e., performing what IDS functions) is securely broadcast to the network using the energy-efficient flooding protocol presented in [35].

3.1.1 WMN Topology

Throughout this research, we evaluate our proposed intrusion detection systems using real-world WMN and simulations. In both scenarios, WMN topologies are based on IEEE 802,11s WLAN Mesh Standard [40] and consist of at least one MAP, one Gateway, and one MP. More precisely, every random topology that is generated in simulations is first investigated for following the aforementioned standard rules, if verified, it will be considered as a valid WMN topology and used for performance evaluations. For example, a disconnected graph created randomly will not be considered as a valid WMN topology in our simulation. Moreover, the radio range, network density, and number of paths in each WMN are chosen carefully and based on our real-world experiments. More details about number of random topologies for each experiment and their characteristics for each proposed IDS mechanism are explained in details in each section of this dissertation.

3.2 Intrusion Detection Engine

The IDS we consider in this research (i.e., for PRIDE, RAPID, EEMON, and TRAIN) is Snort [78]. We chose Snort because it is a mainstream off-the-shelf IDS that consumes less resources than other IDS, e.g., Bro (as was recently shown [42]). Moreover, Snort is readily available for mesh hardware, as part of the OpenWrt (i.e., a Linux distribution for embedded networking devices) development tree. Snort can be configured for different levels of intrusion detection. More complex actions performed by the detection engine (e.g., number of active rule sets) require more system resources. Therefore, the configuration of the detection engine provides opportunities to trade off intrusion detection rate for resource availability.

Resourceful IDS solutions, EEMON and TRAIN, use two configurations for the detection engine: *complete* (CP-DS) and *lightweight* (LW-DS). IN CP-DS configu-

ration, all Snort detection rules are activated (same as Snort default configuration), while a few of them are activated in LW-DS configuration. The detection rules in CP-DS are configured to investigate all traffic flows, while LW-DS detection engine investigate only local clients' traffic flows. EEMON and TRAIN assign monitoring or non-monitoring roles to nodes. A monitoring node runs a complete IDS configuration (CP-DS), while a non-monitoring node runs a lightweight IDS configuration (LW-DS). In fact, non-monitoring nodes only monitor their clients' activities and do not monitor WMN backbone traffic. A monitoring node is always awake, while a non-monitoring node executes with a duty-cycle. The gateway is always a monitoring node. Nodes in resourceless IDS, PRIDE and RAPID, use *customized IDS configuration* (i.e., customized to perform a few IDS functions to no function at all) for the detection engine.

From detection engine perspective, intrusion detection systems can be categorized in three classes; a) anomaly-based IDS, b) misuse-based IDS, and c) hybrid IDS, which is a combination of the former two. The proposed intrusion detection system might seem to detect only known attacks since Snort is a signature-based detection engine. However, unknown attacks are also detected if the intrusion detection engine employs anomaly-based inspection (as claimed on the Snort Website "`http://ww.snort.org`"). Another example of an anomaly-based engine employed by an off-the-shelf IDS is Bro [65], also ported to OpenWrt in [42]. Bro is able to discern traffic anomalies and to detect unknown attacks. Thus, our proposed solutions are not limited to detecting only known attacks, but also stealth attack with appropriate detection engine. We also emphasize here that our contribution in this research is not to propose a new intrusion detection rule, but to propose optimal approaches of applying proposed intrusion detection rules and engines to different WMN applications.

Table 3.1: Attacker models

			Attacker		Target	
			Insider❶	**Outsider**	**Single-hop**	**Multi-hop**
Traffic Agnostic	**Resourceless**	RAPID	✓	✓	✓	✓
	Resourceful	EEMON	✓	✓	✓	✓
Traffic Aware	**Resourceless**	PRIDE	✓	×❷	×	✓
	Resourceful	TRAIN	✓	×❷	×	✓

❶ A malicious client or a compromised router.
❷ Only external hosts, as a type of outsider attackers, are considered. Unauthorized clients are not considered.

3.3 Attacker Models

Table 3.1 summarizes the types of attacks considered in our IDS solutions. An *Insider* attacker is either a malicious client connected to a MAP or a compromised router. As shown in Table 3.1, all four IDS classes address insider attackers. An *Outsider* attacker is either an external host (connected to WMN through gateways) or an unauthorized client not connected to WMN. For example, a malicious external host communicating with a mesh client is considered as an outsider attacker. Furthermore, a malicious unauthorized wireless node physically located in the WMN coverage area, but not associated to a WMN MAP, is also considered as an outsider attacker. As depicted in Table 3.1, PRIDE and TRAIN do not consider unauthorized clients. When evaluating the performance of these IDS solutions, we will show how this type of attack impacts the performance (i.e., intrusion detection rates) of these traffic-aware solutions.

Depending on the attacker type, i.e., insider or outsider, the target can be either single-hop or multi-hop. When considering single-hop targets, the attack can be either a *node-based* attack (targeting a WMN router or a host) or a *link-based* attack (targeting a communication link). A multi-hop attack, however, is always against hosts. An insider attacker, a malicious client or a compromised router can launch

attack against both single-hop or multi-hop targets. When considering outsider attackers, external hosts can only launch multi-hop attacks (against an internal host) while unauthorized nodes can only launch attacks against single-hop nodes or links. As shown in Table 3.1, TRAIN and PRIDE, as two traffic-aware solutions that monitor traffic paths, do not consider single-hop attacks. However, EEMON and RAPID, as two RF-TG solutions, consider both multi-hop and single-hop (i.e., node-based and link-based) attacks.

In order to detect both node-based and link-based attacks, EEMON and RAPID's novel approach for monitoring node selection is to monitor "wireless links" and not "nodes," as existing solutions [75, 83]. The link coverage detects attacks that affect functionality of communication links (e.g., jamming, wormhole, selective forwarding, etc.). Consider a linear topology of four nodes (as shown in Figure 3.2). They are in order ABCD and each node is connected to nodes that are physically adjacent. State of art solutions that monitor nodes (i.e., node coverage approach), may select nodes A and D as monitors (which cover all the nodes). However, this monitoring solution cannot cover the communication link between B and C. For example, if node C in Figure 3.2 is compromised and randomly drops some packets traveling from B to D (a single-hop link-based attack), node C will never be detected by monitors at A and D, unless there is a *cooperation* mechanism between them through another path. The link coverage approach can also improve the performance of traffic-aware solutions (e.g., TRAIN and PRIDE), in addition to traffic-agnostic solutions performance. For example, when selecting nodes to monitor a traffic path, the monitoring node can be chosen from nodes not necessarily located on the path but also those able to cover at least one link on the path. Therefore, we propose the link coverage approach for RAPID, EEMON and TRAIN and expect to achieve higher intrusion detection rate than node coverage approaches.

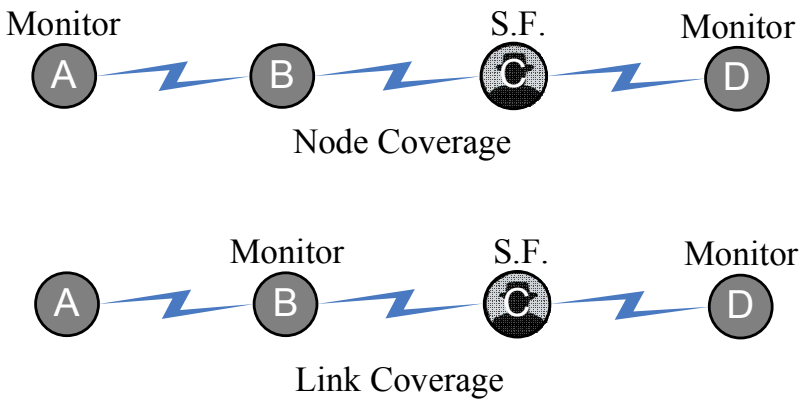


Figure 3.2: An example of selective forwarding attack.

Table 3.2: Detectability of different attacks in our proposed solutions.

		Traffic Agnostic		**Traffic Aware**	
		Resourceless	**Resourcefuls**	**Resourceless**	**Resourceful**
Single-hop	**Severe**	if covered by detecting modules	Only at Monitors	single-hop attacks are not considered	Only at Monitors
	Normal		Detectable		Detectable
Multi-hop	**Severe**	if covered by detecting modules	Detectable	if covered by detecting modules	Detectable
	Normal				

Since the detection engine used by EEMON/TRAIN has two different configurations, the attacks can be categorized into two different severity levels: one detectable by the LW-DS detection engine (i.e., *Normal* attack), and one by the CP-DS detection engine (i.e., *Severe* attack). Table 3.2 summarizes the detectability of these types of attacks in state-of-the-art solutions and the expected detectability in EEMON and TRAIN. Resourceless solutions, e.g., RAPID and PRIDE, that use customized IDS configuration can detect an attack only if its corresponding detection module is activated in the customized IDS. Hence, regardless of the attack's severity level, there is no certainty as to whether the attack will be detected by the resourceless solutions or not. Resourceful solutions (e.g., EEMON and TRAIN), on the other hand, are expected to detect Normal attacks since they are detectable by LW-DS detection engines performed by non-monitors. Obviously, CP-DS detection engines performed

by monitoring nodes can also detect Normal attacks. Severe attacks can be only detected by monitoring nodes running CP-DS engines. Hence, as depicted in Table 3.2, we expect Single-hop Severe attacks to be detectable if launched in the coverage area of a monitoring node. Multi-hop Severe attacks are expected to be detectable by resourceful solutions due to the fact that at least one monitoring node on each path performs CP-DS detection engine.

4. RESOURCELESS AND TRAFFIC-AWARE IDS

In this section*, we propose PRIDE, a PRactical Intrusion DEtection system for resource constrained WMN. The main idea in this section is to use the knowledge a security administrator has about the WMN traffic to distribute IDS functions more efficiently. More precisely, a security administrator, knowing the routing paths of the traffic in the WMN, would employ a traffic-aware framework that optimally places IDS functions on the nodes along the routing paths. For example, the idea of interference-load aware routing metric [59] in WMN, aims to route the mesh traffic through congestion free areas and provides a traffic-aware framework for the security administrator. The information about the busiest and most frequently used paths in the WMN is obtained from routing algorithms (e.g., OLSR) and network monitoring tools (e.g., tcpdump).

A related idea for traffic-aware IDS deployments in wired networks was recently proposed [70], where different IDS responsibilities (i.e., different portions of network traffic) are assigned to each node along the traffic paths while ensuring that no node is overloaded. However, as we will show in Section 4.1, [70] cannot be directly applied to WMN since it assumes that each node performs all IDS functions - infeasible for resource constrained mesh devices. Our proposed solution has no communication overhead (message exchange for cooperative decision making), has no detection latency (i.e., it provides real-time intrusion detection, in contrast to cooperative IDS) and it has a higher detection rate, when compared with traditional monitoring node

*Parts of this section are reprinted with permission from "PRIDE: Practical Intrusion Detection in Resource Constrained Wireless Mesh Networks" by Amin Hassanzadeh, Zhaoyan Xu, Radu Stoleru, Guofei Gu, and Michalis Polychronakis, In Proceedings of 15th International Conference on Information and Communications Security (ICICS), pages 213-228, Beijing, China, 2013, Copyright 2013 by Springer.

solutions. In our proposed solution, *each node along a routing path, runs a distinct and customized IDS*. This *customized* IDS (technically a subset of IDS functions) allows resource conservation. The combination of *distinct* IDS along the path allows for a complete set of IDS functions to be applied to the entire network traffic.

4.1 Motivation and Background

The research presented in this section is motivated by the challenges we faced when we attempted to deploy a common off-the-shelf IDS with a full configuration (i.e., configured to detect the largest set of attacks) on existing WMN router hardware. When loading Snort [78] with its full configuration on a Netgear WNDR3700 router, the router crashes because the RAM is not sufficiently large. In the remaining part of this section we describe in detail the hardware capabilities of our mesh routers, background information on Snort, and experimental results that illustrate how different Snort configurations of increasing complexity and detection capabilities impact the memory load of the router.

The Netgear WNDR3700 router has an Atheros AR7161 processor running at 680MHz, 64MB RAM, 8MB flash memory. It has two wireless cards with Atheros AR9223-bgn and Atheros AR9220-an chipsets, working on 2.4GHz and 5GHz, respectively. The operating system on the router is the most recent release of OpenWrt (i.e., Backfire 10.03.1), a Linux distribution for embedded networking devices, with kernel version 2.6.32.10. We emphasize that our mesh hardware is more powerful (in terms of processing and memory resources) than devices used in some existing real world deployments [1, 4, 6]. Although in this research we focus mainly on Netgear WNDR3700 router hardware, later in this section we present our experience and results with more sophisticated and expensive mesh hardware, e.g., Meshlium Xtreme [53] which has a 500MHz CPU, 256MB RAM, 8/16/32GB disk memory and

WiFi, Zigbee, and GPRS wireless interfaces.

The router runs Snort [78], an off-the-shelf intrusion detection system. Snort's detection engine is based on thousands of detection rules (categorized in multiple rule files, corresponding to known network threats) and several preprocessors. All files are listed in "snort.conf", a global configuration file. Upon activating each rule file in "snort.conf" and running Snort, all detection rules present in the rule file are loaded in memory and are used for packet investigation. A full Snort configuration activates all preprocessors and rule files. A customized configuration activates only some preprocessors and rule files (i.e., IDS functions), thus, the network traffic is analyzed by fewer detection functions.

The intrusion detection in Snort is performed by packet-level rule matching. Each packet is preprocessed, following preprocessing directives for extracting possible plain-text content. The preprocessed packet is then inspected by Snort detection rules, to expose whether it is an intrusion attempt or not. Preprocessors parse network packets and provide abstract data for some high-level detection rules in the rule files. It is important to note that a rule file that contains high-level detection rules has *preprocessor dependency*. This dependency means that the rule file cannot be activated (i.e., Snort generates an error message and stops) unless all the preprocessors required by its rules (usually one or two preprocessors) are also activated. From here on, we refer to a Snort rule file as an "IDS function."

To understand how different Snort configurations impact the memory load on the Netgear WNDR3700 and Meshlium Xtreme, we performed several experiments. Running Snort causes two types of memory loads to the router: 1) *static*, the initial load imposed by packet capturing modules, preprocessors, detection rules, etc. when Snort is loaded; 2) *dynamic*, the variable load imposed by stateful preprocessors (e.g., *Stream5*) which is a function of the traffic load and some configuration parameters.

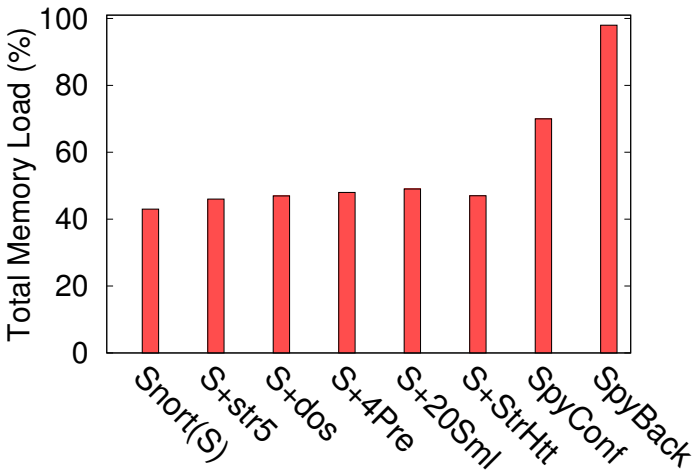


Figure 4.1: The effect of Snort configuration on memory consumption.

We first investigate the static memory load of Snort on the routers when no network traffic is applied. We have observed that a typical memory load on a Netgear WNDR3700 router is ∼30% and on the Meshlium Xtreme router it is ∼60%. This accounts for OS firmware and various services (OLSR routing, DHCP, etc.). Without preprocessors or rule files active, loading Snort on Netgear WNDR3700 increases memory load to 43% ("Snort(S)" in Figure 4.1). Memory load increases to 46% if preprocessor Stream5 is activated ("S+str5" in Figure 4.1), and to 48% if preprocessors *http-inspect*, *smtp* and *ftp-telnet* are also activated ("S+4Pre" in Figure 4.1).

The memory load of a rule file is a function of the number of detection rules in it and the pattern matching algorithm Snort uses (e.g., `Aho-Corasick`). We note here that in this research, we use the default search method of Snort, i.e., `ac-bnfa-nq`, as we experimentally observed that it consumes the minimum memory among all *low memory* search methods, e.g., `lowmem`. For example, using `ac-bnfa-nq` search method, "dos.rules" which has 20 detection rules and requires the Stream5 preprocessor, increases memory load to 47% ("S+dos" in Figure 4.1). A very large file such as "spyware-put" ("SpyConf" in Figure 4.1) which contains ∼1,000 rule

files increases the RAM load to 70%. The memory load caused by activating a set of rule files also depends on their sizes. For example, activating 20 small rule files (i.e., 10 rules per file on average) and the Stream5 preprocessor (which the rules require) increases memory load to 49%. Activating two large rule files, "spyware-put.rules" and "backdoor.rules" ("SpyBack" in Figure 4.1) increases memory load to 98%. We have experimentally verified that adding a few small rule files on top of "spyware-put.rules" and "backdoor.rules" causes the router to crash. *We have observed a similarly overloaded operation for the Meshlium Xtreme router, where a full configuration Snort increases the memory load to 98.5%, leaving almost no room for processes/services beyond stock deployment.* We also emphasize here the rapid increase in the number of Snort rule files (i.e., currently about 70 files) and their sizes as functions of the number of threats. Some rules might not be needed in a particular setting, but conversely, that setting might require many more rules of some other kind (e.g., custom signatures for suspicious or blacklisted domains, which can increase significantly).

Dynamic memory load, imposed by Stream5 when tracking traffic sessions, is the other considerable type of Snort memory load since almost all rule files require this preprocessor. Two configuration parameters of Stream5, `max_tcp` and `memcap`, specify the maximum simultaneous TCP sessions it tracks (similarly, `max_udp`, `max_icmp`, and `max_ip`) and the maximum buffer size for TCP packet storage, respectively. We have experimentally observed that the value of `max_tcp` affects both dynamic and static memory loads. When using the Snort version available on the OpenWrt development tree, the default configuration has max_tcp=8192. Choosing max_tcp=100,000, imposes ∼10% more static load than default "S+Str5" to the routers. Moreover, this value allows more simultaneous TCP sessions to be inspected which also imposes more dynamic load and may cause the router to crash at high traffic rates (note: we

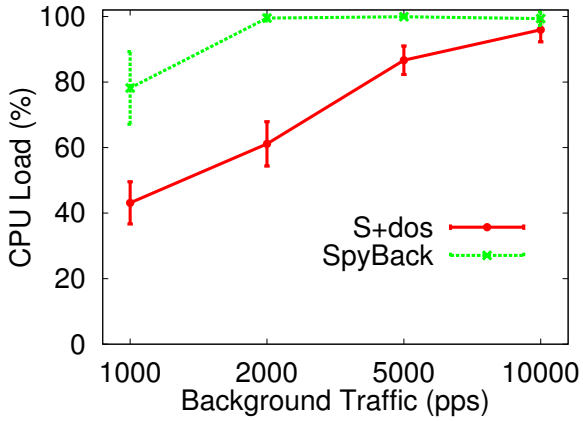


Figure 4.2: The effect of Snort configuration and traffic rates on CPU utilization.

observed that for max_tcp≥150,000 the router crashes if a simple HTTP request is sent using the Linux *wget* tool). Throughout this research, we use the default setting, i.e., max_tcp=8192, and consider the maximum dynamic load this setting imposes on the router. Hence, from here on, the total memory load of Stream5 is assumed to be its static load plus its maximum allowable dynamic load. It is worth mentioning that although hardware improves, also transmission speeds get faster, the amount of traffic that needs to be inspected grows, and the complexity of the applied processing increases. Hence, the fundamental challenge for a resource-limited node to handle *ever-increasing traffic* still remains.

In addition to RAM, processing power (CPU) is also limited on current mesh hardware. Consequently, investigating the impact of Snort IDS on this limited resource might seem worthwhile. Experimentally we have found that network traffic, actually, has a much larger influence on CPU utilization than executing Snort IDS functions. Our experimental results are depicted in Figure 4.2 where we enabled `tcp_track` and `icmp_track` in Stream5 and used *hping3* to generate TCP and ICMP traffic. As shown, for an extremely high traffic rate, both lightweight and heavy Snort configurations impose more than 95% CPU utilization. Similar with our result, it

was shown [42] that even a lightweight IDS exhausted the CPU when traffic rate was extremely high. However, as shown in Figure 4.2, "S+dos", a lightweight IDS configuration, imposes less processing load than "SpyBack", a heavyweight IDS configuration, when the traffic rate is not high. *Consequently, our main concern in this research is the reduction of RAM utilization as we have experimentally observed that it also improves the CPU utilization in regular traffic rates (as shown in Figure 4.2).*

4.2 Preliminaries and Problem Formulation

In this section, we formulate the optimal distribution of IDS functions as an optimization problem and we propose a method to solve it. Although Snort is our target IDS (and present a problem formulation that uses Snort terminology), we believe that other IDS can be analyzed similarly, if their internals and functionality are publicly available. For example, in Di-Sec [88], a security framework recently proposed for wireless sensor networks, the sub-components of M-Core can be modeled as Snort preprocessors while the detection and defense modules play the same roles as Snort rule files.

4.2.1 IDS Function Distribution

We denote the number of nodes and number of links in the wireless mesh network by N and Q, respectively. Considering the information collected from the nodes, we denote the number of nodes and links actively contributing in traffic routing by n ($n \leq N$) and q ($q \leq Q$), respectively. Thus, we model the wireless mesh network (i.e., after removing *idle* nodes/links) as a reduced graph $G = \{V, E\}$, where V is the set of nodes $\{v_1, v_2, \cdots, v_n\}$, and E is the set of links $\{e_1, e_2, \cdots, e_q\}$. An example of a reduced graph, in Figure 4.3, $V = \{v_1, v_2, ..., v_9\}$ and $E = \{e_1, e_2, ..., e_8\}$.

We denote the set of routing paths for the network traffic by $P = \{p_1, p_2, \cdots, p_l\}$, where set $P_i = \{v_j \mid v_j \text{ is located in } p_i\}$ and $P_i \subseteq V$. In Figure 4.3 two paths are

present: p_1 and p_2. Additionally, we denote by matrix $\mathbb{T}_{l\times n}$ the mapping between nodes and paths, i.e., $t_{ij} = 1$ iff node j is located on path i. For the example shown in Figure 4.3, the matrix $\mathbb{T}$ is as follows:

$$\mathbb{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We denote the set of all IDS functions by $\mathcal{F} = \{f_k \,|\, f_k \text{ is a set of detection rules}\}$ with size K (i.e., $|\mathcal{F}| = K$). We denote the set of IDS preprocessors by $\mathcal{C} = \{c_r \,|\exists\, f_k \in \mathcal{F} \text{ that requires } c_r\}$ of size R (i.e., $|\mathcal{C}| = R$). For the example in Figure 4.3, $\mathcal{F} = \{f_1, f_2, ..., f_7\}$ and $\mathcal{C} = \{c_1, c_2\}$. The dependency between IDS functions and preprocessors is stored in matrix $\mathbb{D}_{K\times R}$ where $d_{kr} = 1$ means that activation of function f_k requires the activation of preprocessor c_r. For the example shown in Figure 4.3, the matrix $\mathbb{D}^T$ is as follows:

$$\mathbb{D}^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Let $w : \{\mathcal{F}, \mathcal{C}\} \longrightarrow [0, 1]$ be a cost function that assigns memory load w_k^f and w_r^c to IDS function f_k and IDS preprocessor c_r, respectively. Consequently, vectors $W^f = [w_1^f, w_2^f, \cdots, w_K^f]$ and $W^c = [w_1^c, w_2^c, \cdots, w_R^c]$ represent memory loads for the IDS functions in $\mathcal{F}$ and for the IDS preprocessors in $\mathcal{C}$, respectively (we remark that $w_{Stream5}^c$ is the summation of its static load and its maximum dynamic load). We denote by $B = [b_1, b_2, ..., b_n]$ the base memory load (i.e., without IDS functions loaded) of all nodes.

Finally, we use vector $\Lambda = [\lambda_1, \lambda_2, \cdots, \lambda_n]$ (*also called Memory Threshold*) to represent the maximum allowable memory load after IDS functions are loaded. Memory

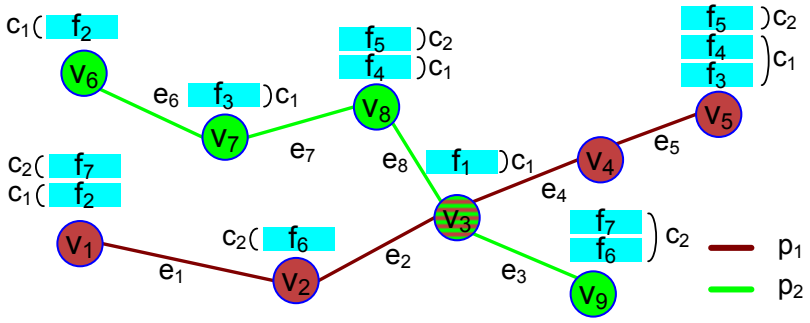


Figure 4.3: An example graph for a mesh network, consisting of 9 nodes and 8 links. As shown, two paths p_1 and p_2 are present. The nodes run different configurations of Snort, e.g., node v_5 runs Snort functions f_3, f_4 and f_5, which require preprocessors c_1 and c_2.

threshold is an important parameter. It is typically set by a network administrator based on the number of active services in the mesh network and the memory space they require.

Definition 1 *An **IDS Function Distribution**, $A = \{(v_j, \mathcal{F}_j, \mathcal{C}_j) | \, v_j \in V, \mathcal{F}_j \subseteq \mathcal{F}$, and $\mathcal{C}_j \subseteq \mathcal{C}\}$, is a distribution of IDS functions in the network, such that node v_j only executes IDS functions $\mathcal{F}_j$ and their corresponding preprocessors $\mathcal{C}_j$.*

For example, the *IDS Function Distribution* in Figure 4.3 is:

$$A = \{(v_1, \{f_2, f_7\}, \{c_1, c_2\}), (v_2, \{f_6\}, \{c_2\}), ..., (v_9, \{f_6, f_7\}, \{c_2\})\}.$$

We represent an *IDS Function Distribution* by matrices $\mathbb{X}_{n \times K}$ and $\mathbb{Z}_{n \times R}$, corresponding to IDS functions and preprocessors active on each node, respectively. For $\mathbb{X}$, $x_{jk} = 1$ iff IDS function f_k is activated on node v_j. For $\mathbb{Z}$, $z_{jr} = 1$ iff preprocessor c_r is activated on node v_j. Matrices $\mathbb{X}$ and $\mathbb{Z}$ for the network in Figure 4.3 are:

$$\mathbb{X} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \qquad \mathbb{Z} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Considering the above mathematical formalism, the dependencies between IDS functions and preprocessors can now be represented more compactly as:

$$z_{jr} = \begin{cases} 1 & if \quad (\mathbb{X} \cdot \mathbb{D})_{jr} \geq 1 \\ 0 & if \quad (\mathbb{X} \cdot \mathbb{D})_{jr} = 0 \end{cases} \tag{4.1}$$

Equation 4.1 indicates that preprocessor c_r must be activated on node v_j if there exists at least an IDS function f_k requiring c_r, assigned to it. It is important to note that $z_{jr} = min\{1, \Sigma_{k=1}^{K} x_{jk} d_{kr}\}$ and $z_{jr} \in \{0, 1\}$.

After the *IDS Function Distribution*, the total memory load for node v_j becomes $L_j = b_j + \Sigma_{c_r \in \mathcal{C}_j} w_r^c + \Sigma_{f_k \in \mathcal{F}_j} w_k^f$ where $w_r^c \in W^c$ and $w_k^f \in W^f$. It is important to mention that an *IDS Function Distribution* in which $L_j > \lambda_j$, i.e., the load L_j is greater than threshold λ_j, for any node v_j, is deemed infeasible.

From a network security administrator point of view, we aim for an *IDS Function Distribution* where all IDS functions are activated on each path. This means that the entire network traffic will be investigated by all IDS functions (albeit at different

times), eliminating the possibility of false negatives.

Definition 2 *For a given path p_i and its corresponding set of nodes P_i,* ***Coverage Ratio (CR)*** *is defined as $CR_i = |U_i|/K$, where $U_i = \bigcup_{v_j \in P_i} \mathcal{F}_j$ is the set of IDS functions assigned to nodes along the path. Path p_i is called* ***covered*** *if $CR_i = 1$ ($U_i = \mathcal{F}$), i.e., for $\forall f_k \in \mathcal{F}$, $\exists\, v_j$ assigned by $\mathcal{F}_j$ such that $f_k \in \mathcal{F}_j$.*

Considering the effect of *IDS Function Distribution* on the memory load of each node and the desired distribution of IDS functions to the nodes, in order to achieve higher intrusion detection rate, we define Path Coverage Problem (PCP) as follows:

Definition 3 ***Path Coverage Problem (PCP)***
Given $G = \{V, E\}$, a set of paths P in WMN, the dependency matrix $\mathbb{D}$, and vectors W^f and W^c, find a distribution $A = \{(v_j, \mathcal{F}_j, C_j) |\, v_j \in V$ and $\mathcal{F}_j \subseteq \mathcal{F}$ and $C_j \subseteq C\}$, such that $\frac{1}{l} \sum_{p_i \in P} CR_i$ is maximized and $L_j \leq \lambda_j\ \forall v_j \in V$.

PCP is an optimization problem which has the objective of maximizing the average coverage ratio, while guaranteeing that memory loads on nodes are below a memory threshold. Although a lower memory threshold λ_j allows more additional processes to execute on the mesh router, it makes solving PCP much more difficult.

4.2.2 Optimal IDS Function Distribution

We formulate PCP as an Integer Linear Program (ILP) that can be solved by an ILP solver. The objective function is maximizing the average coverage ratio of all paths. Additionally, preprocessor dependency and memory threshold are considered as ILP constraints. More specifically, the ILP formulation is as follows:

$$\text{maximize} \quad \frac{1}{l}(\mathbf{1}^T \cdot \mathbb{T})(\mathbb{X} \cdot \mathbf{1}) \tag{4.2}$$

$$\text{subject to:} \quad B^T + \mathbb{Z} \cdot W^{cT} + \mathbb{X} \cdot W^{fT} \leq \Lambda^T \tag{4.3}$$

$$(\mathbb{T} \cdot \mathbb{X})_{ik} \leq 1 \qquad ,\forall i,k \tag{4.4}$$

$$z_{jr} \geq \frac{(\mathbb{X} \cdot \mathbb{D})_{jr}}{K} \qquad ,\forall j,r \tag{4.5}$$

$$z_{jr} \leq (\mathbb{X} \cdot \mathbb{D})_{jr} \qquad ,\forall j,r \tag{4.6}$$

$$x_{jk}, z_{jr} \in \{0,1\} \qquad ,\forall j,k,r \tag{4.7}$$

To better understand the mathematical formulation of the objective function, one can expand it as $\frac{1}{l}\Sigma_{i=1}^{l}\Sigma_{j=1}^{n}\Sigma_{k=1}^{K} t_{ij}x_{jk}$ where $t_{ij} = 1$ means node v_j is located on path p_i and $x_{jk} = 1$ means node v_j is assigned by function f_k. In other words, the average CR has to be maximized.

Constraint 4.3 limits the memory load on every node v_j, i.e., $\Sigma_{r=1}^{R} z_{jr}w_r^c + \Sigma_{k=1}^{K} x_{jk}w_k^f$, to be less than its memory threshold λ_j. Most importantly, (*to ensure that we can formulate PCP as a linear program*), this constraint computes the total memory load as the sum of individual memory loads of preprocessors and rule files. Obviously, one needs to investigate if this linearity assumption always holds (we will discuss this in the next section). Constraint 4.4 ensures that only one copy of each function is assigned to the nodes along each path. Constraints 4.5 and 4.6 ensure that if an IDS function is assigned to a node, its required preprocessors are also assigned to the node. As presented in Equation 4.1, $z_{jr} = 1$ if at least one of the IDS functions assigned to node v_j requires preprocessor c_r, otherwise $z_{jr} = 0$. The maximum number of functions that require a specific preprocessor is at most K. Hence, Constraint 4.5 ensures that $0 < z_{jr} \leq 1$ if there is a function assigned to node v_j that

requires preprocessor c_r. On the other hand, if none of the functions assigned to node v_j requires preprocessor c_r, then Constraint 4.6 enforces z_{jr} to be zero. Taking into account Constraint 4.7, i.e., z_{jr} has to be either 0 or 1, Constraint 4.5 enforces $z_{jr} = 1$ if preprocessor r is required on node j, otherwise, Constraint 4.6 enforces $z_{jr} = 0$.

4.3 PRIDE: Challenges and Solutions

Considering the aforementioned ILP formulation for PCP, we investigated two major challenges that impact the accuracy and time complexity of a solution. First, we experimentally observed that the total memory load of multiple Snort rule files is generally linear (i.e., it is equal to the sum of their individual memory loads), but not always (e.g., for some small rule files and certain rule types). This influences the accuracy of our proposed model for calculating the total memory load on each node (i.e., Challenge 1). Next, one can observe that the complexity of ILP depends on the number of paths in the network, the path lengths, the number of IDS functions, the number of preprocessors, and the memory threshold. For example, considering the number of Snort preprocessors (i.e., more than 20) and the number of Snort rule files (i.e., more than 60), *for single path* p_i, the number of ILP constraints grows to more than $1400 \times |P_i|$, where $|P_i|$ is the path length. Additionally, a lower memory threshold λ_j increases the number of infeasible solutions, thus requiring more iterations for the ILP solver. Hence, the ILP performance degrades as network size increases or memory threshold decreases (i.e., Challenge 2). In this section, we investigate the aforementioned challenges and propose techniques to overcome them. Finally, we present PRIDE protocol that distributes IDS functions to the nodes accurately and fast (i.e., practical).

4.3.1 Memory Consumption Modeling

Experimentally, we observed that when activating multiple *small* rule files (i.e., containing at most 50 detection rules), Snort memory load is much less than the sum of individual memory loads. However, we observed that when multiple *large* rule files (i.e., containing more than 250 detection rules) were activated, the memory load is closer to the sum of the rule file's individual memory loads. When a rule file is activated, depending on: 1) the number of detection rules it has; 2) the preprocessors it activates (if already not activated); and 3) the Snort search method, a different amount of memory load will be imposed to the node. In this subsection we investigate how the aforementioned three factors impact our assumption about memory load linearity (i.e., Constraint 4.3).

Every Snort detection rule has the following structure:

```
[alert_type] [protocol] [src][src_port] ->
[dst][dst_port]:[Options][ContentMetaData][Operations].
```

The string patterns of each rule are organized in an automaton, which has a tree-like structure, thus, we expect a sub-linear behavior when activating new rules. Besides the strings, Snort keeps additional information per rule in its internal data structures, and this increases linearly with the number of rules. The metadata for each rule usually consumes more memory than the strings contained in the rule (most strings are small, and many rules do not even have string patterns). In order to investigate the linearity of memory load, we put all detection rules in a single rule file and then measured the memory load for different number of detection rules being enabled. Since in addition to preprocessor dependency, there exists a dependency between detection rules of each Snort rule file, we had to remove all dependencies (i.e., dependency relaxation) so that we could arbitrarily add/delete rules and change

the size of the file. For this, we removed all keywords that appear in the `options`. Algorithm 1 presents how the dependency relaxation is implemented.

Algorithm 1 Dependency Relaxation

1: $Pr = \{$set of all preproc. directives$\}$
2: $Rr = \{$set of all Snort detection rules$\}$
3: **while** $\exists pr \in Pr$ **do**
4: $\quad Kr \leftarrow GET\_KEYS(pr)$
5: **end while**
6: **for** $\forall r \in Rr$ **do**
7: $\quad H_r \leftarrow GET\_KEYS(r)$
8: $\quad$ **for** $\forall h \in H_r$ **do**
9: $\quad\quad$ **if** $h \in Kr$ **then**
10: $\quad\quad\quad RLX(r, h)$
11: $\quad\quad\quad$ **for** $\exists r' \neq r$ and $r' \rhd r$ **do**
12: $\quad\quad\quad\quad RLX(r, h, r')$
13: $\quad\quad\quad$ **end for**
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: **end for**

Given the set of Snort preprocessing directives and Snort detection rules, Algorithm 1 first creates two sets Pr and Rr (Lines 1, 2). Next, for each preprocessing directive in Pr, the Algorithm extracts a set of *keys* that are keywords dependent to the preprocessing directive (Lines 3-5). The extraction is based on our intimate/expert

knowledge about preprocessing directives. Next, for each rule in Rr it extracts all keywords seen in the rule (Line 7). Since one rule may depend on several preprocessing directives, the Algorithm examines each extracted keyword (Line 8) and checks whether it exists in set Kr or not. If it exists, the Algorithm removes the keyword from the rule (Lines 9, 10). The Algorithm also examines if any of the other rules have a dependency on the current rule r and its keyword h (Line 11). If so, the keyword will be also removed from rule r' (Line 12).

After all dependencies are removed, we can arbitrarily enable/disable each detection rule in the single large file. We group the rules in two ways: i) by simply concatenating their files ("regular" case) and ii) by shuffling them into the single file ("shuffled" case) and plot Snort's memory consumption as we increase the number of loaded rules in each case. We performed the experiment for the `ac-bnfa-nq` and `lowmem` search methods. Figures 4.4(a) and 4.4(b) depict the results for the `ac-bnfa-nq` and `lowmem` search methods, respectively. Thus, *irrespective of rule order and search method, we observe a linear behavior (consistent to our intuition, as explained above) when adding blocks of 250 rules to the set of active rules.* Although the string patterns from all rules are organized in a single automaton for fast string searching (which alone would result to a sub-linear memory consumption pattern) the observed linear behavior is due to other dominant rule-specific data that increase with the number of rules. Such data includes the descriptive message to be printed in the alert, reference numbers and identification codes, numerous other keywords like `rawbytes`, `byte_test`, and `pcre`, as well as other rule metadata. We use these findings to address the non-linearity of memory load for the variable-size rule files in the following subsection.

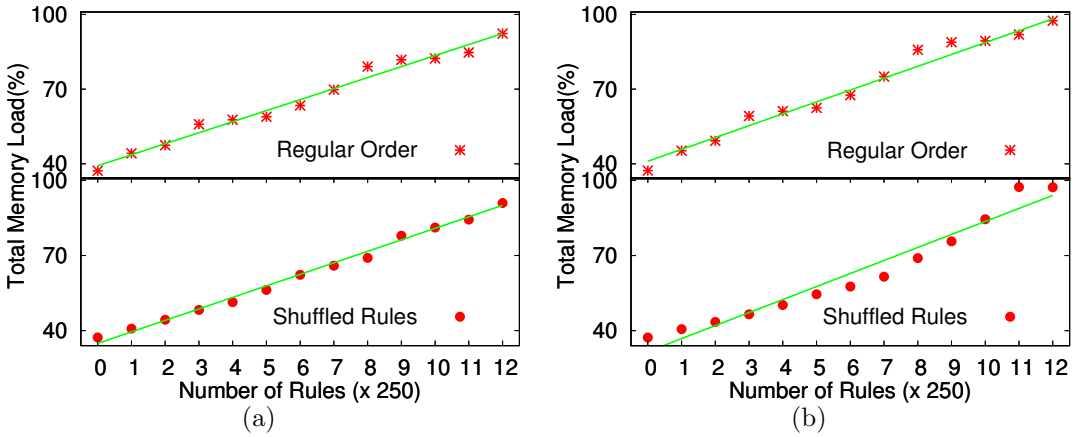


Figure 4.4: Linearity of memory consumption in different search algorithms as the number of activated rules increases: a) AC-bnfa-nq; b) Lowmem.

4.3.2 Rule Files Modularization

To reduce the complexity of the problem the ILP solver faces (i.e., Challenge 2), we propose to reduce the number of individual preprocessors and IDS functions, which would result in a decrease in the number of constraints in ILP. Our proposal is to group multiple IDS functions together and consider them as a single function. *From here on, we refer to each group of rule files as a "detection module" and use the term "group" for a group of preprocessors.* If a detection module is assigned to a node, all rule files in that module will be activated. We experimentally observed that grouping rule files not only reduces the problem complexity (Challenge 2), but also decreases the variance in memory load estimation (Challenge 1). When several small rule files are grouped in a single detection module, it acts as a larger rule file (same as a block of 250 rules), and the estimated memory load is more accurate. In addition, considering the *preprocessor dependency* mentioned in Section 4.2, an efficient rule file grouping reduces the number of preprocessor dependencies. For example, if two rule files require the same preprocessor(s), they can be grouped in

the same detection module. Similarly, multiple preprocessors required for the same rule files, can be grouped together. Hence, when activating a new *detection module*, the load imposed by rules' data structure dominates the load imposed by the new activated preprocessor (that can be ignored). This is very similar to the behavior observed in Figures 4.4(a) and 4.4(b) in the absence of preprocessors.

Grouping rule files together, however, has a disadvantage when the memory threshold set by the system administrator is very low. For low memory thresholds, we cannot assign larger modules to nodes, which results in low coverage/detection ratio. Consequently, despite the positive aspects of grouping small rule files together, memory threshold forces us to avoid large detection modules. Unfortunately, there already exist large detection modules. For example, the memory space required by the "backdoor" rule file is twice the memory space required by a detection module with 25 small rule files. This illustrates the need to also split extremely large rule files into some smaller ones (i.e., creating several detection modules out of a large rule file).

We thus define "modularization" as the procedure that, for a given set of IDS functions (e.g., Snort rule files), i) *groups* small IDS functions together in order to reduce the problem complexity and load estimation error, and ii) *splits* large IDS functions into several smaller functions so that they can be activated with low memory thresholds.

4.3.2.1 Rule File Splitting

When splitting a rule file, we consider the dependency between detection rules and the dependency between preprocessors and detection rules. This is to ensure that two dependent rules along with all of their essential preprocessing directives are included in the same split rule file. In order to split a rule file into several

detection modules, we first pre-parse each detection rule and specify its preprocessing dependency in advance. For example, a detection rule using TCP traffic match (i.e., protocol:TCP) requires the Stream5 preprocessor directive, which enforces all HTTP-relevant rule files to also contain same directive. We summarize all these preprocessing dependencies before splitting the rule files.

In addition, rule dependency is expressed by the options' keywords, e.g., "flowbits." To meet the rule dependency requirements, we parse each detection rule and specify whether the rule contains such keywords or not, if it does, it must be relevant. For example, the "flowbits" options can help us maintain the stateful check in a set of Snort detection rules. When some keys are set by "flowbits" in a detection rule, every other detection rule which does not set the "flowbits," is dependent on that detection rule. Similarly, the keyword "rev:VALUE" in a detection rule, that identifies revisions of Snort rules, denotes that it is related to a detection rule whose "sid" is "VALUE." Thus, using these two types of dependency, we split large rule files properly.

4.3.2.2 Proposed Modularizations

We propose three modularizations with different numbers of detection modules and different sizes. We then compare the execution time of the solver, i.e., Matlab ILP solver, for each modularization.

In the first modularization, the entire set of Snort rule files is classified into 23 detection modules where 6 different groups of preprocessors are required. The average memory load of the 23 detection modules is 3.98% and the standard deviation is 1.68%. The second modularization consists of 12 detection modules of average memory load 6.76% and standard deviation 2.31%, while the third modularization has only 6 detection modules of average memory load 15.06% and standard deviation

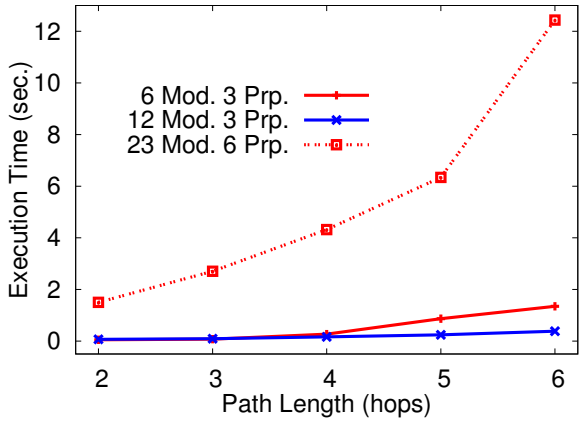


Figure 4.5: ILP execution time for different modularizations.

1.88%. The second and the third modularizations require three groups of preprocessors. Figure 4.5 shows the execution time of the ILP solver when solving the problem for different lengths of a single path. As depicted in Figure 4.5, 12-module and 6-module configurations are much faster than 23-module configuration, especially for longer paths (i.e., more complex problems). With these two modularizations, the ILP solver finds the optimal solution in less than 2 sec, which is very fast, thus practical in real deployments. The longer execution time for the 6-module configuration, comparing to the 12-module configuration, is because of its larger detection modules that increase the number of infeasible solutions for a given memory threshold (increasing the solver's execution time). Details about each modularization, e.g., the rule files in each module, are provided in Section 4.4. We use 6-module and 12-module configurations in our system evaluations.

4.3.3 PRIDE Protocol

Given a modularization chosen for the IDS configuration, PRIDE periodically collects the local information from the nodes, removes *idle* nodes from the network, i.e., those not contributing in the traffic routing, and optimally distributes IDS func-

tions to the nodes along traffic paths. If the reduced graph is disconnected, each graph component is considered as a sub-problem and solved separately. Algorithm 2 presents PRIDE protocol.

Algorithm 2 PRIDE Protocol

1: $Data\_Collection(V, E, N, Q)$
2: $Relaxation(V, E, n, q)$
3: $Path\_Extract(V, E, P)$
4: $\mathcal{P} = P$
5: $g = 0$
6: **while** $\exists\, p_i \in \mathcal{P}$ **do**
7: $\quad g++$
8: $\quad S_g = \{p_i\}$
9: $\quad \mathcal{P} = \mathcal{P} \backslash \{p_i\}$
10: $\quad$ **while** $\exists\, p_j \in Q$ and $\bigcup_{p_k \in S_g} (P_j \cap P_k) \neq \varnothing$ **do**
11: $\quad\quad S_g = S_g \cup \{p_j\}$
12: $\quad\quad \mathcal{P} = \mathcal{P} \backslash \{p_j\}$
13: $\quad$ **end while**
14: **end while**
15: **for** $\forall\, S_g$ **do**
16: $\quad V_g = \{v_j | v_j \in P_i \, and \, p_i \in S_g\}$
17: **for** $\forall\, V_g$ **do** ILP(V_g, P)

Given the set of nodes, the protocol first collects information from nodes and then produces the reduced sets V and E by removing idle nodes/links (Lines 1, 2). Next,

the set of active routing paths P is extracted in Line 3. Given P, the Algorithm creates the set $\mathcal{P}$ of unvisited paths (Line 4), and then defines variable g as the number of sub-problems (Line 5). For every unvisited path p_i in set $\mathcal{P}$ (Line 6), the Algorithm first creates a new sub-problem S_g (Lines 7, 8) and marks it as a visited path (Line 9). The Algorithm then searches $\mathcal{P}$ to find any unvisited path p_j which is *connected* (Two paths are *connected* if they are in the same component of the reduced graph) to at least one path in the current S_g (Line 10). If so, the corresponding path p_j will be added to the current sub-problem S_g and removed from $\mathcal{P}$ (Lines 11, 12). When no more paths in $\mathcal{P}$ can be added to the current S_g, the Algorithm increases g and creates a new sub-problem. This process repeats until there is no unvisited path in $\mathcal{P}$. Next, for every sub-problem S_g, the Algorithm creates the corresponding set V_g as the set of nodes located on the paths of component S_g (Lines 15, 16). Finally, the Algorithm runs ILP on the nodes and paths of each sub-problem S_g (Line 17).

4.4 System Implementation and Evaluation

In this section, we evaluate the performance of PRIDE in a department-wide mesh network. Our mesh network (as shown in Figure 4.6) consists of 10 Netgear WNDR3700 routers deployed in a $50 \times 30m^2$ rectangular area (Note: comparing with other testbeds, DistressNet 8 nodes [16], SMesh 14 nodes [4], and QuRiNet (a large-scale research platform) 30 nodes [92], PRIDE uses an average size testbed.). The presence of the walls in this area makes it a suitable environment for a multi-hop mesh network. Additionally, the "tx-power" parameter in the network configuration file of OpenWrt is used to adjust the communication range of the routers. The routers use OLSR as the routing protocol and provide mesh connections on their 5GHz wireless interfaces. In Figure 4.6, each node is labeled with its local subnet IP address. We will refer to the nodes using the third number in the subnet IP address, e.g.,

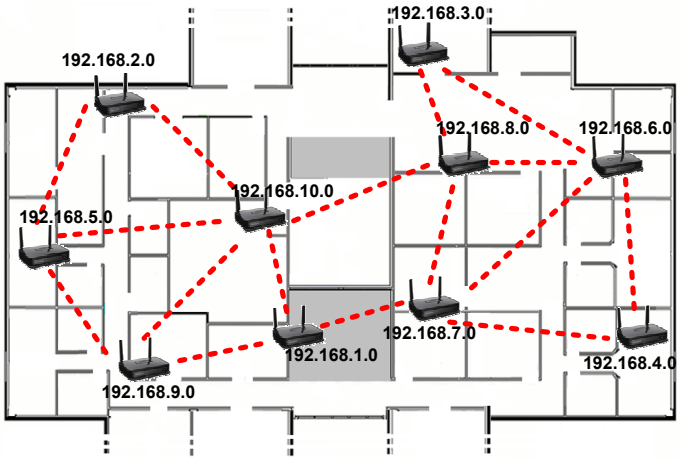


Figure 4.6: Our department-wide wireless mesh network.

192.168.5.0 is node 5. Some routers work as Mesh Access Points (MAP), e.g., node 3, and provide network access for the clients on the 2.4GHz wireless interface. Node 9 is the network gateway that connects WMN to the Internet. PRIDE periodically (i.e., 5 minutes in the current implementation) collects nodes/traffic information and runs ILP. This interval can be optimally chosen by administrator in dynamic networks.

We evaluate the *intrusion detection rate (coverage ratio)* and *average memory load* of nodes. The parameters that we vary are the *Path Lengths (PL)* and *memory threshold (λ)*. In all our experiments, the memory thresholds of all nodes are equal and some of the preprocessors (e.g., perfmonitor) are not used as they are not activated by default or not required by rule files. Since the maximum path length in our mesh network is 4 hops, we consider 2-hop, 3-hop and 4-hop paths ($PL = 2, 3,$ *and* 4). We consider two different paths for each given PL (six paths in total) in our evaluation. As listed in Table 4.1, for example, in the 2-hop scenario ($PL = 2$), $P_1 = \{5, 10\}$ and $P_2 = \{9, 10\}$, and in the 3-hop scenario, $P_1 = \{5, 10, 8\}$ and $P_2 = \{9, 10, 8\}$. The initial memory load on the routers is $\sim 30\%$ (as caused

Table 4.1: Different paths considered in the system evaluation

Path Length	Nodes	
2 hops	p_1: 5, 10	p_2: 9 ,10
3 hops	p_1: 5, 10, 8	p_2: 9 ,10, 8
4 hops	p_1: 5, 10, 8, 6	p_2: 9 ,10, 8, 3

by DHCP, OLSR, and other services). We vary the Snort memory threshold from 30% to 60% (i.e., $60\% \leq \lambda \leq 90\%$). Since implementing the related traffic-aware solution [70] on the mesh devices is infeasible (as shown in Section 4.1), we compare PRIDE with monitoring node solutions ([37,76]). We implement a monitoring node solution [37] to which we refer as "MonSol". A monitoring node loads detection modules up to a given memory threshold based on the indices of detection modules presented in Section 4.4.2. If a monitoring node monitores at least one link of a given path, the entire path is considered as monitored.

4.4.1 Intrusion Detection Evaluation Tool

We define the intrusion detection rate as the ratio between the number of detected attacks and the number of detectable attacks by all modules. For example, for the 12-module configuration, we ran 12 distinct attacks for each path where each attack can be detected by only one of the detection modules (i.e., because the corresponding detection rule is put in that module), and then measure the number of detected ones. To generate attack traffic, we modify an open source Snort test framework - the Rule to Attack (R2A) tool. R2A is a rule-based tool which parses each Snort detection rule and generates an exploitation packet targeting that rule. We modify the R2A's source code to generate real-time exploits for a given set of detection rules. The exploits are launched against the multi-hop remote target through wireless mesh links. If the module that can detect the attack is assigned to the nodes along the

path, then the attack is detected and relevant alerts are generated.

Table 4.2: Snort rule sets and modularizations used in our experiments and evaluations

Rule File	**Size**	**M6**	**M12**	**M23**	**Rule File**	**Size**	**M6**	**M12**	**M23**
community-nntp	1	1	1	2	shellcode	25	1	2	3
community-oracle	1	1	1	2	ddos	30	1	2	3
x11	2	1	1	2	pop3	35	1	2	3
pop2	2	1	1	2	specific-threats	36	2	4	8
community-icmp	2	1	1	2	web-frontpage	38	2	4	9
comm.-inapproprate	3	1	1	2	chat	42	1	2	3
other-ids	3	1	1	2	web-coldfusion	44	2	5	9
community-web-iis	4	2	4	8	community-bot	45	1	2	3
community-policy	4	2	4	8	voip	45	1	2	3
community-exploit	5	1	1	2	imap	60	1	1	3
comm.-web-attacks	5	2	4	8	misc	62	2	4	8
multimedia	5	2	4	9	policy	74	1	2	4
community-game	5	2	4	9	ftp	76	1	3	7
community-dos	6	1	1	2	sql	87	1	3	4
community-smtp	6	1	3	5	icmp-info	93	1	1	1
bad-traffic	6	1	1	2	smtp	94	1	3	5
community-sip	7	1	1	1	web-iis	95	2	5	9
community-web-client	8	2	4	9	web-client	135	2	5	9
community-imap	8	1	1	2	web-php	142	2	5	9
comm.-sql-injection	9	2	4	9	rpc	168	1	3	4
community-virus	10	2	4	9	comm.-web-misc	190	2	5	10
info	10	1	1	2	exploit	208	2	5	10
scan	12	1	1	2	oracle	310	2	6	11
finger	13	1	1	2	web-cgi	357	2	6	11
rservices	13	1	1	2	web-misc	370	3	6	12
comm.-web-cgi	13	2	4	9	netbios	430	3	7	13
nntp	13	1	1	2	comm.-web-php	463	3	7	12
tftp	16	1	3	7	web-activex	587	3	8	14
snmp	16	1	1	2	backdoor-frag1	172	6	8	15
attack-response	17	1	1	2	backdoor-frag2	172	4	9	16
telnet	19	1	3	6	backdoor-frag3	172	4	9	17
dos	20	1	1	2	backdoor-frag4	171	6	10	18
porn	21	1	1	2	spyware-put-frag1	196	4	10	19
dns	22	1	1	2	spyware-put-frag2	196	5	11	20
mysql	22	1	1	2	spyware-put-frag3	195	5	11	21
icmp	22	1	1	1	spyware-put-frag4	195	5	12	22
p2p	23	2	4	8	spyware-put-frag5	195	6	12	23
community-misc	24	2	4	8					

4.4.2 Snort Rule Sets and Modularizations

The Snort rule files we used in our experiments and evaluations are shown in Table 4.2. The second column, namely "Size", specifies the number of detection

rules in each rule file. Columns "M6", "M12", and "M23" specify the index of the detection module that each file belongs to, when that modularization is used. For example, in the 6-module configuration, "community-nntp", "shellcode", and "community-smtp" rule files are put in the first detection module, however, in the 12-module configuration, "community-nntp" belongs to the first detection module, "shelcode" belongs to the second one, and "community-smtp" belongs to the third detection module. As depicted in Table 4.2, "backdoor" and "spyware-put", as two very large rule files, are split into 4 and 5 smaller files, respectively. The 23-module configuration uses less rule files in each detection module. For example, only "community-sip", "icmp", and "icmp-info" rule files belong to the first module in 23-module configuration. Similarly, for large rule files such as "backdoor" and "spyware-put" which are split two nine detection modules, we can see that each rule file fragmentation is considered as an individual detection module in the 23-module configuration.

Table 4.3: Load of detection modules in different modularizations

Config.	ID	Mem.(%)	ID	Mem.(%)
6-Module	M1	13.32	M4	17.33
	M2	14.66	M5	14.66
	M3	13.04	M6	17.33
12-Module	M1	3.4	M7	6.99
	M2	5.14	M8	9.57
	M3	3.75	M9	9.03
	M4	4.52	M10	9.53
	M5	5.49	M11	8.77
	M6	6.13	M12	8.81

Our modularizations are based on the size of the rules files and also preprocessor dependencies, such that the memory loads of detection modules are roughly same and the rule files in each detection module require the same preprocessors. The

amount of memory load caused by each detection module in 6-module and 12-module configurations are shown in Table 4.3. As depicted, the average memory load of detection modules in the 6-module configuration is much higher than the 12-module configuration. We omit the details about memory loads in the 23-configuration here, due to space constraint.

Table 4.4: List of Snort rule files used by the R2A tool for generating exploit against each detection module and the number of generated alarms by each module

Mod. ID	Rule File	#Alarms	Mod. ID	Rule File	#Alarms
M1	other-ids	3	M7	community-web-php	2
	dns	17		netbios	15
M2	ddos	16	M8	backdoor-frag1	33
	chat	33	M9	backdoor-frag2	51
M3	rpc	9		backdoor-frag3	36
	telnet	4	M10	backdoor-frag4	31
M4	p2p	18		spyware-put-frag1	30
	misc	26	M11	spyware-put-frag2	32
M5	exploit	10		spyware-put-frag3	31
M6	oracle	4	M12	spyware-put-frag4	44
	web-cgi	3		spyware-put-frag5	29

Depending on the rule files and modularizations we use, the attacks chosen for intrusion detection evaluation may change. In order to evaluate the detection rate of PRIDE, we choose one or two rule files from each detection module and give them to the R2A tool as the input file. We also provide the IP address of multi-hop targets for R2A so that the attack exploits (malicious traffic) will be sent to the target through a multi-hop network path. Upon running each attack, the detection modules distributed along the path generate corresponding intrusion detection alarms. Table 4.4 specifies the rule files chosen from each detection module of the 12-module configuration and also the number of alarms generated by the corresponding detection module. It is worth mentioning that the same files are used for the 6-module

configuration. For example, in order to generate exploit(s) against module 1 in the 6-module configuration, all rule files in module 1 and module 2 of Table 4.4 are used, as they are all grouped in the module 1 of the 6-module configuration (according to Table 4.2).

4.4.3 Proof-of-Concept Experiment

When assigning IDS functions to multiple nodes on a path, each node can detect only a subset of attacks depending on the detection modules it executes. As a proof-of-concept experiment, we show that distributing two IDS functions to two nodes generates exactly the same alerts as if both detection modules were assigned to a single node (e.g., MonSol). For that purpose, we used two routers and one laptop connected wireless to each router (one laptop was the attacker and the other was the target). We ran a customized Snort on each router (monitoring the mesh traffic) ensuring that every Snort rule file is activated on at least one of the routers. We then generated two R2A exploits such that their corresponding rule files, e.g., "dos.rules" and "exploit.rules", were activated on routers 1 and 2, respectively. When running attacks, the Snort on node 1 generated 4 alerts, while the one on node 2 generated 10 alerts (real-time detection, unlike cooperative IDS). We repeated the experiment where only node 1 was running Snort and both rule files were activated on node 1 (many other rule files were deactivated due to memory constraint). In this experiment, node 1 generated exactly the same 14 alerts upon launching the same exploits. Hence, we have shown that PRIDE can distribute IDS functions to nodes along a path such that network packets are inspected by all IDS functions.

4.4.4 Effects of Memory Threshold and Path Length

Given the network paths in our test-bed mesh network, we evaluate the intrusion detection rate of PRIDE and the average memory load on nodes, using 6-module

and 12-module configurations. For each modularization, we change λ and PL as our evaluation parameters to see their effects on PRIDE performance. Given a memory threshold, we show PRIDE can achieve higher detection rate than MonSol.

Figure 4.7 shows the effect of memory threshold and path length on intrusion detection rate and average memory load on the nodes when using the 6-module configuration. As depicted in Figure 4.7(a), maximum detection rate for MonSol is 50% which occurs when $\lambda = 90\%$. However, PRIDE can achieve 100% detection rate even in a lower memory threshold (e.g., at $\lambda = 80\%$ for $PL = 4$ and $PL = 3$). This is because more than one node is assigned with IDS functions and packets are inspected by more detection modules. In this modularization, for a low memory threshold (e.g., $\lambda = 60\%$), only module 3 can be activated on the nodes, and thus, PRIDE cannot achieve a higher detection rate than MonSol. Figure 4.7(b) depicts the average estimated memory load on the nodes for different memory thresholds. It can be observed that PRIDE usually impose less memory load than MonSol, especially for the longer paths, since the modules are distributed to multiple nodes.

The results for the same evaluations performed on the 12-module configuration are shown in Figure 4.8. As depicted in Figure 4.8(a), the intrusion detection rate for the 12-module configuration is higher than the detection rate for the 6-module configuration (for the same memory threshold). This is because the size of the detection modules in the 12-module configuration is smaller than for the 6-module configuration, which allows more modules to fit in the small free memory spaces. In contrast with the 6-module configuration, where at low memory thresholds the detection rate was similar to MonSol, in the 12-module configuration the detection rate at 60% (a low memory threshold) is higher than for MonSol. This is because more modules are activated on the nodes even at this low memory threshold. The average estimated memory loads for this modularization are shown in Figure 4.8(b).

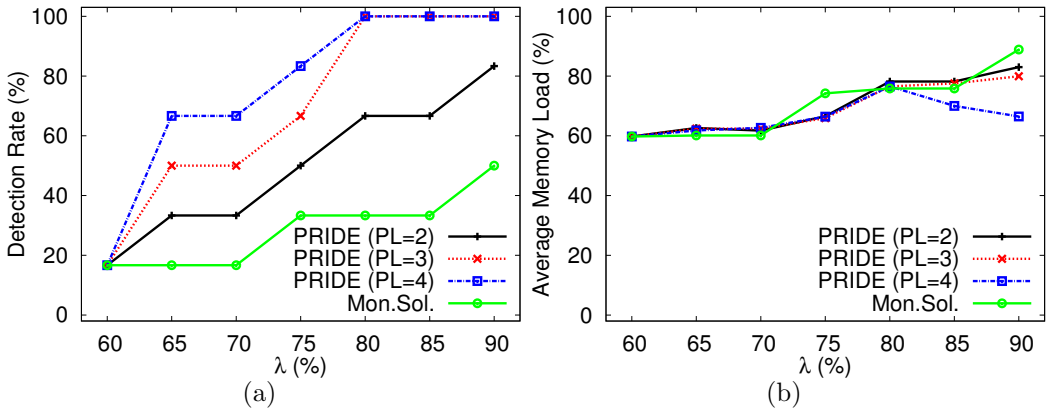


Figure 4.7: 6-module configuration: effect of λ and PL on a) Intrusion detection rate, and b) Average estimated memory load.

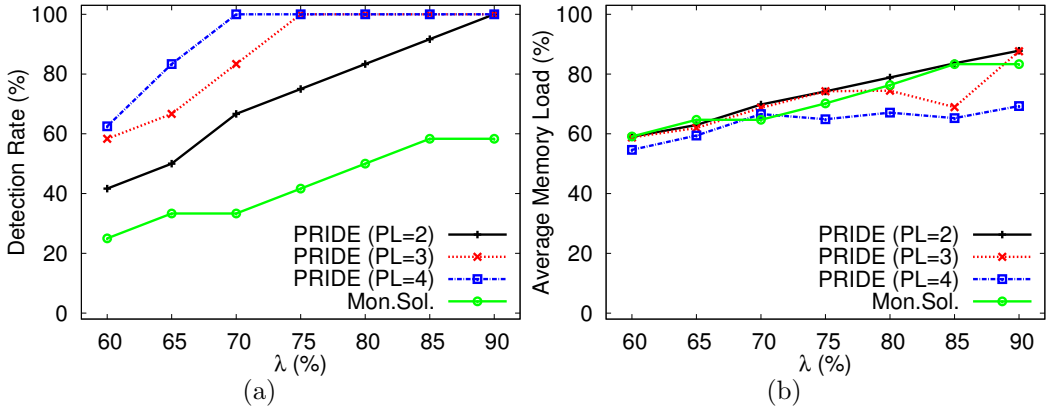


Figure 4.8: 12-module configuration: effect of λ and PL on a) Intrusion detection rate, and b) Average estimated memory load.

Similar to the 6-module configuration, it is observed that the 12-module configuration usually impose less memory load than MonSol solution for the longer paths.

When considering PRIDE and MonSol, one can observe that for an adversary it will be significantly harder to compromise multiple IDS nodes (as in PRIDE), than a single monitoring node (as in MonSol). A compromised IDS node in PRIDE implies the loss of few IDS functions while, in MonSol, it means the loss of the entire set of activated functions on the monitoring node (i.e., higher vulnerability). Hence,

in addition to achieving higher detection rates and lower memory loads, PRIDE provides a higher degree of IDS attack tolerance than MonSol.

We also compare the estimated memory loads and the actual memory loads of the two configurations in all of the experiments, i.e., different memory thresholds and path lengths. Figures 4.9(a) and 4.9(b) show the difference between estimated memory load and actual load measured on the routers when using 6-module and 12-module configurations, respectively. One can see that the difference is below $\sim$5%, thus giving confidence in our ILP formulation and memory consumption modelling. It is also worth mentioning that the estimated values for the 12-module configuration are closer to the real values than the 6-module configuration because the modules are roughly the same size as 250-rule blocks.

Figure 4.10 shows the ILP solver execution time for $PL = 3$ and $PL = 4$, and for each modularization. As depicted, the execution time of the algorithm ranges from a few seconds to tens of seconds, thus making it practical for real world deployments. As shown, the lower the memory threshold is, the longer the execution time is. This is because lower memory thresholds increase the number of infeasible solutions and the solver requires more iterations to obtain feasible and optimal solutions. As shown in Figure 4.10, the execution time increases with the path length as well. As mentioned in Section 4.3, this is because the number of ILP constraints (i.e., the problem complexity) is a direct function of path length.

Although we showed that PRIDE can achieve high detection rates by solving an ILP in less than a minute, one may argue about the communication overhead caused by the message exchange between nodes and the base station. For example, a question that arises here would be "Is it possible for the nodes along each path to randomly choose an IDS configuration and still achieve reasonable intrusion detection (path coverage) rates for WMN paths?" It is obvious that such a distributed

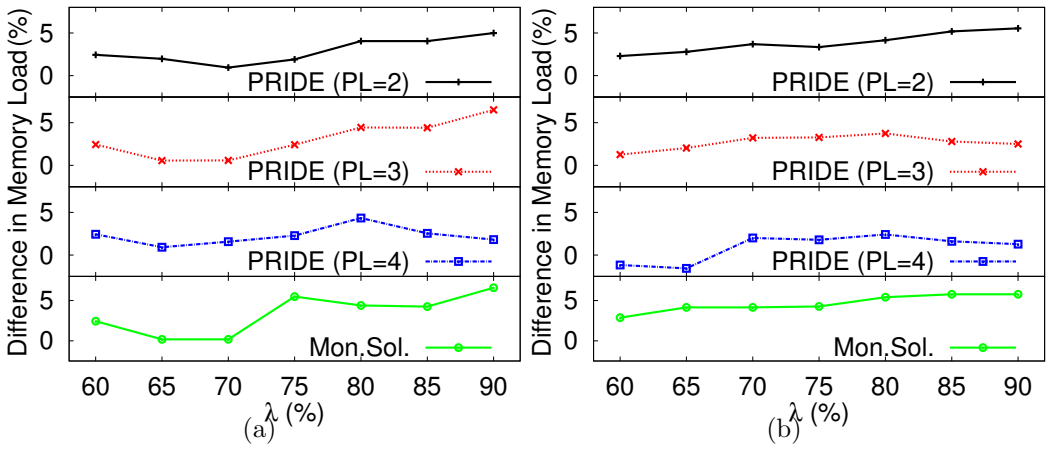


Figure 4.9: The difference between estimated and actual average memory load: a) 6-Module configuration, and b) 12-Module configuration.

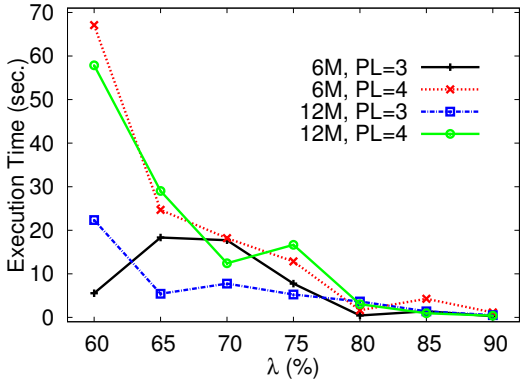


Figure 4.10: ILP solver execution time for different parameters.

approach will provide a locally optimal solution since the decision is made based on the local information available on each node, however, centralized approaches (e.g., PRIDE) make decision based on global information about WMN routing paths that results in optimal solutions. In order to compare PRIDE with an ad-hoc IDS Function Distribution mechanism, we implemented a distributed mechanism where each active node, based on its memory threshold, randomly chooses the set of IDS functions (i.e., detection modules) it can perform. The average path coverage rates for

100 random solutions produced for the WMN topology and its corresponding paths shown in Figure 4.6 are depicted in Figure 4.11. The results show that PRIDE outperforms distributed approach in most of situations, especially when there is enough memory for activating multiple detection modules. It is worth mentioning that although PRIDE achieves higher detection rates at the price of higher communication overhead, it is very important to achieve 100% detection rates (at higher prices) in mission critical scenarios and a sub-optimal solution will not be accepted as an applicable solutions.

4.4.5 PRIDE-aware Attacks

This section evaluates PRIDE's performance for PRIDE-aware attacks. We categorize PRIDE-aware attacks in two levels of severity: 1) the attacker is aware of PRIDE in the WMN but cannot compromise the secure communication between nodes and the base station (Level 1); 2) the attacker is aware of PRIDE and also the content of secure information exchanged between nodes and the base station (Level 2). Obviously, the later type of attack is more severe and very difficult to defend. In fact, the second attack type assumes that the attacker has broken the secure communication link between the base station and all routers and has access to all information (i.e., memory loads and traffic paths) and the IDS distributions. Unlike our attacker model presented considered for PRIDE, we assume that a PRIDE-aware attacker can launch an attack against an *intermediate* node in a traffic path (not necessarily the multi-hop *destination* on the path). This type of attack sounds reasonable because a PRIDE-aware attacker aims to compromise some intermediate nodes running specific detection modules, and finally attack the destination.

We concentrate on Level-1 attack because the possibility of running Level-2 attack in WMN depends on the robustness of key distribution and also encryption

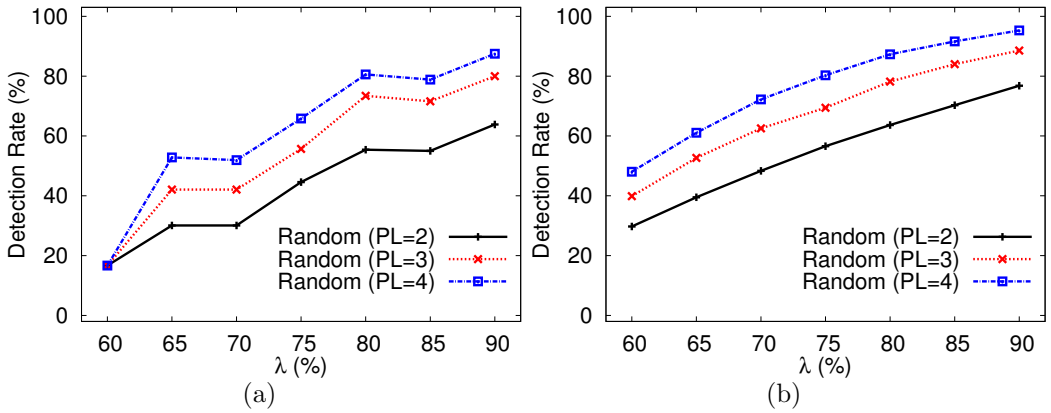


Figure 4.11: Distributed approach based on random module selection: effect of λ and PL on the intrusion detection rates in a) 6-Module Configuration, and b) 12-Module Configuration.

mechanisms used in wireless networks, which is out of our scope in this research. In Level-1 attack, the attacker can not produce the same IDS distribution (to find the most beneficial node to be compromised) as the base station produces. This is because the ILP solutions depend on nodes information (securely sent to the base station) and the initial random solutions. Thus, we consider an attacker that knows WMN nodes are assigned *some* IDS functions but does not know which node is running which module. The attacker first connects to an AP, then chooses a node (destination or an intermediate) as the target and a *random* type of attack (i.e., an attack among those detectable by 6 or 12 modules), and finally lunches the attack. It is obvious that the average detection rate for the PRIDE-aware attacks against destination nodes is always equal to the PRIDE coverage ratio (i.e., as shown in Figures 4.7(a) and 4.8(a)). Hence, we only consider attacks against intermediate nodes.

We perform an experiment to evaluate the PRIDE performance (detection rate) when a PRIDE-aware attacker at Level 1 runs attacks against intermediate nodes. Considering the IDS distributions produced for our real-world WMN, with the detec-

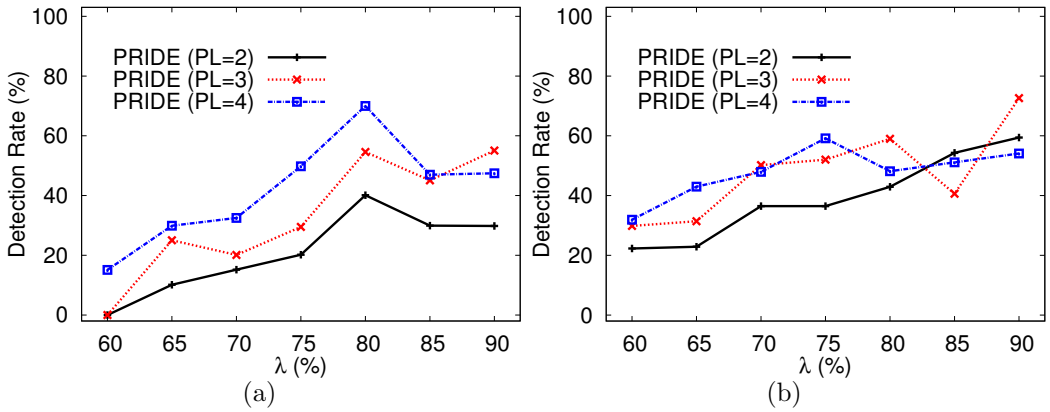


Figure 4.12: Average detection rate for PRIDE-aware attacks: a) 6-Module configuration, and b) 12-Module configuration.

tion rates shown in Figures 4.7(a) and 4.8(a), we ran 1000 random attacks for each modularization. In each of 1000 attacks, the attacker chooses a random intermediate node and a random attack (among those detectable by 6 or 12 modules). If the corresponding detection module is not activated on the nodes along the path (starting from the attacker AP towards the intermediate node) the attack cannot be detected, otherwise it is detectable. Figures 4.12(a) and 4.12(b) show the average detection rate for Level-1 PRIDE-aware attacks against intermediate nodes, for 6-module and 12-module configurations, respectively. As depicted, the detection rate increases as the λ increases. We observe that the 12-module configuration has a higher detection rate than the 6-module configuration (for the same reason we explained about average detection rate in PRIDE). One may argue about the detection rate for (PL=4) in the 6-module configuration that decreases at higher memory thresholds. This is because we perform all 1000 random attacks against a specific IDS distribution used in our real test. In that experiment, for higher thresholds, most of the IDS modules were assigned to the destination, thus, intermediate nodes ran less IDS modules and were unable to detect attacks. Same argument applies for (PL=3) in the 12-module

configuration.

5. RESOURCELESS AND TRAFFIC-AGNOSTIC IDS

The research presented in this section* is motivated by the fact that in many WMN applications traffic paths change very often, which consequently degrades the performance of traffic-aware IDS solutions. For example, routing paths in large scale WMN that provide networking services for mobile clients are subject to change due to client mobility. Additionally, WMN topology, especially in outdoor deployments, may change due to node failures or drastic link-quality changes. Hence, the traffic knowledge has to be very accurate and up-to-date in traffic-aware solutions, which is not always feasible. In this section, we propose two traffic-agnostic intrusion detection mechanisms for resource-constrained WMN that monitor all communication links, instead of only few paths. Such an approach in WMN IDS is traffic-independent, but requires more mesh nodes to participate in detection mechanism.

5.1 Non-Cooperative IDS

This section introduces RAPID, an IDS based on traffic-agnostic and link-coverage approaches that irrespective to the changes in WMN traffic paths, is able to monitor the entire WMN traffic, at the price, however, of putting IDS load on all WMN nodes instead of those located only along routing paths. In RAPID, each node, depending on its available resources, is assigned a subset of IDS functions, i.e., a customized IDS configuration, and investigates the entire network traffic on the set of communication links it can monitor (i.e., in its coverage area). This customized IDS allows resource conservation on resource-constrained WMN nodes and also increases the probabil-

*Parts of this section are reprinted with permission from "On the optimality of cooperative intrusion detection for resource constrained wireless networks" by Amin Hassanzadeh and Radu Stoleru, Computers & Security (Elsevier), Volume 34, pages 16 - 35, 2013, Copyright 2013 by Elsevier.

ity of monitoring a WMN link with multiple distinct IDS functions activated on all WMN nodes that can monitor the link. It is worth mentioning that for a given network size, the complexity of traffic-agnostic solution is larger than traffic-aware solution as it needs to find optimal IDS function distribution for all nodes. Hence, RAPID has to be fast and scalable.

5.1.1 Goals and Features

PRIDE considers static resource-constrained WMN where network topology does not change often (compared to other ad hoc networks). It assumes that network information periodically collected by the base station reflects the most recent network topology. However, research has shown [14, 16, 64] that even static WMN topology and routing paths are subject to change due to: a) link-quality variations caused by weather, noise and other radio signals, etc.; b) mobility of clients and their requested services that result in changes of WMN routing paths; c) node failure (e.g., running out of power) or node replacement (e.g., administrative reasons) during network lifetime. Hence, traffic awareness might be a strong assumption for many WMN applications. Motivated by this fact, we propose a traffic-agnostic IDS solution.

PRIDE is not a *scalable* solution because its execution time (i.e., to find optimal IDS function distribution for WMN nodes) significantly increases when network size, number of paths, and number of IDS functions increase, or when the memory threshold on the nodes decreases. The results shown in [39] are for a 10-node WMN for only 2 paths (for each given path length). When applied to a larger network (e.g., 30 nodes and 15 paths), however, it takes more than an hour to obtain the optimal IDS function distribution. Thus, a practical IDS solution must be able to quickly produce optimal results when used for large scale WMN. We note here that the traffic-agnostic solution, proposed here, has to solve a more complex problem

because all WMN nodes perform IDS operations. Therefore, we need to develop an algorithm that can produce optimal IDS function distribution for a large WMN in a short period of time.

PRIDE only considers *multi-hop* attacks which means the attack traffic (malicious packets(s)) is routed across multiple nodes (i.e., at least one WMN backbone link). In addition, the experimental results [39] show that the longer the path is, the higher the detection rate will be. We aim to design an IDS that can detect both single-hop attacks (i.e., both attacker and target are clients connected to same router) and multi-hop attacks, routed through short paths (e.g., 2 hops).

PRIDE proposes a centralized algorithm that requires periodic data collection from WMN nodes and a computationally powerful base station to produce the optimal IDS function distribution. In this research, we propose an IDS solution that can also be implemented in a distributed manner where WMN routers independently choose the optimal set of IDS function to perform. The distributed approach is based on random IDS function selection by the nodes that incur no communication overhead (caused by data exchanges between nodes and the base station). It also no longer requires a computationally powerful base station. We show that random IDS function selection surprisingly achieves near optimal network coverage ratios especially for high density WMN.

PRIDE uses a node-coverage approach, which means that only nodes along each routing path participate in traffic monitoring. However, RAPID uses link-coverage approach to achieve a higher link/path coverage ratio in WMN. Hence, in addition to the nodes located on each routing path, other nodes can also participate in traffic monitoring if they can monitor at least one link of that path. We use a link-coverage approach in our proposed IDS and show how it increases the link/path coverage ratio.

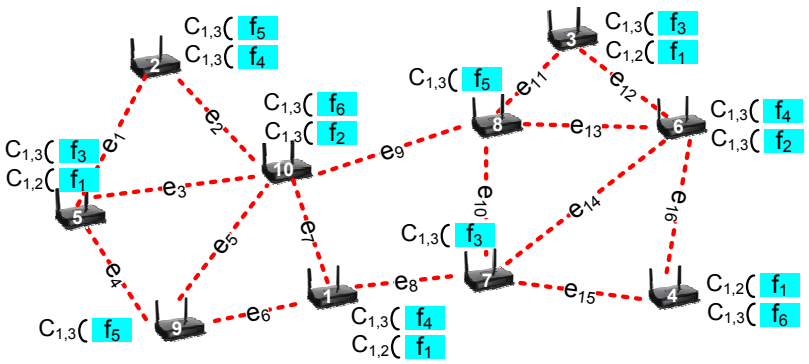


Figure 5.1: A WMN graph, consisting of 10 nodes and 16 links. As shown, a 6-module configuration is used in this WMN where Snort preprocessors are also grouped in three sets of preprocessors [38]. The nodes run different Snort configurations, e.g., node v_1 runs detection modules f_1 and f_4, which require preprocessors c_1, c_2 and c_3.

5.1.2 Preliminaries

Given a wireless mesh network, we denote the number of its nodes and number of its links by n and q, respectively. We model the wireless mesh network as a graph $G = \{V, E\}$, where V is the set of mesh nodes (routers) $\{v_1, v_2, \cdots, v_n\}$, and E is the set of backbone links $\{e_1, e_2, \cdots, e_q\}$. An example of such a graph, is shown in Figure 5.1 where $V = \{v_1, v_2, ..., v_{10}\}$ and $E = \{e_1, e_2, ..., e_{16}\}$. Figure 5.1, represents the network graph a real-world WMN deployed over the floor of a building. We denote by matrix $\mathbb{M}_{q \times n}$ the mapping between nodes and links, i.e., $m_{ij} = 1$ iff node v_j can monitor link e_i. Based on the *link-coverage* definition [37], v_j can monitor e_i if e_i is incident to v_j or v_j is connected to the two end points of e_i. The set of all links that can be monitored by node v_j is called *Covering Set* of node v_j represented by CS_j [37]. Accordingly, we denote by MS_i the set of all nodes that can monitor link e_i, i.e., *Monitoring Set* of link e_i. For the example shown in Figure 5.1, the matrix $\mathbb{M}$ is as follows:

$$\mathbb{M}_{16\times 10} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ \vdots & \vdots & & & & \cdots & & & & \vdots \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

We denote the set of all IDS functions (detection modules) by $\mathcal{F} = \{f_k \,|\, f_k$ is a set of detection rules $\}$ with size K (i.e., $|\mathcal{F}| = K$) where $K = 6$ in 6-module configuration and $K = 12$ in 12-module configuration. We also denote the set of IDS preprocessors (as in Snort) by $\mathcal{C} = \{c_r \,|\exists\, f_k \in \mathcal{F}$ that requires $c_r\}$ of size R (i.e., $|\mathcal{C}| = R$) where $R = 3$ in both 6-module and 12-module configurations. For the example presented in Figure 5.1, $\mathcal{F} = \{f_1, f_2, ..., f_6\}$, i.e., 6-Module configuration is used, and $\mathcal{C} = \{c_1, c_2,$ and $c_3\}$. The dependency between IDS functions and preprocessors is stored in matrix $\mathbb{D}_{K\times R}$ where $d_{kr} = 1$ means that activation of module f_k requires the activation of preprocessor c_r. For the example shown in Figure 5.1, the matrix $\mathbb{D}^T$ is as follows:

$$\mathbb{D}^T_{3\times 6} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Let $w : \{\mathcal{F}, \mathcal{C}\} \longrightarrow [0, 1]$ be a cost function that assigns memory load w^f_k and w^c_r to detection module f_k and preprocessor c_r, respectively. Consequently, vectors $W^f = [w^f_1, w^f_2, \cdots, w^f_K]$ and $W^c = [w^c_1, w^c_2, \cdots, w^c_R]$ represent memory loads for the detection modules in $\mathcal{F}$ and for the preprocessors in $\mathcal{C}$, respectively. Considering the 6-module configuration in PRIDE, for the configuration used in Figure 5.1,

$W^f = [13.3\%, 14.6\%, 13\%, 17.4\%, 14.6\%, 17.3\%]$ and $W^c = [15.6\%, 1.1\%, 1\%]$. It is worth mentioning that $w_1^c = 15.6\%$ is the total load caused by Snort base line, *stream5* (both static and dynamic loads as explained in PRIDE), and *frag3* - the most common and required Snort preprocessors for all detection modules [38]. We denote by $B = [b_1, b_2, ..., b_n]$ the base memory load (i.e., before performing IDS) of all nodes. Finally, the maximum allowable memory load (after detection modules and preprocessors are loaded) is represented by vector $\Lambda = [\lambda_1, \lambda_2, \cdots, \lambda_n]$, (*also called Memory Threshold*). Vector Λ depends on the memory space required by active services in WMN, and it is typically set by the security administrator.

5.1.3 Problem Formulation

The main objective of our proposed IDS is to monitor all WMN links using the maximum allowable number of detection modules that can be performed on WMN nodes (i.e., activated and executed by Snort on nodes). A higher number of detection modules executed by node v_j means more attack traffic can be detected on the links in CS_j. Thus, our IDS solution aims at assigning Snort detection modules on the WMN nodes, such that all of WMN links are monitored by the maximum number of modules and none of the nodes is overloaded. In order to mathematically formulate this problem, we first introduce several definitions.

Definition 4 ***IDS Function Distribution****,*
represented by $T = \{(v_j, \mathcal{F}_j, \mathcal{C}_j) | \, v_j \in V, \mathcal{F}_j \subseteq \mathcal{F}, and \, \mathcal{C}_j \subseteq \mathcal{C}\}$*, is a distribution of detection modules and preprocessors in the WMN, such that modules* $\mathcal{F}_j$ *and their corresponding preprocessors* $\mathcal{C}_j$ *are assigned to node* v_j *(i.e., they will be activated on the customized Snort executed on* v_j*).*

After the *IDS Function Distribution*, the set of detection modules and preprocessors assigned to WMN nodes are represented by binary matrices $\mathbb{X}_{n \times K}$ and $\mathbb{Z}_{n \times R}$,

respectively. Accordingly, $x_{jk} = 1$ means module f_k is activated on node v_j and $z_{jr} = 1$ implies that preprocessor c_r is activated on node v_j (i.e., there is at least one module assigned to node v_j that requires preprocessor c_r). For example, the *IDS Function Distribution*, and matrices $\mathbb{X}$ and $\mathbb{Z}$ for the example given in Figure 5.1 are:

$$T = \{(v_1, \{f_1, f_4\}, \{c_1, c_2, c_3\}), (v_2, \{f_4, f_5\}, \{c_1, c_3\}), ..., (v_{10}, \{f_2, f_6\}, \{c_1, c_3\})\},$$

$$\mathbb{X}_{10\times 6} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & & \cdots & & \vdots \\ \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbb{Z}_{10\times 3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ & \vdots & \\ \\ 1 & 0 & 1 \end{bmatrix}.$$

The total memory load of node v_j, after the *IDS Function Distribution*, becomes $L_j = b_j + \Sigma_{c_r \in \mathcal{C}_j} w_r^c + \Sigma_{f_k \in \mathcal{F}_j} w_k^f$. Obviously, an *IDS Function Distribution* in which there is at least one v_j such that $L_j > \lambda_j$ is deemed infeasible because the load L_j is not allowed to exceed the threshold λ_j.

Definition 5 *For a given link e_i and its corresponding monitoring set MS_i,* ***Link Coverage Ratio (LCR)*** *is defined as $LCR_i = |U_i|/K$, where $U_i = \bigcup_{v_j \in MS_i} \mathcal{F}_j$ is the set of detection modules assigned to nodes that can monitor the link.*

Definition 6 *Link e_i is called* ***Fully Covered*** *if $LCR_i = 1$ ($U_i = \mathcal{F}$), i.e., for*

$\forall f_k \in \mathcal{F}$, $\exists\, v_j \in MS_i$ *assigned with* $\mathcal{F}_j$ *such that* $f_k \in \mathcal{F}_j$.

Definition 7 ***Link Coverage Problem (LCP)***

Given $G = \{V, E\}$, *vectors* W^f *and* W^c, *and matrix* $\mathbb{D}$, *find a distribution* $T = \{(v_j, \mathcal{F}_j, C_j) |\, v_j \in V and\, \mathcal{F}_j \subseteq \mathcal{F} and\, C_j \subseteq C\}$, *such that* $\frac{1}{q}\sum_{e_i \in E} LCR_i$ *is maximized and* $L_j \leq \lambda_j$, $\forall v_j \in V$.

LCP aims at maximizing the average link coverage ratio while ensuring that memory loads on nodes are below their memory thresholds.

Given matrices $\mathbb{M}$ and $\mathbb{X}$, we denote by matrix $\mathbb{Y} = \mathbb{M} \cdot \mathbb{X}$ the mapping between links and the modules activated on the monitoring set of the links, i.e., y_{ik} is in the range $[0, n]$. For example, $y_{ik} = 0$ means that module k is not activated on any of nodes in MS_i while $y_{ik} > 0$ implies that there is at least one node in MS_i running module k. According to the *LCR* (union of all $\mathcal{F}_j$ for $\forall v_j \in MS_i$), $y_{ik} > 0$ is equivalent to $y_{ik} = 1$ since both of them mean link e_i is monitored by detection module f_k (redundant modules do not count). Thus, we define function $BN : \{\mathbb{Y}\} \longrightarrow \{0, 1\}$ that converts y_{ik} to a binary value, i.e., if $y_{ik} = 0$, $BN(y_{ik}) = 0$, otherwise $BN(y_{ik}) = 1$. For the example shown in Figure 5.1, matrices $\mathbb{Y}_{16\times6}$ and $BN(\mathbb{Y}_{16\times6})$ are as follows:

$$\mathbb{Y}_{16\times6} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \mathbf{2} & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ \vdots & \vdots & & \cdots & & \vdots \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad BN(\mathbb{Y}_{16\times6}) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ \vdots & \vdots & & \cdots & & \vdots \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

The objective function of *LCP* is non-linear. This is because the link-coverage requires the non-linear function *BN*. Thus, unlike PRIDE, *LCP* cannot be formulated as an ILP. In addition to non-linearity, *LCP* is more complex than path coverage problem (as defined in PRIDE) for a given network. This is because the number of paths to be covered is usually less than the number of communication links in WMN [39]. Moreover, we aim for a scalable IDS solution that can be applied to large WMN (i.e., more links have to be monitored). Thus, we need to develop a technique to reduce the complexity of link coverage problem when compared to path coverage problem.

One can observe that matrix $\mathbb{D}$, for both 6-module and 12-module configurations [38], can be summarized as: i) every detection module requires the first group of preprocessors of size 15.6%; ii) every detection module requires either the second group of preprocessors (1.1% load) or the third group of preprocessors (1% load). We propose a *dependency relaxation* to run all three groups of preprocessors on every single node at the price of at most 1.1% extra load. Accordingly, the total memory load of node v_j, after the *IDS Function Distribution*, becomes $L_j = b_j + 17.7\% + \Sigma_{f_k \in \mathcal{F}_j} w_k^f$. However, it reduces the complexity of *LCP* when compared to path coverage problem in PRIDE.

Thus, *LCP* can be formulated as a non-linear optimization problem with integer (binary) variables as follows:

$$\text{maximize} \qquad \frac{1}{q}(\mathbf{1}^T \cdot \text{BN}(\mathbb{M} \cdot \mathbb{X}) \cdot \mathbf{1}) \tag{5.1}$$

$$\text{subject to:} \qquad B^T + (17.7)\mathbf{1}^T \ + \ \mathbb{X} \ \cdot {W^f}^T \leq \Lambda^T \tag{5.2}$$

$$x_{jk} \in \{0, 1\} \qquad\qquad , \forall j, k \tag{5.3}$$

where the objective function is to maximize the average link coverage ratio in the network; constraint 2 limits the memory load on every node v_j to be less than its memory threshold λ_j; and constraint 3 forces x_{jk} to be either 1 or 0 meaning node v_j is either running module f_k or not.

5.1.4 RAPID Protocol

In this section, we propose RAPID, a protocol to solve LCP in centralized and distributed manners. The centralized approach requires a base station that periodically collects nodes' information (e.g., network connectivity and memory utilization), solves LCP, and finally broadcasts IDS function distributions to the nodes. The distributed solution does not need the base station (i.e., nodes locally decide which detection modules they should run).

5.1.4.1 Centralized Solution

Given a modularization chosen by the security administrator for the IDS configuration (e.g., the 12-module configuration imposes higher execution time to the solver but is suitable for low memory thresholds [39]), the centralized RAPID periodically collects the local information from nodes, decides on an optimal set of detection modules to be executed by each node, and distributes them to the nodes. Since LCP has a non-linear objective function, linear constraints, and integer variables, we cannot use integer linear programming. Thus, we propose a Genetic Algorithm (GA), a popular and effective type of evolutionary algorithms.

GA starts with a set of *random* solutions and then derives better solutions using the Darwinian process of "survival of the fittest." The survival of the fittest process is iterative, and uses genetic operations, such as Selection, Crossover, and Mutation on the current set of solutions (from here on we will use "set of solutions" and "population" interchangeably). Selection gives the most fit solutions the chance

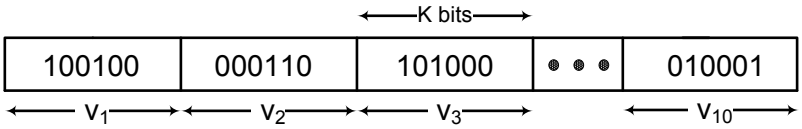


Figure 5.2: The matrix $\mathbb{X}$ for the 10-node mesh network shown in Figure 5.1 is encoded as a chromosome.

to survive. Crossover combines solutions in each generation to produce offsprings (i.e., new solutions) of the next generation, and mutation is used to maintain genetic diversity in two consecutive generations. GA solutions are encoded as bitstrings (i.e., chromosomes) of specific length and tested for fitness. In our formulation, matrix $\mathbb{X}$ is a solution that can be encoded as a chromosome of length $n \times K$. Figure 5.2 depicts the chromosome corresponding to the IDS function distribution (i.e., the solution represented by matrix $\mathbb{X}$) of the WMN shown in Figure 5.1. The fitness (objective) value of each solution is the average LCR in the network. The genetic operations used in this RAPID are based on operations explained in [34].

The centralized RAPID protocol is presented in Algorithm 3 as performed on the base station. Given the set of WMN nodes, the base station first collects information from nodes and then produces matrix $\mathbb{M}$ (Lines 1 and 2). Moreover, matrix $\mathbb{X}$, number of initial solutions (*POP_SIZE*) and number of generations (*GEN_SIZE*) are initialized in Line 2. Next, the base station generates a set of *POP_SIZE random* solutions called $S_{\mathbb{X}}$ (Line 3). Starting from the first population, the Algorithm then iteratively performs genetic operations and creates another population for the next generation (Lines 4-9). The Algorithm stops generating a new population if either the number of generations exceeds *GEN_SIZE* (Line 5) or the stopping criteria holds (Lines 10-12), i.e., no improvement in the recent α optimal values has been observed, where α is set by the network administrator. Algorithm 3 then extracts matrix $\mathbb{X}$ from the best solution in $S_{\mathbb{X}}$ of last generation (Line 15) and securely broadcasts the

Algorithm 3 Centralized RAPID

1: $Data\_Collection(V, E, n, q)$
2: $Initialization(\mathbb{M}, \mathbb{X}, POP\_SIZE, GEN\_SIZE)$
3: $Initial\_Solutions(POP\_SIZE, S_{\mathbb{X}})$
4: $g = 1$
5: **while** $g \leq GEN\_SIZE$ **do**
6: $Elitism(POP\_SIZE, S_{\mathbb{X}})$
7: $Selection(POP\_SIZE, S_{\mathbb{X}})$
8: $Crossover(POP\_SIZE, S_{\mathbb{X}})$
9: $Mutation(POP\_SIZE, S_{\mathbb{X}})$
10: **if** $Stopping\_holds(\alpha)$ **then**
11: $break$
12: **end if**
13: $g++$
14: **end while**
15: $\mathbb{X} = Best\_Sol(S_{\mathbb{X}})$
16: $Sec\_BRDCST(\mathbb{X})$

IDS functions to the WMN nodes (Line 16).

5.1.4.2 Distributed Approach

The main purpose of a distributed approach for RAPID is to remove the communication overhead caused by message exchange between nodes/base station and the computation overhead of running GA for large networks. Additionally, this approach is adaptive to frequent path and topology changes where the base station might not have the most recent routing information (unless the nodes' information is collected frequently, which might incur very high communication and computation overhead). Hence, in the distributed RAPID (presented in Algorithm 4), each node, depending on its memory threshold, chooses a set of *random* detection modules to perform.

As shown in Algorithm 4, Line 1, each node requires some preliminary information such as the set of modules and their corresponding memory weights, set of preprocessors, and memory threshold λ which are assumed to be already set on the

Algorithm 4 Distributed RAPID

1: $Mod\_Setting(\mathcal{F}, \mathcal{C}, K, R, W^f, W^c, b, \lambda)$
2: $L = b + \sum_{r=1}^{R} w_r^c$ // $\sum_{r=1}^{R} w_r^c = 17.7\%$
3: $Rand\_Perm(\mathcal{F}', \mathcal{F})$
4: **for** $f = 1$ to K **do**
5: $Mod = \mathcal{F}'(f)$
6: **if** $L + w_{Mod}^f \leq \lambda$ **then**
7: $Activate(\mathcal{F}, Mod)$
8: $L = L + w_{Mod}^f$
9: **end if**
10: **end for**

device by the security administrator. The base memory load b is obtained from system logs (Line 1) and added to the total memory load imposed by all preprocessors (Line 2). The algorithm then creates a new set of detection modules in a random order denoted by $\mathcal{F}'$ in Line 3. Next, detection modules in $\mathcal{F}'$ are iteratively checked (Lines 4-6) if they can be activated on the Snort configuration (depending on their memory weight and threshold λ). If so, the module will be activated and the total memory load L will be updated (Lines 7-8).

We will show that this approach works very well (produces near optimal solutions) and its performance surprisingly increases (i.e., achieves the centralized performance) in high memory thresholds or high network density. It is worth emphasizing that such a good performance is achieved without any communication overhead and with a very simple algorithm when compared to the centralized RAPID.

5.1.5 Performance Evaluation

In this section, we first demonstrate, through a proof of concept experiment using WMN hardware, that the ideas of link-coverage and multi-interface Snort are practical. Next, through extensive simulations, based on real data obtained from real-world WMN deployment and memory measurements, we evaluate the performance

of our proposed centralized and distributed RAPID solutions. The main reason we use simulation is to be able to evaluate RAPID's performance for large networks and for different network densities, which are extremely difficult to be evaluated in a real testbed.

5.1.5.1 Proof of Concept Experiment

Multi-interface Snort: A common way for running Snort and other similar passive network monitoring applications on multiple network interfaces is to *bridge* all interfaces into a single *virtual* network interface (a process also known as "bonding"), and run a single instance of the IDS on that virtual interface. On a mesh router, however, this solution is not possible because in Linux the bonded interfaces cannot be configured with routable IP addresses, and consequently the router cannot perform its main task of routing packets. Another option would be to run two Snort instances, one for each interface. Snort includes support for running multiple instances, but due to its single-threaded design, each instance is a different process, with separate copies of all buffers and data structures. Although this approach works well for typical multi-core IDS sensors with ample RAM, it is not practical for a mesh router with very limited CPU and memory resources [39].

To be able to run a single Snort instance that receives traffic from both network interfaces without altering the network configuration of each interface, we followed an alternative approach and modified Snort to capture packets concurrently through two Libpcap handles. This is possible by opening two Libpcap packet capture handles, one for each interface, and then asynchronously retrieving packets from either handle through `select()`, whenever packets are available. A Libpcap handle can be put into "non-blocking" mode using `pcap_setnonblock()`, and then a file descriptor that can be monitored using `select()` can be obtained through `pcap_get_selectable_fd()`.

This design allows us to: i) avoid the overhead of running a second Snort instance (context switches, duplicate data structures); ii) capture traffic from both local and upstream network interfaces concurrently; and iii) preserve the routing configuration of both interfaces. We experimentally observed that our multi-interface Snort imposes only $\sim 4\%$ extra memory load (compared to the original Snort) when running on a Netgear WNDR3700 router used in the PRIDE testbed.

Experimental Verification: We performed an experiment in a small-size indoor WMN to validate link-coverage monitoring and multi-interface monitoring. We note here that the idea of intrusion detection using a set of detection modules distributed on multiple WMN nodes was previously demonstrated and evaluated in PRIDE.

In our experiment, we used three Netgear WNDR 3700 routers (e.g., nodes A, B, and C) connected to each other creating a triangle WMN topology. Each router was configured to run a multi-interface Snort instance, monitoring network traffic on both 2.4 GHz (local traffic among its clients) and 5 GHz (WMN backbone traffic) wireless interfaces. Each of routers A and B had one client, while two clients (laptops) were connected to router C. Using the Rule to Attack (R2A) tool [38], we launched two different types of attacks: i) A's client targeting B's client (multi-hop attack); ii) a C's client targeting another C's client (single-hop attack). The corresponding detection modules for each attack were activated on multi-interface Snort running on node C. The alerts generated by the multi-interface Snort on router C proved the detection of both single-hop and multi-hop attacks simultaneously. Therefore, our proposed link-coverage (i.e., monitoring WMN backbone traffic on A-B link) and multi-interface Snort (i.e., monitoring both local and upstream interfaces concurrently) was shown to be practical for WMN.

5.1.5.2 Simulation Results

We performed a thorough set of simulations to evaluate the performance of centralized and distributed RAPID in covering WMN links and detecting different types of attack. We compare our simulation results with PRIDE and monitoring node solutions as two state-of-the-art solutions. We implemented a monitoring node solution (*Mon.Sol.*) based on the formulation presented in [37]. The objective function, however, was changed to select nodes with higher total memory so that more detection modules can be run on monitoring nodes, thus having a fair comparison with RAPID.

All algorithms are implemented in MATLAB and run for different network sizes and densities. More precisely, our evaluation metrics are Average LCR in WMN, Average Memory Load on WMN nodes, and Average Intrusion Detection Rates for different types of attack with respect to two tuning parameters, Memory Threshold λ and Network Density. The average base line memory of the nodes (vector B) was 20%. Our simulation results are based on 6-module and 12-module configurations.

5.1.5.3 Average LCR and Memory Consumption

To evaluate the average LCR and average memory load, we created 100 random networks of size 30 (Note: PRIDE performance is evaluated on a 10-node WMN.) The average LCR and its standard deviation obtained from centralized RAPID, distributed RAPID and monitoring node solution are depicted in Figure 5.3. Figure 5.3(a) shows the average LCR for 6-module configuration while Figure 5.3(b) depicts the average LCR for 12-module configuration.

As shown, the average LCR increases as λ increases which means more detection modules are executed on the nodes. The monitoring solution (consistent to the results shown for path coverage in) has the minimum coverage ratio since the selected monitoring nodes are resource-constrained and cannot perform all detection mod-

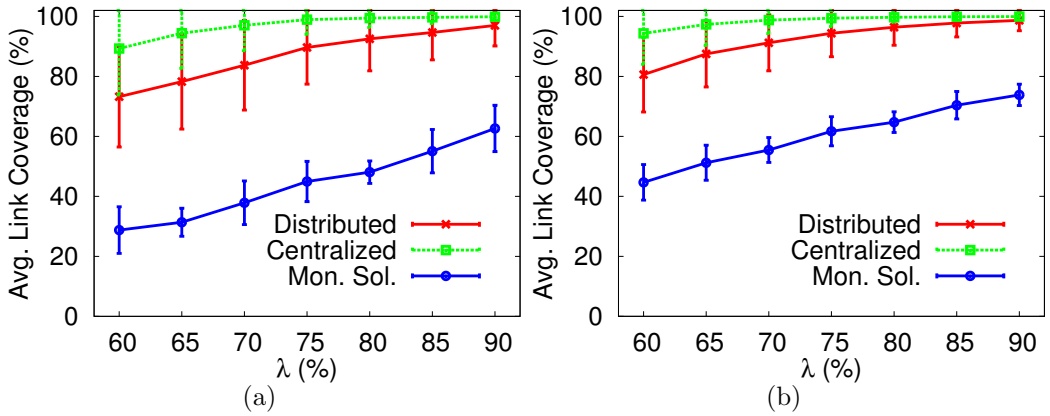


Figure 5.3: The effect of λ on the average link coverage in: (a) 6-Module configuration; (b) 12-Module configuration.

ules and do not help each other to achieve higher coverage ratios. Obviously, the average LCR in centralized RAPID is higher than that of distributed RAPID as the centralized approach uses global information and produces optimal IDS distribution. The distributed RAPID, however, achieves an almost similar LCR to the centralized RAPID for large λ.

These results are comparable to the path coverage ratio obtained for 2-hop paths in PRIDE. We note here that the execution time for the centralized RAPID is at most $\sim$ 5 seconds (for 30-node WMN) while it was more than 1 minute for the 10-node WMN in PRIDE (using ILP solver) and more than 1 hour for 30-node WMN. Moreover, when considering the average LCR, both centralized and distributed RAPID outperform PRIDE because RAPID uses the link-coverage approach, which allows more nodes to participate in traffic monitoring. As expected, the average LCR is slightly higher in 12-module configuration especially for small λ. This is because the size of detection modules are smaller than those in 6-module configuration, which allows more modules to fit in the small free memory spaces. It is worth mentioning that such a better performance obtained from 12-module configuration is at the price

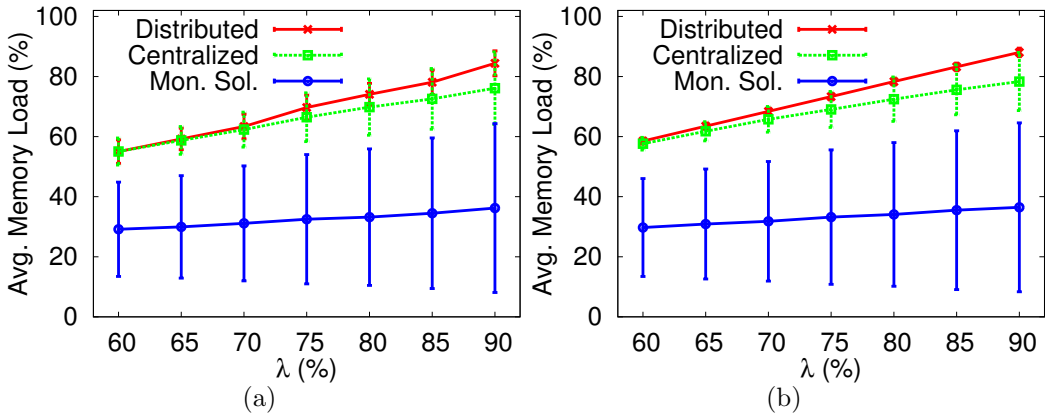


Figure 5.4: The effect of λ on the average memory load in: (a) 6-Module configuration; (b) 12-Module configuration.

of *slightly longer execution time* in RAPID, PRIDE, and Mon.Sol.

The average memory load on WMN nodes and its standard deviation of all three IDS solutions for the 6-module and 12-module configurations are depicted in Figures 5.4(a) and 5.4(b), respectively. It is important to note that the average memory load for the RAPID solution (both centralized and distributed) is always higher than that of monitoring node solution, i.e., consistent with results shown in PRIDE. This is because in monitoring node solution, only monitoring nodes are assigned with detection modules and the non-monitoring nodes are not loaded with any detection modules. Therefore, only few selected nodes will have high memory load as opposed to RAPID where all WMN nodes are loaded with the maximum number of detection modules that can fit. The large standard deviation of average memory load in Mon.Sol. indicates the difference between total memory load on monitoring nodes and non-monitoring nodes.

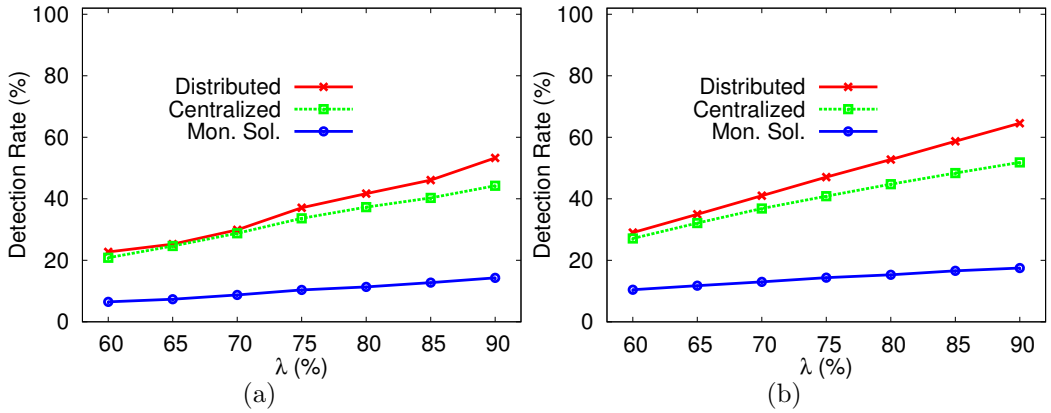


Figure 5.5: The effect of λ on the detection rate of single-hop (local) attacks in: (a) 6-Module configuration; (b) 12-Module configuration.

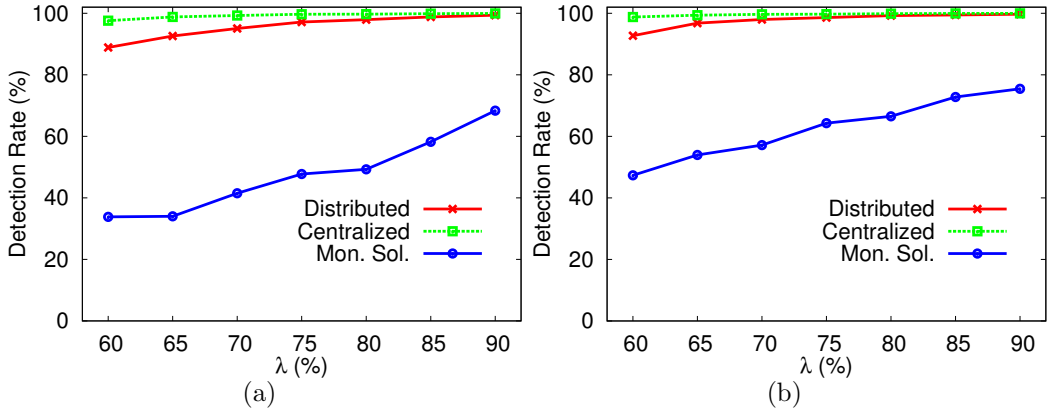


Figure 5.6: The effect of λ on the detection rate of multi-hop attacks in: (a) 6-Module configuration; (b) 12-Module configuration.

5.1.5.4 Average Detection Rates for Different Attacks

As mentioned in our attacker model, we consider both single-hop (local) and multi-hop attacks in WMN. For a given WMN of size n, we simulated $10 \times n$ single-hop attacks and $2 \times n$ multi-hop attacks of random types (i.e., detectable by random detection modules as listed in [38]) and measured the detection rates based on the activated detection modules on the nodes.

Figure 5.5 shows the average detection rate of all $10 \times n$ single-hop attacks obtained from different IDS solutions. The results are produced for 100 random WMN of size n=30. Figure 5.5(a) depicts the average detection rates of single-hop attacks in all IDS solutions when the 6-module configuration is used. As shown, the larger the λ, the higher the detection rate is. This is because a larger memory threshold allows nodes to load and execute more detection modules and detect more local attacks, since the neighbors cannot help the node in detecting local attacks. As depicted in Figure 5.5(b), the average detection rate for 12-module configuration is slightly higher than those of 6-module configuration in all three IDS solutions. It is worth mentioning that, although the detection rates for both centralized and distributed RAPID are at most $\sim 60\%$, they are much better than for the monitoring node solution (i.e., at most $\sim 20\%$) and for PRIDE (i.e., 0% for local attacks).

To evaluate the performance of IDS solutions in detecting multi-hop attacks, we considered 100 random networks of 30 nodes and 60 random paths. The path length of each attack is randomly chosen between 2 and 5 hops. Figures 5.6(a) and 5.6(b) depict the average detection rates of multi-hop attacks in all three solutions for 6-module configuration and 12-module configuration, respectively. As shown, the detection rates for multi-hop attack in RAPID and Mon.Sol. are much higher than those for single-hop attacks. This is because as traffic packets go through more IDS nodes, they will be more likely inspected by more distinct detection modules. Moreover, as previously observed, the larger the λ, the higher the detection rate will be. Also, the 12-module configuration again outperforms the 6-module configuration (at the price of slightly larger time complexity).

Figures 5.7(a) and 5.7(b) show the simulation results for average detection rates of compromised node attacks in 6-module and 12-module configurations, respectively, in all IDS solutions. The results are obtained from 100 random networks of 30 nodes

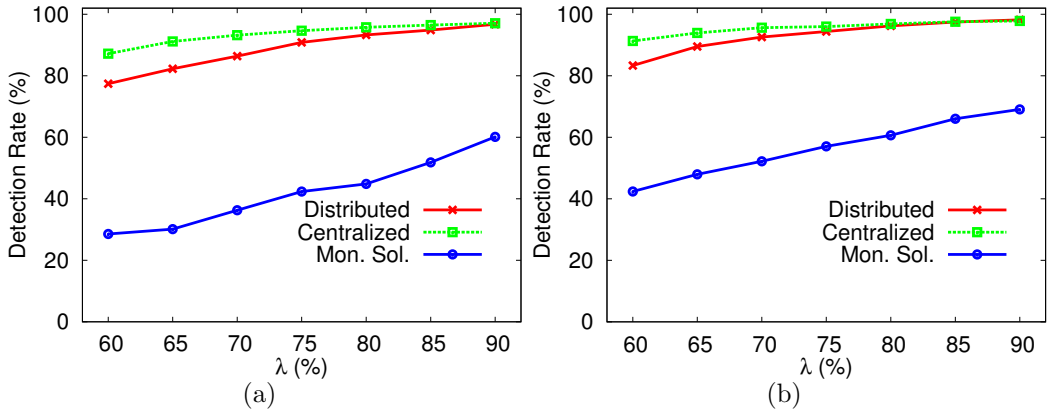


Figure 5.7: The effect of λ on the detection rate of compromised node attacks in: (a) 6-Module configuration; (b) 12-Module configuration.

where 60 random attacks are considered for each network. The compromised node is randomly chosen among WMN nodes to run either a single-hop (targeting a neighbor WMN node) or multi-hop attack. As shown, the results are slightly worse than multi-hop attacks because the compromised node itself is considered unable to detect the attack, which results in inspecting attack traffic with less detection modules.

The last type of attack we consider for intrusion detection evaluation is unauthorized client attack. An unauthorized client is assumed to be physically located in WMN area but not associated with any of MAPs (i.e., outsider). The attacker can launch attacks against WMN nodes (e.g., DoS, battery depletion, spoofed de-authentication, etc.) or WMN links (e.g., jamming, blackhole/grayhole, etc.). We assume that in the attack against a WMN node, the target is unable to participate in the intrusion detection process. Figures 5.8(a) and 5.8(b) show the average detection rate of unauthorized client attacks targeting WMN *nodes* for 6-module and 12-module configurations, respectively. The results are obtained from 100 random networks of 30 nodes where 300 random attacker locations and targets are considered. The results show that these attacks are highly detectable by RAPID algorithms as

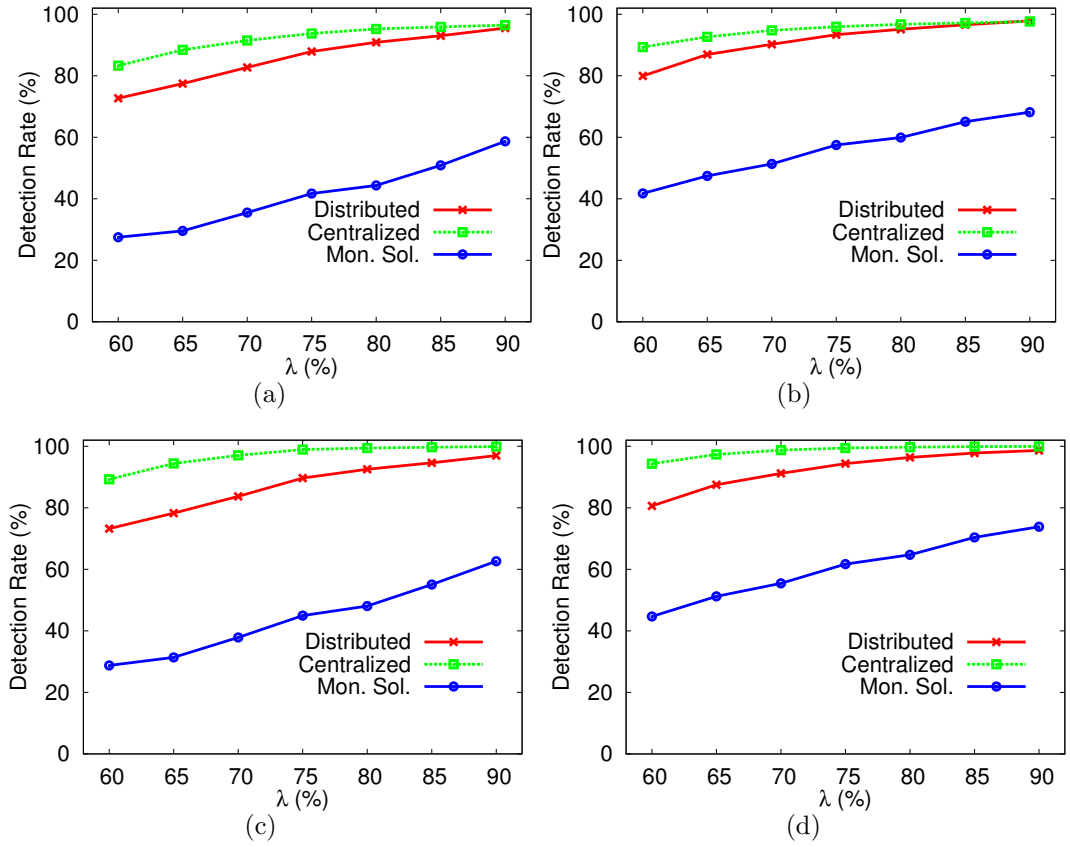


Figure 5.8: The effect of λ on the detection rate of unauthorized client (outsider) attacks: (a) against nodes in 6-Module configuration; (b) against nodes in 12-Module configuration; (c) against links in 6-Module configuration; (d) against links in 12-Module configuration.

opposed to Mon.Sol. solution that can achieve at most $\sim 60\%$ detection rate. We note here that PRIDE cannot detect such attacks since the attack traffic is not routed through WMN nodes. Figures 5.8(c) and 5.8(d) show the average detection rate of unauthorized client attacks targeting WMN *links*, when using 6-module and 12-module configurations, respectively. As depicted, the results are slightly better than those targeting WMN nodes because more nodes participate in monitoring the target WMN link.

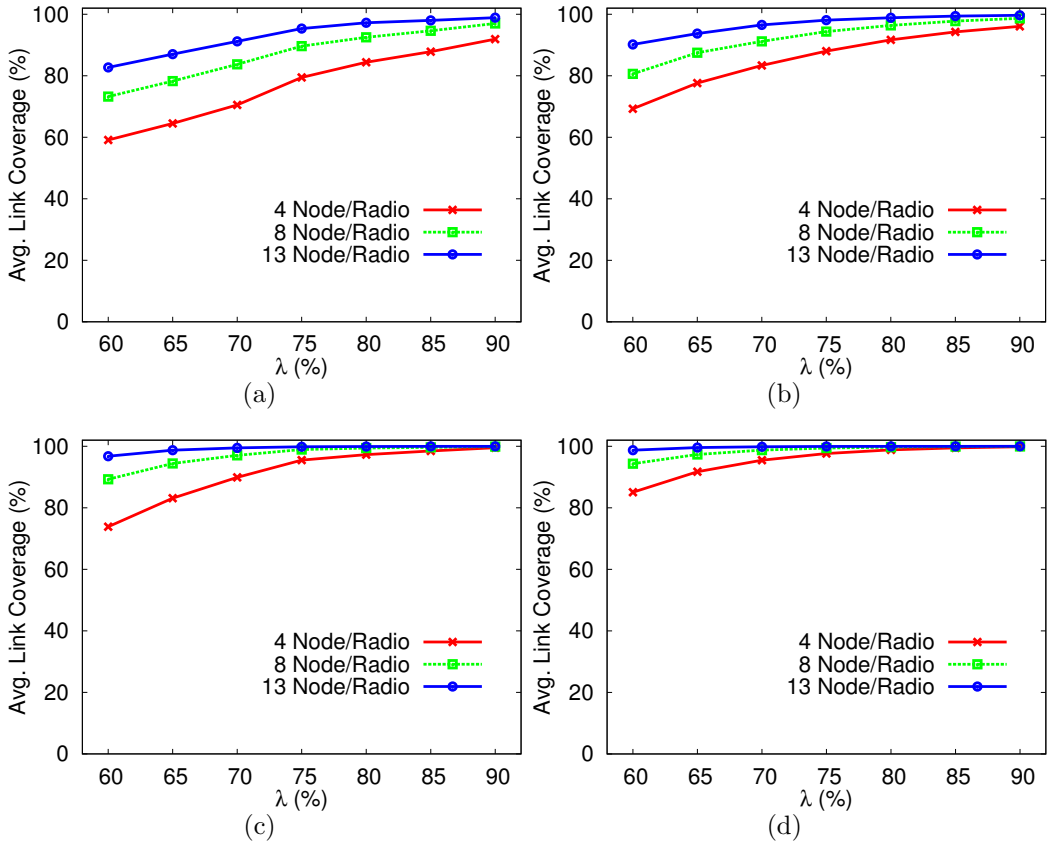


Figure 5.9: The effect of λ and network density on the average link coverage in: (a) 6-Module configuration of distributed RAPID; (b) 12-Module configuration of distributed RAPID; (c) 6-Module configuration of centralized RAPID; (d) 12-Module configuration of centralized RAPID.

5.1.5.5 The Effect of Network Density on RAPID Performance

In order to show the effect of network density on performance of RAPID, we repeated all previous simulations (i.e., network density was 8 nodes per radio range) for two more network densities, 4 and 13 nodes per radio range. Intuitively, the higher the network density should result in participating more neighbors in traffic monitoring that would increase the average link coverage ratio and consequently the intrusion detection rate. In this section, we show the simulation results for average

LCR and average detection rates of different attacks as functions of λ and network density.

Figures 5.9(a) and 5.9(b) show the average LCR obtained from distributed RAPID for 6-module and 12-modules configurations, respectively. The results confirm that the average LCR increases asλ or network density increase. Figures 5.9(c) and 5.9(d) depict the average LCR obtained from centralized RAPID for 6-module and 12-modules configurations, respectively. The results obtained from centralized RAPID are better than those obtained from distributed RAPID, at the price of some communication and computation overheads. We note here that the network density has no effect on the average LCR of PRIDE (because of using node-coverage instead of link-coverage approach) and Mon.Sol (because it only affects the number of monitoring nodes and not the number of detection modules they perform).

Figures 5.10(a) and 5.10(b) show the effect of λ and network density on the detection rate of multi-hop attacks in the distributed RAPID for 6-module and 12-modules configurations, respectively. Surprisingly, the multi-hop attacks are almost always detectable for $\lambda \geq 70\%$ and network density larger than 8 nodes per radio range in both 6-and-12-module configurations. Figures 5.10(c) and 5.10(d) show the results for the centralized RAPID which are above 90% even for the lowest network density and memory threshold. We note here that network density has no effect on single-hop attack detection as only one node (the local router) is responsible for intrusion detection and other WMN nodes do not participate in the intrusion detection process.

Figures 5.11(a) and 5.11(b) show the effect of λ and network density on the detection rate of compromised node attacks in the distributed RAPID for 6-module and 12-modules configurations, respectively. As depicted, the detection rate increases as network density and λ increase which means more nodes with more detec-

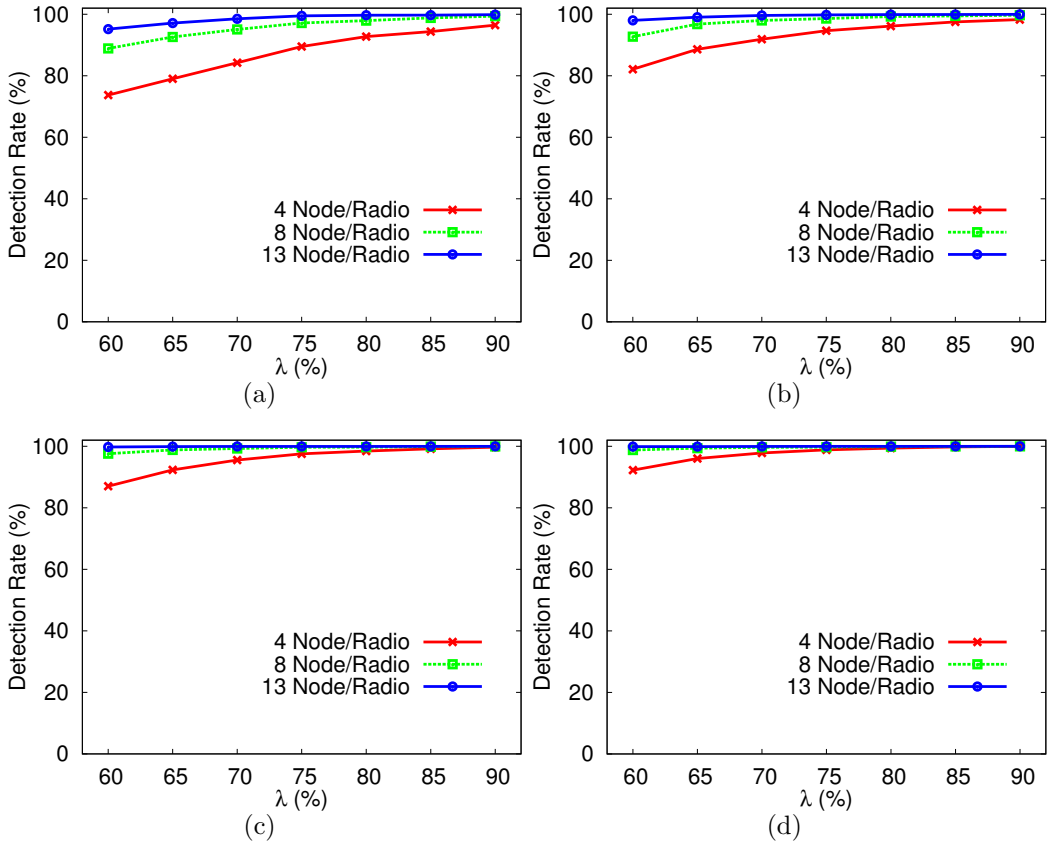


Figure 5.10: The effect of λ and network density on the detection rate of multi-hop attacks in: (a) 6-Module configuration of distributed RAPID; (b) 12-Module configuration of distributed RAPID; (c) 6-Module configuration of centralized RAPID; (d) 12-Module configuration of centralized RAPID.

tion modules inspect the attack traffic generated by the compromised nodes. Figures 5.11(c) and 5.11(d) show the same results for the centralized RAPID when using 6-module and 12-modules configurations, respectively. The results show that centralized RAPID outperforms distributed RAPID, however, at the price of higher computation and communication overheads.

The effect of λ and network density on the detection rate of unauthorized client (outsider) attacks against WMN nodes in the distributed RAPID are shown in Figures 5.12(a) and 5.12(b) for 6-module and 12-modules configurations, respectively.

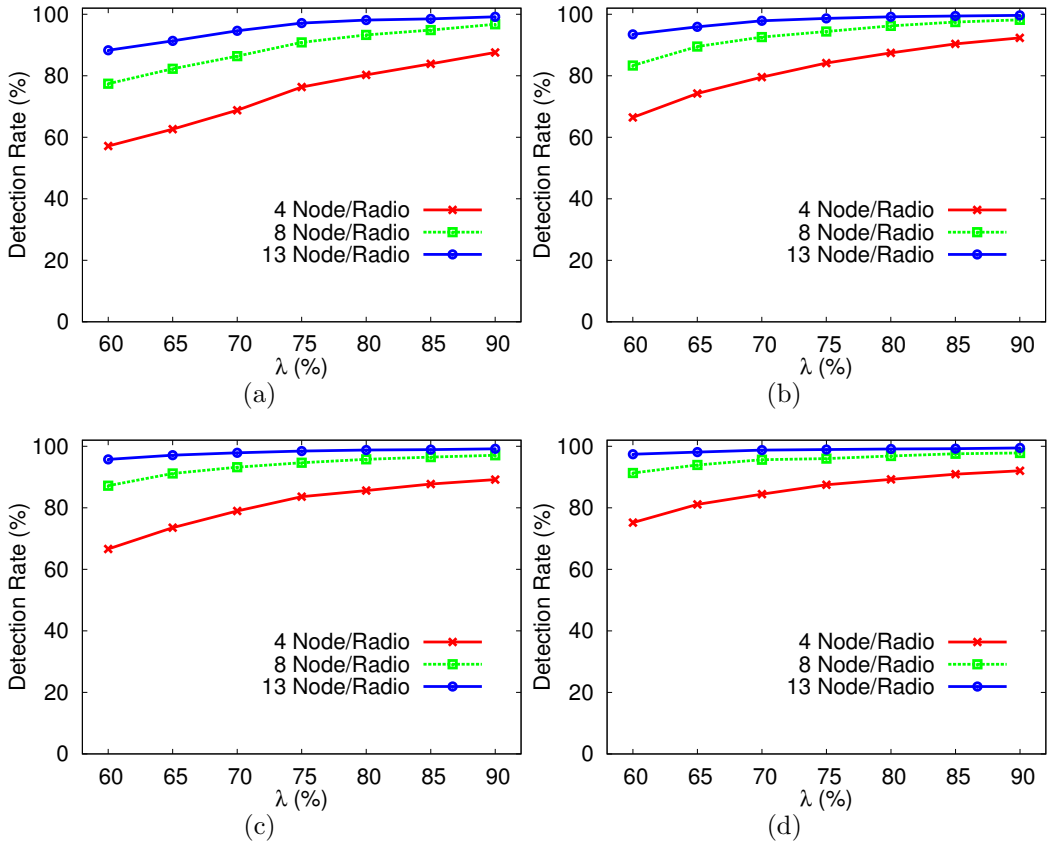


Figure 5.11: The effect of λ and network density on the detection rate of compromised node attacks in: (a) 6-Module configuration of distributed RAPID; (b) 12-Module configuration of distributed RAPID; (c) 6-Module configuration of centralized RAPID; (d) 12-Module configuration of centralized RAPID.

Also, Figures 5.12(c) and 5.12(d) show the same results for the centralized RAPID when using 6-module and 12-modules configurations, respectively. The results confirm that the larger the λ and network density, the higher the detection rate will be. Moreover, centralized approach works better than distributed approach as 12-module configuration also works better than 6 module configuration.

Finally, we show the effect of λ and network density on the detection rate of unauthorized client (outsider) attacks against WMN links in both distributed and centralized RAPID. Figures 5.13(a) and 5.13(b) show the detection rates in the distributed

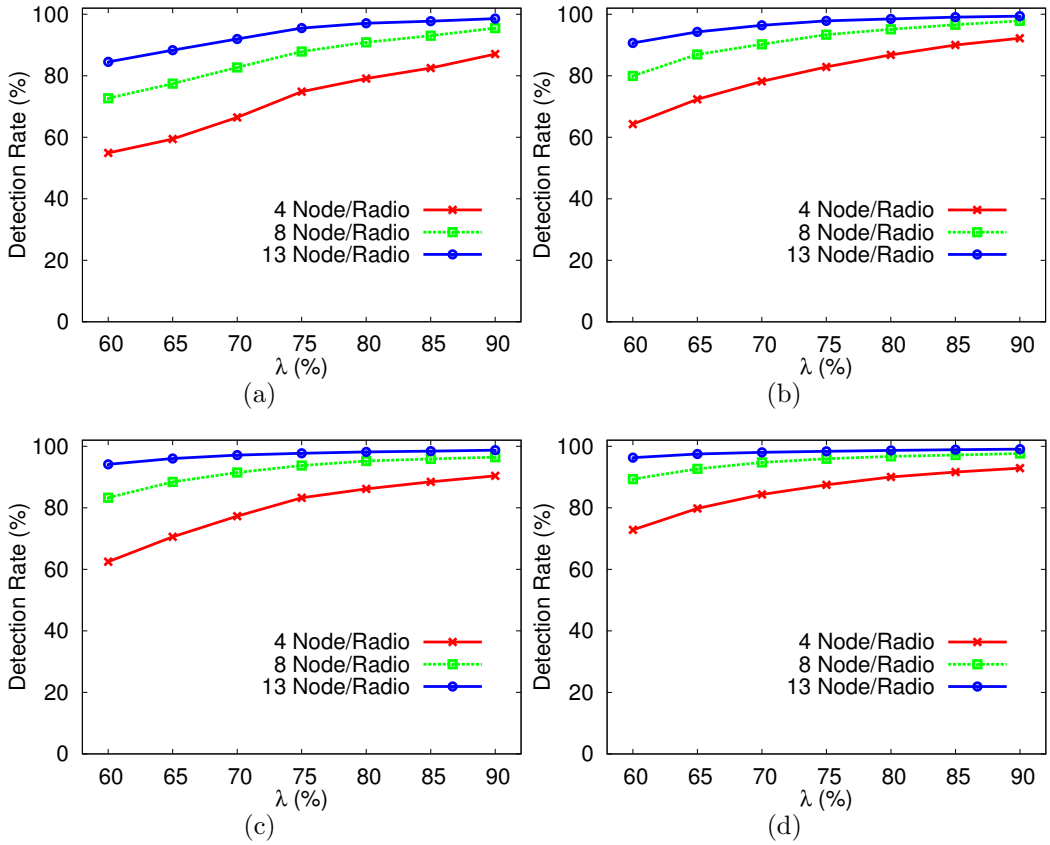


Figure 5.12: The effect of λ and network density on the detection rate of unauthorized client (outsider) attacks against nodes in: (a) 6-Module configuration of distributed approach; (b) 12-Module configuration of distributed approach; (c) 6-Module configuration of centralized approach; (d) 12-Module configuration of centralized approach.

RAPID for 6-module and 12-module configurations, respectively. The results are slightly better than those obtained from attacks against WMN nodes since more nodes participate in traffic monitoring. Figures 5.13(c) and 5.13(d) show the same results for centralized RAPID when using 6-module and 12-module configurations, respectively.

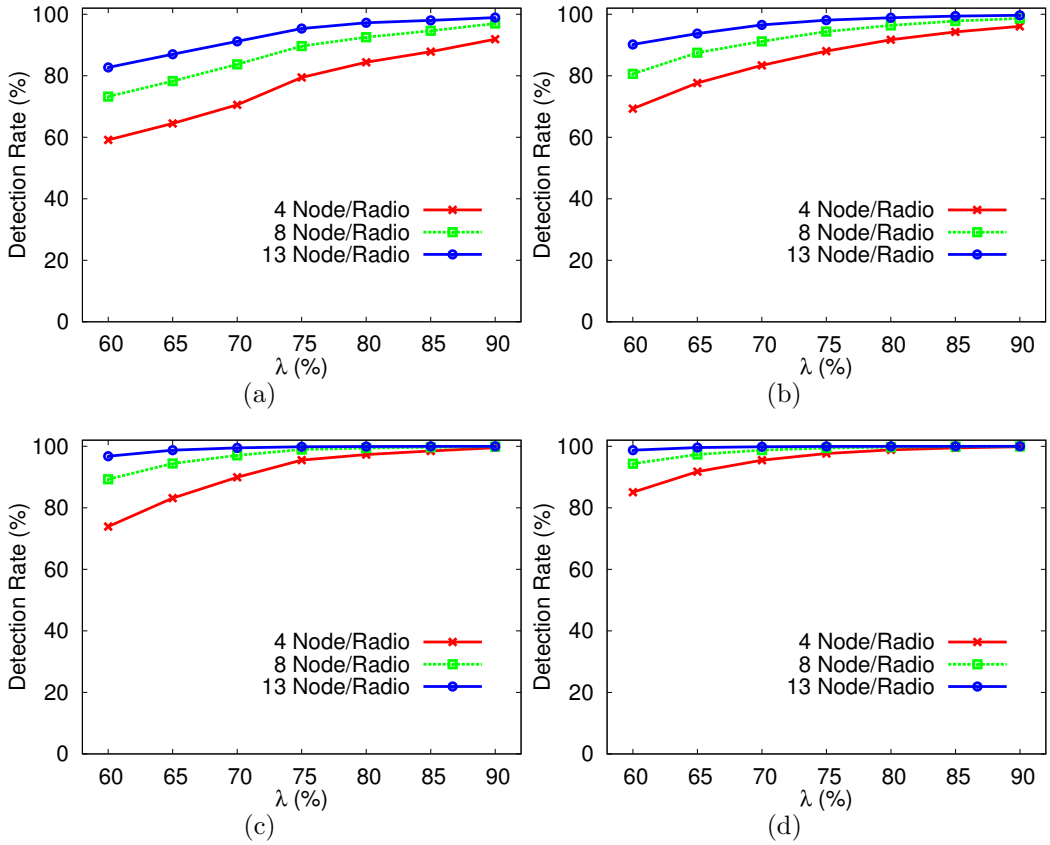


Figure 5.13: The effect of λ and network density on the detection rate of unauthorized client (outsider) attacks against links in: (a) 6-Module configuration of distributed approach; (b) 12-Module configuration of distributed approach; (c) 6-Module configuration of centralized approach; (d) 12-Module configuration of centralized approach.

5.2 Cooperative IDS

Another approach to conserve resources and manage the intrusion detection mechanism for resource-constrained WMN is to use cooperative IDS solutions. In cooperative IDS, the network topology used for communicating intrusion detection reports has an important effect on network performance and resource consumption. Finding the optimum network topology for nodes that execute cooperative IDS has been shown to be an NP-hard problem [75, 83]. Previous research investigated distributed

solutions in which nodes, using information about their neighbors, elect a local candidate for executing cooperative IDS functions [46, 79]. These solutions, however, are suboptimal and incur high communication overhead. In this section, we propose a solution in which a base station which has knowledge about network (e.g., node resources, locations, etc.) and security requirements (e.g., maximum permissible delay in reporting an event, minimum network coverage), computes the optimal distribution of roles specific to cooperative IDS. Our proposed solution allows execution of sophisticated algorithms that optimize multiple objectives related to network performance and security effectiveness.

5.2.1 Cooperative IDS Architecture and Problem Formulation

Our target system is a battery-powered WMN [28] consisting of resource constrained wireless networks (i.e., battery powered wireless mesh and sensor networks) with a single base station (e.g., a Command & Control Center). The network is loosely time synchronized and all nodes know their locations. The base station, with extensive computational capabilities, periodically collects network information and uses this information to decide which IDS functions to run, and where to run them. This decision is done periodically, or when network conditions change, e.g., when nodes' residual energy is below a threshold.

The decision of how to distribute IDS functions, results in a network organized as a set of *cluster trees*, each running a distributed cooperative IDS. Figure 5.14(a) depicts an example of a network cooperative IDS, which consists of a cluster tree with 5 nodes and one additional node, not associated with the cluster tree. The dotted lines in the figure 5.14(a) represent network connections while the arrow lines show the cooperation direction. Figure 5.14(b) shows a generic node architecture in a cooperative IDS. The *Data Collection* module, executed by all nodes, collects

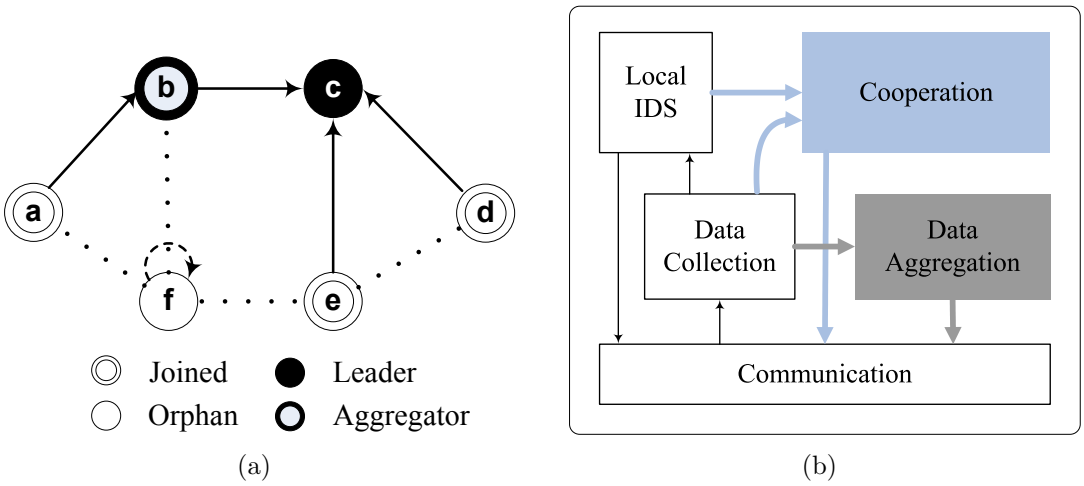


Figure 5.14: (a) Example of a network with a cooperative IDS (nodes responsibilities vary). (b) A generic architecture for nodes in a cooperative IDS.

local system and network data, for intrusion detection purposes. Nodes also have a *Local IDS* module that evaluates a set of intrusion detection rules. The remaining two modules, "Cooperation" and "Data Aggregation" are explained below within the context of Aggregator and Leader roles for a node.

Inspired by previous intrusion detection research [49, 54, 81, 84] we identify four distinct roles for the nodes in our distributed IDS:

- *Joined* nodes are leaves in a cluster tree. They monitor local activity (e.g., communication, processes running, data produced) and run a local IDS, as depicted in Figure 5.14(b). Results are reported to the parent, which can be an Aggregator or a Leader.

- *Aggregator* nodes also monitor local activity, as a joined node does, receive reports from children, either joined or other aggregator nodes, and aggregate this data with their information, using the *Data Aggregation* module in Figure 5.14(b). The aggregated data is reported to a parent, i.e., either an aggre-

gator or a leader.

- *Leader* is the root of a cluster tree. A leader receives reports from its children, either joined or aggregator nodes, and executes sophisticated IDS functions, e.g., alert correlation, as part of a *Cooperation* module (Figure 5.14(b)). The results are reported to the base station. *Leaders of all cluster trees form a connected graph, which contains the base station.* As shown in Figure 5.14(a) there is a single-cluster tree, with node c as its leader.

- *Orphan* nodes are not part of a cluster tree. They run local IDS and do not forward their observations to their neighbors.

The nodes with the aforementioned roles can largely be described as: 1) *Tasked* nodes if they are either Joined, Aggregator, or Leader; 2) *Untasked* nodes - the Orphan nodes. Similarly we will refer to all children of a node as its *Followers*.

Individual nodes, organized in cluster trees, communicate and aggregate data for the purpose of detecting intruders, in a collaborative manner. The proposed cluster tree organization impacts energy consumption (hence, network lifetime), event reporting delay, network coverage, and quality of data collected. Each of these properties represents an individual optimization objective in a multi-objective optimization problem, formally described in Section 5.2.1.1.

The proposed system architecture, based on cooperative IDS, is the main focus of our research. For the intrusion detection system we employ existing IDS engines, e.g., Snort [78], which is based on rulesets. For our approach, it is critical to observe that more complex actions performed by the detection engine (e.g., number of rules evaluated, processing stages involved) will pose a higher demand on available resources (e.g., computation, communication). Consequently, the configuration of the IDS engine presents opportunities to tradeoff intrusion detection accuracy for

resource availability. Thus, we consider three types of intrusion detection engines: lightweight (LW-DS), employed by joined and orphan nodes; medium (RE-DS), employed by aggregators; and heavy (HW-DS), employed by leaders. The types of attacks we consider in this research are the following:

- *Flooding:* the purpose of this attack is to exhaust both network and host resources by sending a rapid succession of many request-type packets. As examples for this attack, we consider the SYN and ICMP flood attacks. For detecting these attacks an LW-DS is sufficient.

- *Port scanning:* this is a generic attack that probes a target node for open ports. As an example, we consider TCP port scanning employing TCP SYN and FIN packets. For detecting a port scanning attack, an RE-DS or HW-DS is required.

- *Web exploits:* for this, the attacker hosts an HTTP server and executes HTML exploits (e.g., information disclosure), against clients. Due to the complexity of this attack, only a HW-DS is capable of detecting it.

5.2.1.1 Problem Formulation

Given a set $N = \{n_1, n_2, ..., n_k, b\}$ of $k + 1$ nodes, which includes base station b, and the roles of nodes identified at the beginning of the section, let L be the set of leader nodes, A the set of aggregator nodes, J the set of joined nodes and O the set of orphan nodes in the network. We denote by b_{n_i}, the residual battery charge of node n_i. Each node n_i in the network is assigned a single responsibility. Using the proposed cooperative IDS architecture, nodes are organized in cluster trees. Let $G = \{T_1, ..., T_q\}$ be the set of q cluster trees formed in the network. Each cluster tree T_i has only one leader ($|L_{T_i}| = 1$), and one or more aggregators A_{T_i}

and joint nodes J_{T_i}: $T_i = L_{T_i} \cup A_{T_i} \cup J_{T_i}$. As mentioned, the set of leader nodes forms a *connected* graph. All nodes communicate with radio range R. Leaders can communicate over greater distances using radio range $R' = \beta R$. More formally, we define $G_L(L \cup \{b\}, L \times L)$ as the graph formed by leaders $L_i \in L$ such that L_i is connected to L_j iff $\text{dist}(L_i, L_j) \leq R', \forall i \neq j$.

Different cluster tree topologies exhibit different characteristics for energy consumption, event reporting delay, network coverage, and data accuracy. Some of these properties need to be minimized (e.g., energy consumption and delay), while others need to be maximized (e.g., network coverage and data accuracy). Individually, each of these properties can be treated naively as a single objective optimization problem. However, a more complex, but more commonly sought goal in a cooperative IDS for a set of cluster trees, is to find the lowest aggregate energy consumption and delay with the highest aggregate data accuracy and network coverage. It is obvious that single objective optimizations may negatively impact the performance of other objectives in a multi-objective environment. Therefore, for any set of cluster trees, we define the network performance $F(G) = f(P_G, H_G, D_G, C_G)$, as a multi-variate objective function, where P_G, H_G, D_G and C_G are functions for Power, Information, Delay and Coverage, respectively, in network G. This multi-variate objective function trades off Power and Delay, two minimization objectives, in order to improve Information and Coverage, two maximization objectives. In addition, since the objective values of these functions are in different scales, they are normalized using C_p, C_d, C_h and C_c:

$$F(G) = f\left(\left(1 - \frac{P_G}{C_p}\right), \frac{H_G}{C_h}, \left(1 - \frac{D_G}{C_d}\right), \frac{C_G}{C_c}\right)$$

Consequently, our Multi-objective Optimization (MOO) problem is:

$$\text{maximize} \quad \left[\left(1-\frac{P_G}{C_p}\right), \frac{H_G}{C_h}, \left(1-\frac{D_G}{C_d}\right), \frac{C_G}{C_c}\right] \tag{5.4}$$

$$\text{subject to:} \quad |L_{T_i}| = 1, \forall i = 1, ..., q \tag{5.5}$$

$$\text{if } n_j \in A_{T_i}, J_{T_i} \text{ then}$$

$$\text{parent } (n_j) \in A_{T_i} \cup L_{T_i}, \forall i = 1, ..., q \tag{5.6}$$

$$|J_{T_i}| \geq 1, \forall i = 1, ..., q \tag{5.7}$$

$$G_L \text{ is a connected graph} \tag{5.8}$$

$$b_{L_i} \geq b_L^{th}, \forall L_i \tag{5.9}$$

$$|T_i| \leq |T|_{th}, \forall i = 1, ..., q \tag{5.10}$$

where (5.4) is a vector of all objective functions; constraint (5.5) indicates that a single leader node is in each cluster tree; constraint (5.6) enforces the construction of the cluster tree (to conform with our cooperative IDS architecture); constraint (5.7) indicates that at least one joined node must be in each cluster tree; constraint (5.8) enforces the leaders to be connected to the graph G_L formed by leaders including base station; and constraint (5.9) says that the residual battery charge of the leader has to be greater than a defined threshold value b_L^{th}. Finally, constraint (5.10) enforces the size of cluster trees to be smaller than a defined threshold $|T|_{th}$, to prohibit the creation of large cluster trees. This constraint seeks solutions that are sets of trees instead of one large tree (a typical single point of failure).

For investigating solutions for MOO, we need normalizing constants C_p, C_d, C_h and C_c. These upper bound values are obtained by solving the corresponding Single Objective Optimization (SOO) problems. As an example, for maximizing Information in the network and finding C_h, the corresponding SOO is:

$$\text{maximize} \qquad \sum_{i=1}^{q} \sum_{j=1}^{|T_i|} h_{ij} \tag{5.11}$$

$$\text{subject to:} \qquad \text{same constraints as MOO} \tag{5.12}$$

where h_{ij} represents the information available to node j in cluster tree i. Similarly we can formulate single optimization problems that minimize Delay and Power, and that maximize Coverage.

5.2.2 Models for Single Objectives

In this section we develop mathematical models for P_G, H_G, D_G, C_G, the Power, Information, Delay, and Coverage functions, respectively, i.e., individual objectives that compose the multi-objective optimization problem.

5.2.2.1 Power Model

Nodes in the network, depending on their roles, consume different amounts of energy for communication and computation. Leader and aggregator nodes perform functions that consume more power (i.e., cooperation module in leaders and aggregation module in aggregators). Similarly, cluster trees may vary in the number of tasked and untasked nodes and in the number and distribution of followers for each leader and aggregator. This means that the total power consumption of any set of cluster trees will vary from other possible sets. Thus, one single objective optimization problem is to minimize total power consumed by all nodes in a set of cluster trees G formed in the network, as given by:

$$P_G = \sum_{i=1}^{k} p_i = \sum_{i=1}^{|L|} p_{L_i} + \sum_{i=1}^{|A|} p_{A_i} + \sum_{i=1}^{|J|} p_{J_i} + \sum_{i=1}^{|O|} p_{O_i} \tag{5.13}$$

where p_i is the power consumption of node i for both communication and computation activities, which will depend on the role assigned to it (i.e., leader, aggregator, joint or orphan). In the remaining part of the section, we present the power consumption models for leader nodes p_{L_i}, aggregator nodes p_{A_i}, joined nodes p_{J_i}, and orphan nodes p_{O_i}.

For our power consumption model, we extend the one proposed in [26]. We consider the power consumed for radio communication or processing (computation). We denote by P_{TX} and P_{RX} the power consumed for transmitting and for receiving a report, respectively. We denote by P_{Rep} and P_{Recv} the power consumed for transmitting and for receiving an intrusion detection alarm, respectively. The power consumed for processing/computation, denoted by P_{Proc}, depends on the role assigned to a node. The following notations are with reference to the proposed system architecture, shown in Figure 5.14(b) - the "Cooperation" and "Data Aggregation" modules. For an aggregator node, which has the "Data Aggregation" module active, $P_{Proc} = P_{ModA} + P_{CodA}$ where P_{ModA} is a fixed power consumed for maintaining the module active and P_{CodA} is a variable power, consumed for coding/aggregating data from follower nodes. For a leader node, which has the "Cooperation" module active, $P_{Proc} = P_{ModC} + P_{CodC}$ where P_{ModC} is a fixed power consumed for maintaining the module active and P_{CodC} is a variable power, consumed for coding/cooperating data from follower nodes.

We denote by E_i an event vector, which is a set of observed security parameters reported by node i. We denote by l_{E_i} its length ($l_{E_i} = |E_i|$). We remark here that l_{E_i} is the same for all joined nodes (since they do not have followers) and that it increases with the number of descendants. Considering our system architecture for cooperative IDS, each tasked node i, except for leaders, sends E_i in each time slot, at a rate λ packets per second. We assume that λ is equal for all nodes. We use F_i

as the set of followers of node i. Considering the tasks mentioned in Section 5.2.1, the power consumed by a leader node becomes:

$$\begin{aligned} p_{L_i} &= P_{RX_i} + P_{Proc_i} + P_{Rep_i} + P_{Recv_i} \\ &= p_{rx} \sum_{j=1}^{|F_i|} \lambda l_{E_{ij}} + P_{CodC} \left[\lambda l_{E_i} + \sum_{j=1}^{|F_i|} \lambda l_{E_{ij}} \right] \\ &+ P_{ModC} + \eta l_A (p_{tx} + p'_{tx} + p'_{rx}) \end{aligned}$$

where p_{tx} and p_{rx} are the power consumptions required to transmit and receive one bit, respectively, and p'_{tx} and p'_{rx} are the power consumptions required for transmitting one bit over a long distance between leaders. Similarly, $p_{A_i} = p_{L_i} + P_{TX_i} = p_{L_i} + p_{tx}\lambda l_{E_i}$.

Reporting activity consumes $P_{Rep} = \eta l_A p'_{tx}$, depending on the attack frequency η in the network ($0 \leq \eta \leq 1$) where l_A is the size of alarm. Leaders need to communicate with each other, using long distance communication, to exchange alerts in case of attack detection as well as sending alert to the followers, using regular radio range communication. Since aggregators forward alerts to their children, they consume P_{Rep} and P_{Recv} for transmitting/receiving alarms, respectively.

In contrast to leaders and aggregators, the only tasks of joined nodes are to transmit their event vector to their parents and to receive alarms from them. Therefore, the power consumption of a joined node is $p_{J_i} = P_{TX_i} + P_{Rep_i} = p_{tx}\lambda l_{E_i} + \eta l_A p_{rx}$. Since orphan nodes do not execute any cooperative IDS processes, their power consumption is 0 ($p_{O_i} = 0$).

5.2.2.2 Information Model

For IDS cooperation, in which each node has a local IDS and sends reports to its parent, the way nodes are organized in cluster trees and cooperate affects the amount of data collected. An optimal solution for our MOO problem collects as much data as possible, but, at the same time reduces the amount of useless data collected. Thus, for defining one additional objective in our MOO, we develop a model for data collected in a cooperative IDS.

We already mentioned that each node i sends an event vector E_i to its parent. Here, for simplicity, we assume that each vector contains only one security parameter which is a random variable x_i with a Gaussian distribution ($x_i \sim N(\mu, {\sigma_i}^2)$). The amount of information contained in x is its entropy, $H(x)$. We denote by $H(x_i)$ the data entropy of a reporting parameter of node i, and by $H(x_i, x_j)$ the joint entropy of two variables x_i and x_j, where node i is the parent of node j. So, the data entropy at an aggregator is the joint entropy of all received data from its followers combined with its own observations.

For a set of cluster trees, the Information function is defined as summation of all data entropy at the nodes as $H_G = \sum_{i=1}^{k} h_i$, where $h_i = H(x_i)$ for any joined node and $h_i = H(x_i, ..., x_n)$, as the entropy of all parameters received by aggregator i combined with its own security parameter. This model guarantees that upper level aggregators get access to all information reported by their descendants but it does not require an aggregator node to forward all received packets. Hence, this reporting process ensures that all cooperators have access to all information in the local cooperating cluster tree. This data aggregation reduces the total amount of transmitted data between aggregators. Each observed parameter contains $H(x) = \frac{1}{2}\log(2\pi e\sigma^2)$ bits of information where σ^2 is the variance of Gaussian random variable x. The joint

entropy of n Gaussian random variables is $H(x_1, ..., x_n) = \frac{1}{2}\log[(2\pi e)^n det(\Sigma)]$, where Σ is the covariance matrix of the joint variables [20]. Matrix Σ contains all σ_i's as standard deviation of Gaussian random variables, σ_i^2's as their variances, and ρ_{ij} as the correlation coefficient of any pair of variables i and j as follows:

$$\Sigma_{x_1,x_2,...,x_n} = \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n1}\sigma_n\sigma_1 & \rho_{n2}\sigma_n\sigma_n & \cdots & \sigma_n^2 \end{pmatrix}$$

In Σ, the correlation coefficient ρ_{ij} is a function of nodes i and j locations. Formally, $\rho_{ij} = f(X_i, Y_i, X_j, Y_j)$, where X_i and Y_i are the x and y coordinates of node i. This means that ρ depends on the location, length and orientation of the line segment between two nodes. We assume that the location of the attacker has a random distribution in the network area, so that variable x is a stationary random variable and independent of node location. Thus, $\rho = f(d_{ij})$, which means that ρ is only a function of the distance between two communicating nodes regardless of their location in the network. In this research, we assume $\sigma = 1$ considering security parameter x_i as standard Gaussian random variable. Assuming the same circular radio range for all nodes $A = \pi R^2$, then $\rho_{ij} = (A_i \bigcap A_j)/A$, if $d_{ij} \leq 2R$, and 0, otherwise, where $A_i = A_j = A$. Overlapping area $(A_i \cap A_j)$ is calculated as $A_i \cap A_j = 2R^2 \cos^{-1}\left(\frac{d_{ij}}{2R}\right) - \frac{d}{2}\sqrt{4R^2 - d_{ij}^2}$.

5.2.2.3 Delay Model

In the proposed IDS cooperation model, we assume that an aggregator node processes all received reports during timeslot τ and sends the aggregated report to its parent during timeslot $\tau + 1$. The delay between a report transmitted by a node,

until it is received by a leader located at a distance μ_i hops is equal to the number of hops. Thus, the total event reporting delay in the network becomes $D_G = \sum_{i=1}^{k} \mu_i$.

5.2.2.4 Network Coverage Model

The final objective function to be maximized is the coverage of nodes in the network. A node which is a member of cluster tree T_i, regardless of its role, is considered covered ($c_{ij} = 1$ iff $\exists T_i$ where $n_j \in T_i$). Accordingly, the only nodes in the network that are not covered are orphan nodes. Thus, the node coverage function in the network is $C_G = \sum_{i=1}^{q} \sum_{j=1}^{|T_i|} c_{ij} = k - |O|$.

5.2.3 Solutions for Optimal Monitoring in Cooperative IDS

The optimization problem described by Equations 5.4-5.10, has properties that limit the methods that can be used for solving it.

First, the number of possible cooperation topologies grows exponentially with the number of nodes. Cayley's theorem [77] gives the number of trees and clusters in a graph: 1) given n labeled nodes, the number of different trees is n^{n-2}; 2) given n labeled nodes, of which k nodes are chosen as root nodes, the number of forests (clusters) that can be formed is kn^{n-k-1}. Although, lower network densities affect trees and clusters formation, the number of possible cooperation topologies is extremely large, primarily because nodes in our clustering method are assigned different tasks. Any permutation of tasks in a cluster tree results in a new cooperative IDS solution which we need to consider (as an example, consider 3 nodes connected in a straight line; each of the nodes can be a leader, while the other two nodes can be either joined or aggregator nodes). To conclude, the exponential growth in the number of feasible solutions, i.e., $\Omega(n^n)$, makes the search space of the problem extremely large and the optimization problem NP-hard.

Second, our multi-objective optimization problem has some non-linear constraints

(e.g., constraint (3)), which makes the problem impossible to be solved by linear methods. Moreover, the solutions of our optimization problem are not numeric values, but complex cluster trees. Thus, the objective functions are discrete functions.

Given the aforementioned challenges, we propose to employ evolutionary algorithms for solving the optimization problem. In the following sections we present two evolutionary algorithms-based solutions to our MOO, and analyze their time complexity and the optimality of their solution.

5.2.4 Evolutionary Algorithms for MOO

For solving our MOO, we propose to use a Genetic Algorithm (GA), one popular type of evolutionary algorithm. A GA starts with a set of random solutions (from here on we will use the terms "solution", "individual" and "chromosome" interchangeably; consequently we will use "set of solutions" and "population" interchangeably), and derives better solutions by combining older solutions using the Darwinian process of "survival of the fittest" [8, 71]. This "survival of the fittest" process is iterative, and uses genetic operations, such as Selection, Crossover, and Mutation. Selection gives the most fit solutions chance to survive. Crossover combines solutions in the previous generation to produce offsprings (i.e., new solutions), and mutation is used to maintain genetic diversity from one generation to the next.

Our optimization problem has multiple objectives. For assessing the goodness of GA solutions, all objectives must be encapsulated in a single fitness function (which computes solution's fitness). Generally, there is no unique solution for a multi objective optimization problem. Instead, a set of solutions, called the Pareto set, is sought. Some multi-objective problems, however, can be reformulated such that some of the objectives act as constraints (also called regularization terms for objectives). In this research we pursue this latter approach primarily because Pareto

frontier produces a variety of optimal solutions with respect to different objectives. A security administrator, however, has to choose only one solution. Choosing the most suitable solution among all in the Pareto set, is an administrative decision which depends on the network situations, e.g., average battery level and attack frequency in the network. In our approach, i.e., the Penalized method, some objectives act as secondary constraints and are specified by administrator, based on the characteristics of the optimization problem. The question that remains, and will be addressed shortly, is which constraints to use as secondary, and why these constraints.

The cooperative intrusion detection techniques proposed [41, 49, 81] aim to exchange security information among intrusion detectors distributed throughout the network. In the optimal monitoring problem that we formulated, every node has a certain amount of information (obtained from its local observation) that can be exchanged with neighbors, to improve the intrusion detection rate. On one hand, the model enforces the nodes to participate in clusters, to achieve higher Information and Coverage values (i.e., we prefer to not have orphan nodes in the network). Hence, as clustering occurs in the network (to maximize Information objective, the main objective of the system), Power and Delay objectives suffer. To conclude, Information model is a primary optimization function while the other objective functions are considered secondary.

Algorithm 5 Genetic Algorithm

1: $nSol \leftarrow S$
2: Collect Network Info()
3: Create Adjacency Matrix()
4: First Population ($nSol$)
5: Fitness Values(n_Sol, α)
6: **for** $NumGen = 1$ to g **do**
7: Find Elite ($nSol$)
8: Selection($nSol$, ϕ_1)
9: Crossover($nSol$, ϕ_2, ω)
10: Mutation($nSol$, ϕ_3, φ)
11: Replace Elite($Elite$)
12: Fitness Values($nSol$, α)
13: **end for**

In the following subsections we present our GA, with pseudocode shown in Algorithm 5. Our algorithm, based on the Penalized function method for solving the MOO, uses Power, Delay, and Coverage objectives as regularization terms (or secondary constraints) for the Information objective.

5.2.4.1 Problem Encoding

First, GA solutions are encoded as bitstrings (i.e., chromosomes) of specific length, and tested for fitness. The goodness of a solution is evaluated by its fitness value. Second, a solution to our problem is a set of cluster trees $G = \{T_1, T_2, ..., T_q\}$. The set of cluster trees G can be conveniently represented as an adjacency matrix (e.g., our GA collects necessary information in Algorithm 5 Lines 2-3). Consequently, an immediate way to encode a solution (feasible or not) as a chromosome, is to con-

catenate the rows of G's adjacency matrix in a single bitstring. The chromosome is thus composed of multiple segments, where each segment is a row from G's adjacency matrix. In the example depicted in Figure 5.15, the chromosome titled "Instant Chromosome" is the encoding for the cluster tree presented in Figure 5.14(a). Based on our proposed architecture (i.e., a set of cluster trees) a chromosome must have a single bit set to 1 in each segment. Figure 5.15 also depicts two infeasible solutions (non-chromosomes) in the bitstrings titled "Infeasible Solution". In one infeasible solution, two bits in the same segment are set to 1, while in the second a node has an invalid parent. Through our proposed encoding, it is easy to observe that the length of a chromosome is k^2.

5.2.4.2 Initial Population

The initial population, consisting of S individual chromosomes -each representing a single solution- is randomly generated. A population of large size S allows the GA to start with a large set of feasible solutions. A chromosome is obtained from a randomly generated bitstring, if it abides by constraints (5.5-5.9) of our optimization problem. If the bitstring conforms, then the GA considers it a solution (Algorithm 5, Line 4).

5.2.4.3 Computation of Fitness

The fitness of a chromosome (i.e., a solution) represents its ability to keep genetic properties for next generations. As mentioned earlier, our proposed GA computes the fitness value of a solution G through a Penalized function technique (Algorithm 5 Line 5). Since any objective function can be considered as an optimizing objective, we discriminate among objectives through parameter α.

In our proposed solution, Information is considered as primary objective function, while the other objectives are secondary constraints and regularization terms. As the

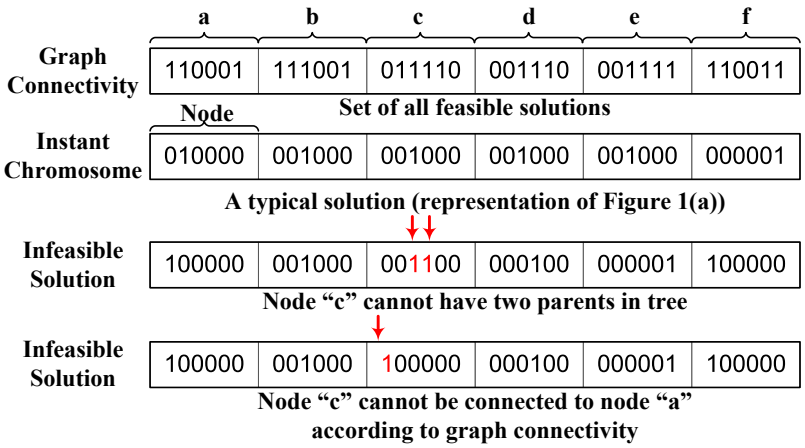


Figure 5.15: The connectivity graph is encoded as a string suitable for GA. All feasible solutions can be derived from graph connectivity string.

Power and Delay values of a solution increase, or the Coverage becomes less than 100%, the function penalizes the Information value to a smaller value, so that the corresponding solution has less chance to survive. The penalized function for MOO approach is $f(G) = [2 - (P_G/C_P) - (D_G/C_D)] \times ((1 + \lfloor C_G/k \rfloor)/2) \times H_G$, where a set of constraints penalizes the fitness value for each solution: the penalty method maximizes data entropy (H_G), while normalized values of power (P_G) and delay (D_G) operate as inverse coefficients of the fitness value. We use coverage (C_G) as a penalty parameter as well. To find the constants C_P and C_D (i.e., the maximum delay and power consumption), we solve the corresponding two single objective optimization problems. The maximum outcomes of several iterations of these SOO problems are assigned as constants C_P and C_D.

Once the fitness of the initial population is computed, our GA attempts to find better solutions in an iterative manner (Algorithm 5 Line 6), by applying genetic operations, further described below.

5.2.4.4 Elitism

The purpose of Line 7 in Algorithm 5 is to ensure that the best solution of each generation will not be lost during crossover and mutation. After computing the fitness values, the best solution among all individuals is passed on directly to the next generation.

5.2.4.5 Selection

This step chooses the best individuals whose descendants will form the next generation. Chromosomes in the current population are selected using either "tournament" or "roulette wheel" mechanisms, and are weighted with the chromosome's fitness value. In the tournament method, two random individuals are selected from the current population and the individual with the highest fitness value is selected for mating. The process is repeated S times. If the selection method is set to roulette wheel, S random chromosomes are chosen. Selection probability for each chromosome G is $f(G)/\Sigma_1^S f(G)$. Our algorithm invokes this function in Line 8, and it uses $\phi_1 = 1$ for tournament, and $\phi_1 = 2$ for roulette wheel.

5.2.4.6 Crossover

New solutions, or offsprings, are formed by mating, which is handled as an exchange in our algorithm on Line 9. Three different recombination schemes are available: block exchange (B) (where a block is a contiguous sequence of segments), segment exchange (S), and multi-segment exchange (M). Block exchange selects a random node (segment) of each chromosome and replaces all subsequent segments with the same segments of another parent. Segment exchange trades one segment from each parent with each other, while multi-segment exchanges more than one segment. Algorithm 5 Line 9 uses ϕ_2 to specify the crossover method (i.e., $\phi_2 = 1, 2, 3$).

Crossover induces change. In order to control the rate of change, the crossover probability (ω) must be set to a conservative value.

5.2.4.7 Mutation

Considering the specific requirements of our problem (e.g., cluster tree formation for cooperative IDS), our GA employs two mutation methods. Composition (C) is a mutation method useful for maximization problems while decomposition (D) is useful for minimization problems. Both methods, specified by ϕ_3 in Line 10 of Algorithm 5, randomly mutate the parent of a random node in a valid solution ($\phi_3 = 1$ means compositions and $\phi_3 = 2$ means decomposition). This may result in merging of cluster trees to form a single, larger, tree (composition), or the splitting of one tree into two (decomposition). An important observation is that mutation may also cause the loss of fittest solutions. Because of such possibility, the mutation probability (φ) must be very low.

Finally, before computing the next generation, our GA replaces the worst solution obtained from the three aforementioned processes, with the elite solution from the previous generation.

5.2.4.8 Genetic Algorithm Analysis

It is well known that the optimality of a solution obtained by GA is strongly correlated to the size of the population under consideration, and the number of generations. A large population prevents the GA from falling into a local minima and a large number of generations allows the GA to generate near-optimal solutions. As mentioned at the beginning of Section 5.2.3, the search space of our optimization problem is extremely large, especially for very large networks (e.g., hundreds to thousands of nodes). If optimality of the obtained solution is of primary concern, the execution time of our GA could still be very large (e.g., a large population and

a large number of generations are required).

Additionally, the GA does not control the distribution of leaders and their corresponding followers in the network. For example, the followers are not necessarily located evenly around their leader. It is possible to have a follower node connected to a more distant leader, then to one leader nearby. Also, the size of cluster trees is not necessarily the same. It is possible to have cluster trees varying from very small, to very large.

Motivated by the aforementioned observations, we propose a Hybrid Algorithm (HA) which, primarily, trades off solution optimality for speed of obtaining a solution. As a consequence, for very large networks, the HA algorithm might be able to obtain a better solution than GA, in a limited, short, amount of time. The proposed HA executes in two phases. In the first phase the algorithm, building on state of art solutions in leader selection [54, 85], selects a set of leader nodes that exhibit higher connectivity with follower nodes and higher residual battery charge. The selected leaders must also abide by the connectivity constraint of our MOO (e.g., the leaders and base station form a backbone communication, a connected graph). In the second phase, the HA algorithm finds the optimal role assignment for the followers of each leader, employing our proposed GA technique. We provide more details about HA in the following section.

5.2.4.9 Two-phase Hybrid Algorithm for MOO

As already mentioned, our proposed HA consists of two major phases. In the first phase, called the r-hop leader selection, our HA selects a set of suitable nodes as leaders (i.e., the leaders will form clusters in a cluster tree organization), while the second phase assigns roles to each leader's r-hop neighbors. Algorithm 6 shows the HA pseudocode.

Algorithm 6 Hybrid Algorithm

1: find F_i for $\forall i$
2: $L = \{\}$
3: **while** $O \neq \emptyset$ **do**
4: $\quad L_i = max_{i \in N \setminus b}\{|F_i| \times b_i\}$
5: $\quad L = L \cup \{L_i\}$
6: $\quad O = O - F_{L_i}$
7: $\quad$ update-followers()
8: **end while**
9: Modify-Clusters()
10: **for** $i = 1$ to $|L|$ **do**
11: $\quad$ run GA on $L_i \cup F_i$
12: **end for**

Constraints 5.9 and 5.10 of our MOO problem require the set of leaders and base station to form a connected graph, and the residual battery charge of leaders to exceed a threshold. In addition, each leader will have followers in at most r-hops. Since a larger r imposes a larger event reporting delay, our algorithm will select as leaders, nodes with more r-hop neighbors and with sufficient battery charge.

Initially, the algorithm computes F_i, the set of followers-to-be in at most r-hops, for every node i in the network (Algorithm 6 Line 1). Next, the algorithm iteratively selects leaders based on their residual battery charge and number of r-hop neighbors, if and only if they meet the connectivity requirement for leaders (Algorithm 6 Line 4). Upon selecting a node as leader, the node is added to the set of leaders L (Algorithm 6 Line 5) and all of its r-hop neighbors are removed from the set of Orphan nodes,

i.e., $O = N \setminus \{b\}$ (Algorithm 6 Line 6). Since adding/removing nodes changes the number of r-hop neighbors for the remaining nodes, the algorithms updates F_i for all unselected nodes (Algorithm 6 Line 7). The algorithm repeats the leader selection until no more orphan nodes exist.

At the end of the first phase, a set of leaders and their corresponding sets of followers are selected. We remark here that it is possible to have a few single-node clusters (i.e., Orphan nodes) at the end of the first phase, and that these orphan nodes may have many neighbors (the leader election algorithm prevented these orphan nodes/clusters from joining adjacent clusters because the fixed r). To address this issue, we add one step to the first phase of the algorithm. In this step, a single-node cluster is merged with the smallest cluster it is connected to (Algorithm 6 Line 9). This step guarantees full coverage with clusters and aims to balance cluster size.

The clusters obtained as the results of executing the first phase can now be considered as individual networks of smaller size. The nodes in these clusters can now be assigned roles for cooperative IDS. In the second phase, our proposed HA (Algorithm 6 Lines 10-12) executes the proposed GA in each cluster.

5.2.5 Simulation Results

In this section we present the performance evaluation of our proposed GA and HA. We also compare the time complexity of GA and HA and the optimality of their solutions. Finally, we evaluate the intrusion detection delay against random attack locations for both SOO and MOO (as solved using GA and HA).

5.2.5.1 Performance Evaluation of GA

We implemented the proposed GA in Matlab for SOO (for each objective) and MOO. We investigate SOO for two reasons: to obtain the scaling factors C_P and C_D of the penalized MOO, and to compare SOO solutions with those obtained from

penalized MOO. For comparison we use GA convergence and the balance in cluster formation. Before comparing SOO and MOO solutions, we are interested in the effects various algorithm parameters (e.g., genetic operations, population size, number of generations, etc.) have on SOO/MOO solution optimality. To find the best combination of genetic operations for our problem, we investigate all combinations of selection, crossover, and mutation operations. The probability of crossover and mutation are set to 65% and 10%, respectively. Other GA parameters are listed in Table 5.1. These values are chosen after several evaluations of GA convergence history and final fitness values.

As mentioned in Section 5.2.4.8, the population size must be chosen based on the network size, to ensure GA does not fall into a local minima in the search space. The papulation size in our simulation is set to $\text{Max}\{50, 2 \times k\}$. The number of generations, i.e., $200 \times k$, also varies with the network size and is set large enough so that crossover and mutation operations get sufficient chance to generate variety of offsprings. We perform simulations for different network deployments and network sizes: uniform deployment in a grid topology of 9, 16, 25, 36, and 49 nodes; and random deployment of 10, 20, 30, 40, and 49 nodes.

Effects of Genetic Operations on Performance: As presented in Section 5.2.4, our GA employs different methods for selection, crossover and mutation. We solve the penalized MOO problem starting from the same initial population and applying all twelve combinations for genetic operations. We use a 3-digit number, called combination ID, to indicate each combination where the first digit specifies the selection method and two next digits determine crossover and mutation methods, respectively (i.e., Cmb. ID = $\phi_1\phi_2\phi_3$). For example, combination ID 121 uses the "tournament" selection method, "segment exchange" crossover method, and "composition" mutation method.

Table 5.1: GA Parameters

Selection Meth.(ϕ_1)	Tournament, Roulette wheel
Crossover Meth.(ϕ_2)	Block, Segment, Multi Seg.
Mutation Meth.(ϕ_3)	Composition, Decomposition
Crossover Prob.(ω)	65%
Mutation Prob.(φ)	10%
Population Size(S)	Max {50, 2 $\times$ Network Size}
# of Generations (g)	200 $\times$ Network Size

Table 5.2 depicts the final fitness values of penalized function for both random and grid networks. To find the best genetic operations combination, we compare the average objective value for each combination, over all network sizes. Since the objective value varies with the network size, the objective values for different networks are normalized using the maximum value achieved for the corresponding network size. The average normalized values are shown in the rightmost column of Table 5.2.

As shown, combination 211 has the highest average objective value among all of the combinations. It is worth mentioning that the objective value for combinations that use the decomposition method for mutation, is always less than the objective value obtained by composition. One explanation for this is that the decomposition method tends to split cluster trees into smaller ones, an undesired goal in maximization problems in which larger cluster trees are desired.

Table 5.2 also depicts the standard deviation (STD) of the objective values. We use this metric to show how much variation there is in the objective values, due to different combinations. As we mentioned earlier, the values in columns are in different ranges, because of different network sizes. It is interesting to observe that the fitness value for two equal size networks can also be in different ranges, due to the deployment type (e.g., 49-node grid or random network). This is because for

Table 5.2: Optimal objective value of penalized function achieved by different genetic operations

	Random					Grid					
Cmb.	**10**	**20**	**30**	**40**	**49**	**9**	**16**	**25**	**36**	**49**	**Norm.**
111	27.72	61.40	99.01	151.69	197.53	25.47	47.00	80.39	141.45	208.20	**0.992**
112	27.21	58.65	94.73	143.25	185.16	24.49	45.83	76.18	137.07	199.87	**0.951**
121	27.72	61.40	99.21	151.44	198.45	25.47	46.60	80.32	141.82	207.61	**0.992**
122	26.73	59.44	95.28	146.66	187.88	24.49	45.69	75.76	134.90	199.82	**0.953**
131	27.72	61.40	97.87	150.81	197.32	25.47	46.48	80.64	142.00	207.33	**0.989**
132	26.76	58.93	94.85	148.02	187.01	24.37	45.68	76.78	137.26	197.74	**0.953**
211	27.82	61.58	99.47	152.10	200.34	25.47	47.14	80.61	142.46	210.25	**0.997**
212	26.76	58.18	95.11	142.83	186.38	24.81	45.85	77.44	134.97	198.09	**0.950**
221	28.06	61.76	99.80	150.34	199.83	25.47	47.03	80.96	142.24	208.75	**0.996**
222	26.27	59.60	96.83	144.79	189.20	24.81	45.82	76.64	135.60	199.96	**0.955**
231	27.82	61.76	99.55	151.08	199.47	25.47	47.03	80.61	143.19	207.12	**0.995**
232	26.67	59.19	95.14	144.60	186.39	24.72	45.68	76.63	134.31	198.77	**0.951**
Avg.	27.27	60.27	97.24	148.13	192.91	25.04	46.32	78.58	138.94	203.63	
STD	0.60	1.38	2.11	3.53	6.29	0.46	0.61	2.13	3.52	4.89	
CV	0.02	0.02	0.02	0.02	0.03	0.01	0.01	0.02	0.02	0.02	

Information as objective, the value is a function of the distance between nodes, which varies in a random deployment, and is constant in a grid network. Consequently, to compare objective values, we use the coefficient of variation (CV), defined as the ratio of the standard deviation to the mean value. Small values of CV (i.e., $CV << 1$) indicate that all genetic operations perform well. We also note here that larger networks have larger CV values, since the search space is considerably larger (this imposes a higher variation in the final fitness value).

Figures 5.16(a) and 5.16(b) show the convergence history for the objective value in 49-node random and grid networks, respectively, when only composition is considered. First, the methods that use "tournament" selection (i.e., 111, 121, and 131) help GA remove solutions with very low objective value (i.e., bad solutions) from the population, during early generations. Removing those solutions, however, prevents the algorithm to create near-optimal solutions from bad solutions by genetic operations (e.g., mutation). Thus, such approaches achieve high objective values in the early generations, which do not improve in subsequent generations because the

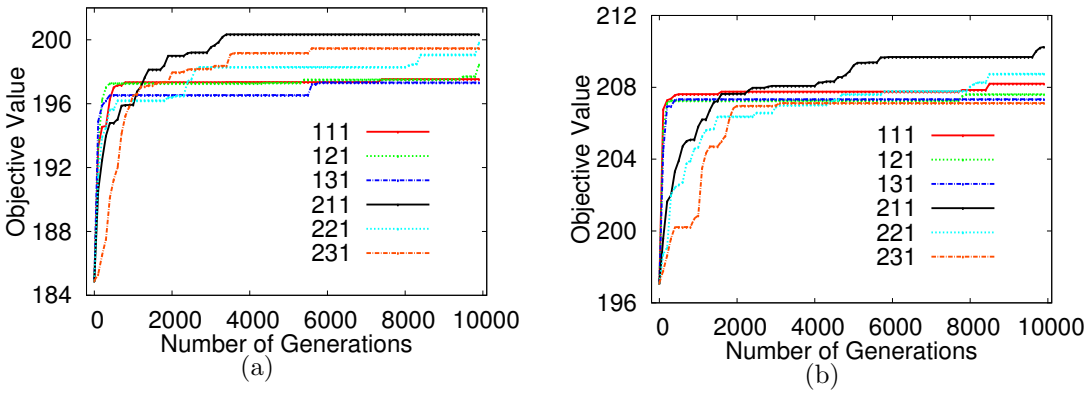


Figure 5.16: Convergence of penalized objective value using different combinations of genetic operations for: (a) 49-node Random network; (b) 49-node Grid network.

algorithm falls in a local minimum. These approaches are useful when optimally is not as important as algorithm execution time. For fast solutions, the number of generations can be small, with GA obtaining a near-optimal solution. Second, the "roulette wheel" selection, can usually pass a few undesired solutions to the next generations where a mutation process may produce a desired solution. From here on, all simulations use genetic combination 211, i.e., "roulette wheel" for selection, "block exchange" for crossover, and "composition" for mutation.

Effects of Initial Population on Performance: Evolutionary algorithms start with a random initial population as a set of sample solutions from the entire search space. Hence, they may lack robustness in producing optimal solutions, owing to their probabilistic nature (i.e., the genetic operations and the initial population). To show the robustness of our algorithm, we run simulations on different network sizes using the best combination (211) found in the previous experiment. This experiment is repeated 10 times for each network size, starting with different initial populations for each run. We reduce the number of generations to 6,000 generations (i.e., $\min\{6000, 200 \times k\}$) because the major improvements in fitness value for the

larger networks occurred before 6,000 iterations.

Table 5.3: Optimal objective value of penalized function when combination 211 is applied on different initial populations

	Random					Grid				
Run ID	**10**	**20**	**30**	**40**	**49**	**9**	**16**	**25**	**36**	**49**
1	27.58	61.80	99.09	151.91	199.25	25.47	47.16	80.49	143.65	203.00
2	27.78	61.32	98.24	150.97	198.24	25.47	47.53	81.10	142.37	209.79
3	27.58	61.75	98.41	151.96	194.41	25.47	47.77	80.81	142.94	209.35
4	27.72	61.76	99.50	151.81	193.76	25.47	47.03	80.55	143.13	207.39
5	27.58	61.76	99.38	150.38	196.57	25.47	47.77	81.01	141.71	208.63
6	27.58	61.58	99.39	152.97	198.93	25.47	47.53	81.11	142.08	209.61
7	27.58	61.80	99.45	152.93	196.28	25.47	47.15	81.30	143.00	204.69
8	27.58	61.58	99.01	150.84	195.75	25.47	47.53	80.62	142.37	207.12
9	27.58	61.75	99.55	151.56	197.27	25.47	47.16	81.52	143.48	209.81
10	27.58	61.75	99.02	151.46	199.89	25.47	47.15	81.06	142.14	208.58
Average	27.61	61.69	99.11	151.68	197.04	25.47	47.38	80.96	142.69	207.80
STD	0.07	0.15	0.45	0.83	2.06	0.00	0.27	0.33	0.64	2.31
CV	0.002	0.002	0.004	0.005	0.01	0	0.005	0.004	0.004	0.01

Table 5.3 shows the average fitness values for the penalized function over all experiments. As shown, the CV values are extremely small (i.e., $CV << 0.1$), thus the dispersion in the final values is negligible. Considering the smaller number of generations in this experiment and comparing the CV values with Table 5.2, the proposed algorithm is robust in achieving near-optimal solutions regardless of the initial population.

Figures 5.17(a) and 5.17(b) depict convergence history for 49-node random and grid networks, respectively. Although we show only seven instances in each figure, due to space limit, the graphs contain the whole distribution of final values. It is also worth mentioning that the convergence is smoother, when compared with the results presented in Figures 5.16(a) and 5.16(b), where different genetic combinations were used.

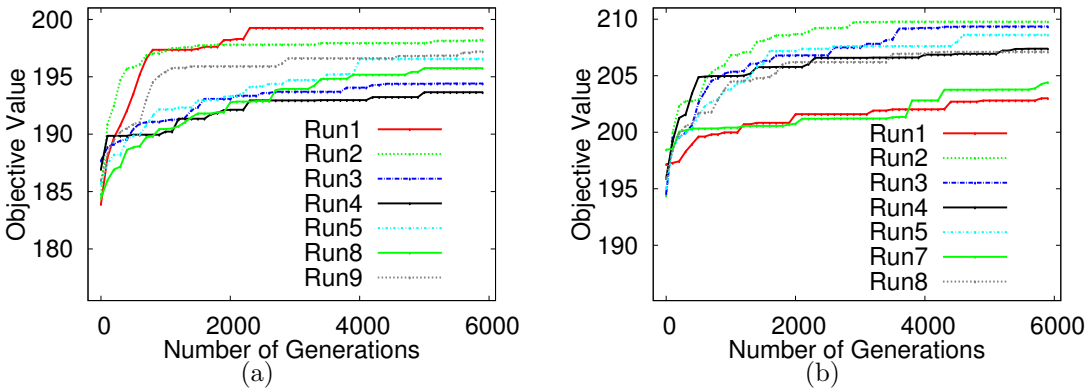


Figure 5.17: Convergence of penalized objective value using different initial population and same genetic operations. (a) 49-node Random network. (b) 49-node Grid network

Cluster Count RMSE: As mentioned, the larger the clusters are, the higher their Information objective value is. Thus, it is obvious that the maximum Information is achieved in a single large cluster. However, other objectives, namely Power and Delay, are minimal when nodes are individual clusters without followers. The Power and Delay minimization objectives prevent formations of large clusters, thus trading off high Information for low Power/Delay.

Considering the importance of the Information objective in our cooperative IDS, one may be tempted to consider only the SOO problem that maximizes Information and Coverage, and to disregard minimizing the Power and Delay. This SOO formulation will seek as solution a single large cluster tree. This solution presents a single point of failure and will incur high energy consumption near the leader. Additionally, the delay in intrusion detection and power consumption will likely be very high. Instead, we prefer that GA produces a set of clusters, as indicated by constraint 5.10 in our problem formulation. By tuning the parameter $|T|_{th}$, we tradeoff fault-tolerance (i.e., number of cluster trees) with compromised node-tolerance (i.e., maximum number of nodes that must be compromised before a system fails).

Table 5.4: RMSE value for number of cluster trees

	Random					Grid				
	10	**20**	**30**	**40**	**49**	**9**	**16**	**25**	**36**	**49**
RMSE - SOO	0	0	0.04	0.07	0.02	0	0	0	0.01	0.02
RMSE - MOO	0.25	0.44	0.57	1.2	0.72	0	0.11	0.1	0.17	0.32

In this section, we investigate how the parameter $|T|_{th}$ affects the number of cluster trees. We chose different values, depending on the network size: 5 for 10-node network and 7 for 20-node network. By changing $|T|_{th}$ we expect that solutions to the SOO and MOO problems will have different numbers of cluster trees. We use the root mean square error (RMSE) in the cluster count, to show the bias and variance of number of cluster trees obtained by the GA. We normalize the expected number of cluster trees, for different network sizes to 1 and then calculate RMSE for each network size. Tables 5.4 depicts the RMSE of our estimator's expected value (number of clusters) for both SOO and MOO approaches in different deployments (i.e., random and uniform) and network sizes.

As shown, the cluster count RMSE for the SOO formulation, is very close to zero and it is smaller than that for the MOO formulation. The explanation for this is that SOO, in which Information is the objective function, tends to create as large clusters as possible. Consequently, the number of clusters we expect (as specified by $|T|_{th}$) is achieved. MOO formulation, however, avoids solutions with large clusters because of high Power and Delay costs. The cluster count RMSE for both SOO and MOO approaches increases with the network size. The increasing RMSE can be explained by the fact that for larger size networks the search space is much larger, thus preventing the GA from reaching the optimal value in a limited amount of time. Finally, very small RMSE (i.e., close to zero) throughout this experiment, indicates that solutions produced by our GA, in terms of number of cluster trees, approach an

intuitively optimal value.

SOO Solutions vs. MOO Solutions: In this section, we compare the convergence and cluster tree formation for SOO and MOO, given a fixed amount of time and a set of random initial populations. We investigate the convergence to the best fitness values, of the GA for both random and grid networks to compare their speed and monotony. We also show how considering more objectives in the optimization problem affects clustering, preventing GA from producing long linear cluster trees.

Figures 5.18(a) and 5.18(b) show the convergence of the objective value for 49-node random and grid networks, respectively. The figures show that the objective value for SOO converges faster than for MOO, which fluctuates. The explanation for this is as follows. Starting with an initial population, SOO selects better solutions in terms of Information objective value. These solutions are passed to the next generation without restrictions (e.g., high Power value), causing the GA to converge very fast. A good solution for MOO, however, is a solution that meets all objectives. Thus, it is possible to have a solution with a lower value of Information, passed to the next generation, instead of a solution with higher Information value, simply because the penalized value of the former is better. This happens because maximization and minimization objectives are not independent.

Figures 5.19(a) and 5.19(c) depict cluster tree solutions obtained by the GA, when the objective to maximize is Information, without considering Power or Delay. The obtained topologies are largely linear and, while producing solutions that maximize Information, incur high Power and Delay costs. To assess the effect of the penalized fitness function on these solutions, we ran our GA on the same networks, using Power and Delay as penalties to the fitness function. Figures 5.19(b) and 5.19(d) show that smaller cluster trees, which achieve a balance between all objectives, are obtained. It is also shown how SOO achieves less number of cluster trees of larger

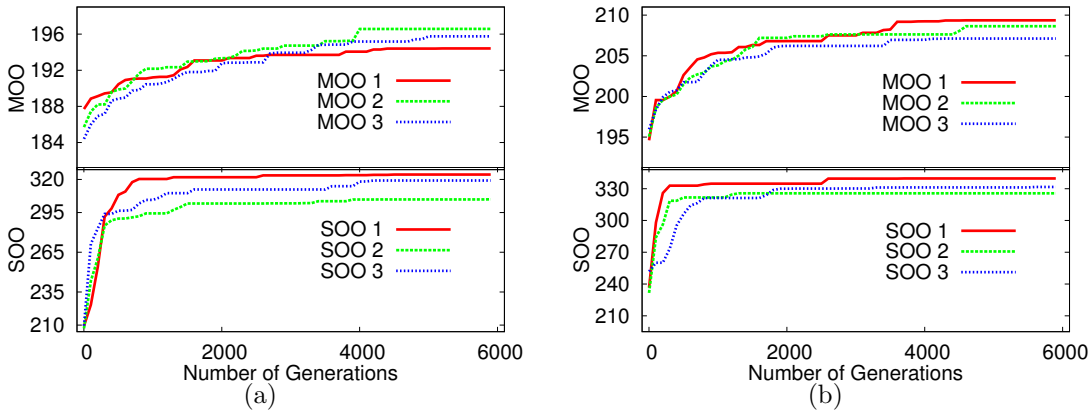


Figure 5.18: Convergence of SOO and MOO objective values in 49-node networks: (a) Random; (b) Grid.

sizes while MOO benefits from more cluster trees of smaller sizes, a tradeoff among all objectives.

Pareto Curve: A Pareto curve shows the non-dominated solution space of one objective, i.e., Power, Information or Delay, in terms of two other objectives. It presents the optimal front and the diversity across objectives along this front. We performed simulations using a penalized MOO on a 49-node random network. The Pareto curve of the Penalized Function is shown in Figure 5.20. As shown, the Pareto curve contains solutions with different objective values for Information, but the average penalized value of them is definitely the maximum possible values that GA could achieve during generations. The Pareto curve also shows that Power, Information and Delay are highly correlated, giving one other explanation for the observed fluctuations in the convergence graph of the multi-objective fitness value. It is worth mentioning that in all other experiments, we chose a solution with a maximum penalized value. This allows us to compare final objective values of multiple approaches (i.e., combination of different GA operations, initial solution set, etc.) for convergence and cluster formation results.

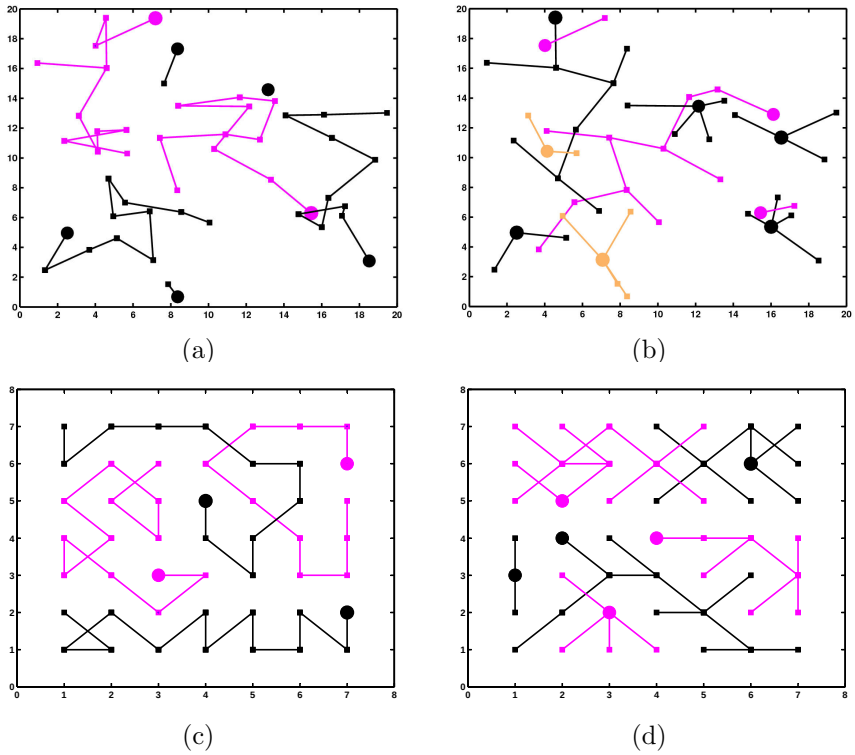


Figure 5.19: Effects of constraints on the output of the optimization problem: (a) SOO of Information in 49-node Random. (b) Penalized MOO of Information in 49-node Random. (c) SOO of Information in 49-node Grid. (d) Penalized MOO of Information in 49-node Grid.

Given the Pareto curve, we can identify the maximum possible value for Information in a given network, while the other objectives are constrained. For example, maximization of Information may be used as the objective function while Power, Delay and Coverage objectives are constrained to threshold values P^{th}, D^{th}, $|N \setminus \{b\}| = k$, as $\sum_{i=1}^{q} \sum_{j=1}^{|T_i|} p_{ij} \leq P^{th}$, $\sum_{i=1}^{q} \sum_{j=1}^{|T_i|} d_{ij} \leq D^{th}$ and $\sum_{i=1}^{q} \sum_{j=1}^{|T_i|} c_{ij} = k$. This set of solutions can help a network administrator choose the optimal one at any time, based on the importance of objectives. For example, in a newly deployed network, which benefits from fully charged nodes, an optimal solution can be chosen from those with higher Information, even if more power is consumed. The same

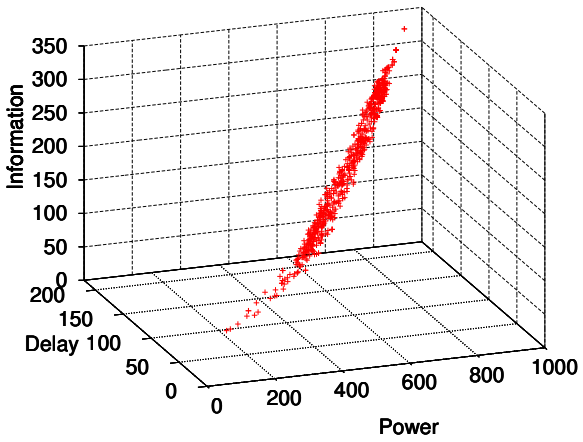


Figure 5.20: A Pareto diagram emphasizes the relationship among objective functions. The high degree of correlation is evidenced by the narrow surface displayed.

reasoning can be applied to scenarios where the network might be under attack and intrusion detection delay is important.

5.2.5.2 Performance Evaluation of HA

In this section we compare GA and HA. We implemented HA in Matlab for both SOO and MOO. Considering our analysis in Section 5.2.4.8 we are interested in the execution time and solution optimality. We show how HA provides solutions consisting of well-formed cluster trees considerably faster than GA, while the optimality of the SOO solution is still close to that of GA (for small networks). We will remark that HA may provide more optimal solutions than GA, especially for larger networks. We will also investigate HA solutions for our MOO problem.

First, we investigate how HA forms clusters for both SOO and MOO. The cluster trees for grid and random topologies shown in Figure 5.19 are depicted in Figure 5.21. As shown in Figures 5.21(a) and 5.21(c), the cluster trees produced for SOO formulation (i.e., Information) have similar characteristics: long linear topologies with many aggregator nodes. The HA solution does not contain single-node or

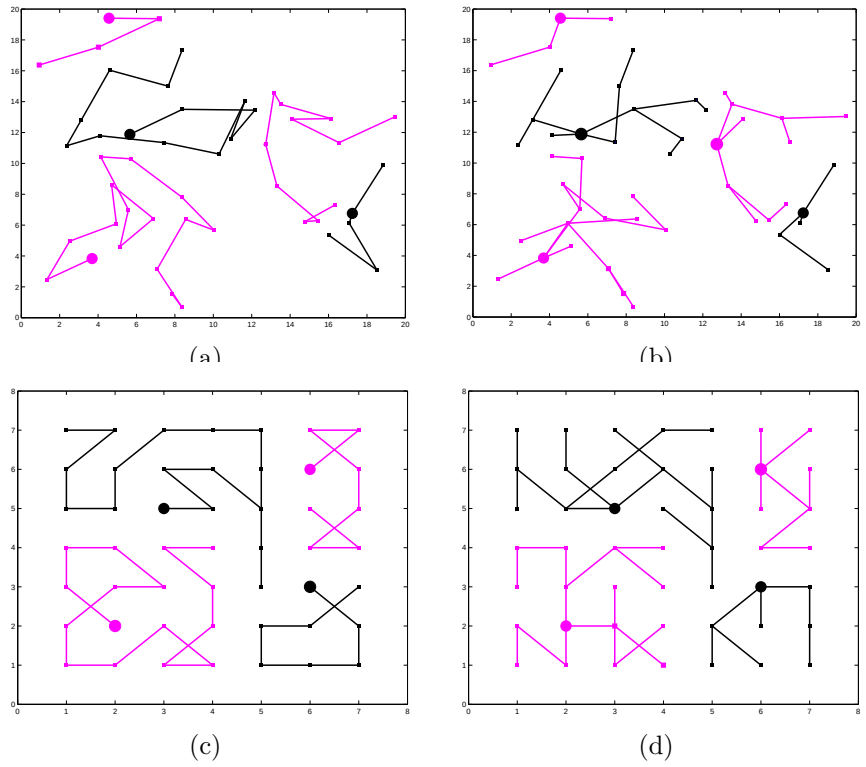


Figure 5.21: Cluster trees obtained by HA from the same networks as Figure 5.19: (a) SOO of Information in a 49-node Random. (b) Penalized MOO of Information in a 49-node Random. (c) SOO of Information in 49-node Grid. (d) Penalized MOO of Information in 49-node Grid.

2-node cluster trees (e.g., Figure 5.19(a) vs. Figure 5.21(a)). Moreover, as depicted in Figures 5.21(b) and 5.21(d), the produced cluster trees for MOO problem formulation are completely separated from each other (i.e., there is no overlap and no link crossing between clusters). As presented before, the number of cluster trees is a metric indicating the robustness of our algorithm. It is worth mentioning that HA has stronger control over this metric, as we can see in the results depicted in Figure 5.21(b) where we obtain 5 clusters (as expected in a 50-node network). We will show, however, that this balanced clustering incurs higher Power and Delay costs in the MOO formulation, where HA has less control over the minimization objectives.

Next, we investigate execution time and SOO solution optimality. As we mentioned in Section 5.2.4.8, the execution time of GA and optimality of solutions are highly correlated with the network size, number of initial solutions and number of generations. From extensive simulations we observed that for larger networks, we need more initial solutions and generations for reaching the optimal solution. Hence, to have a fair comparison for execution time and optimality of GA and HA, we experimentally define a relation between network size and number of initial solutions and generations: the number of initial solutions is twice the network size, and the number of generations is $\min\{6000, 2 \times k\}$. Figures 5.22(a) and 5.22(c) depict the execution times of GA and HA for solving SOO, i.e., Information objective, for different network sizes and deployments. As shown, GA obtains a solution for 49-node random network in more than 400 minutes (using the values we specified for number of solutions and generations). HA splits the network into five smaller clusters, and runs GA on each cluster. The execution time for HA is the summation of execution time of GA on smaller clusters, 35 minutes in total.

We note here that the execution times for both GA and HA are obtained from running the simulation on Dell PC with 3 GHz Intel Core 2 Duo CPU and 4 GB RAM. The proposed algorithms (GA and HA) are not parallelized and are run only on one core. However, considering the currently powerful PCs (using 8-core CPU) that can be used as the base station in our system model, and the possibility of parallelized implementation of GA and HA, the execution time would be much smaller.

Finally, we compare solutions optimality for GA and HA when Information is considered as the only maximization objective. Considering the Information model discussed in Section 5.2.2.2, $H_G = \Sigma_{i=1}^{q} \Sigma_{j=1}^{|T_i|} h_{ij}$ is a function of the distance between communicating pair of tasked nodes. The maximum possible Information in a highly dense network (given a number of nodes) is much smaller when compared with a low

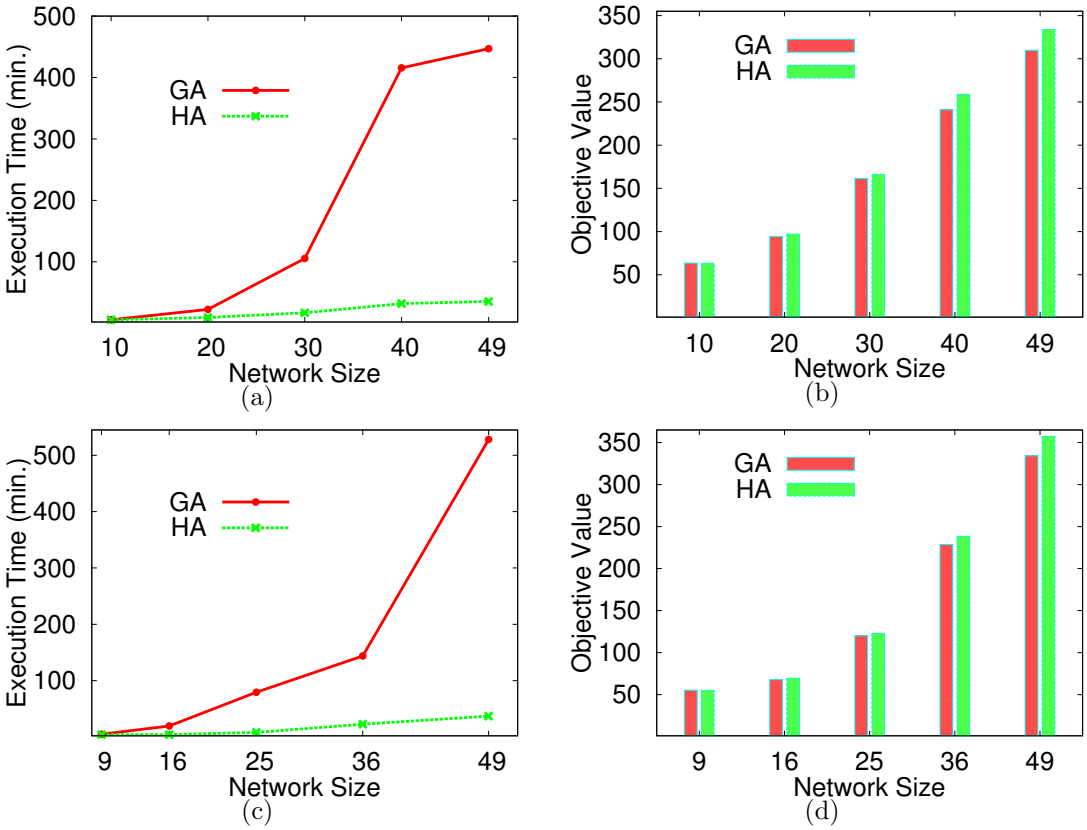


Figure 5.22: Comparison between the execution time and optimality of the solutions both GA and Hybrid solutions for different network sizes. (a)(b) Random Network. (c)(d) Grid Network.

density deployment of the same number of nodes. Hence, the final value for SOO can only be compared when the same deployment topology is used. As a result, the optimal objective values for the 49-node random deployment can not be directly compared with the 49-node grid deployment.

Figures 5.22(b) and 5.22(d) depict the objective value of Information model for different networks sizes. For a given network, we performed simulations for both GA and HA with different random seeds. We assigned random residual battery levels to nodes and started simulations with different initial solutions. We then compared the solutions' optimality. As we can observe from results, for a given population size and

number of generations, there is not much difference between the optimality of the solutions produced by GA and HA for small network sizes. However, HA produces better solutions for larger networks. We remark here again that the scale for the objective value is not relevant since network sizes and deployments are different.

Considering the first phase of HA, i.e., a heuristic approach, a solution produced by HA is expected to have large cluster trees of equal size and, possibly a few smaller ones, depending on r and node density. This means the HA solutions potentially have larger Information objective values, because of the large clusters. GA, on the other hand, may not reach near-optimal objective values in the specified number of generations. Thus, HA has better performance than GA, when maximum Information is needed. Considering the high correlation between Information, Delay, and Power, however, HA cannot produce optimal solutions for MOO. Since the potential clusters are formed in the first phase of HA, there will be a relatively high lower bound for Power and Delay, which HA can not lower significantly.

5.2.5.3 Intrusion Detection Delay

In this section, we investigate the intrusion detection delay for insider and outsider (randomly distributed in the network) attackers. The cooperative IDS solutions we focus on are those obtained in Section 5.1.5 where the proposed GA and HA were used for solving SOO and MOO problems. The results are depicted in Figure 5.19(a) and Figure 5.19(b) for GA and Figure 5.21(a) and Figure 5.21(b) for HA, respectively. For this investigation, we considered the most sophisticated attack reported in this research, namely *Web exploits*. In our scenario, a leader node within the radio range of an outsider attacker, can detect the attack with a delay of 0 time units (i.e., timeslots employed by our TDMA protocol), since a HW-DS intrusion detection engine is employed by a leader.

In our simulations, we considered 10^5 random locations for the outsider attacker. For the insider attacker case, we randomly chose non-leader nodes as attackers. We estimated the detection delay as the shortest hop count between a node within attacker's radio range, and node's leader. Our simulation results for an outsider attacker scenario against GA solution are presented in Figure 5.23(a), where two topologies are analyzed (i.e., Figures 5.19(a) and 5.19(b)). Our results indicate that most attacks will be detected in 0 or 1 hops, if the solution is obtained by solving the MOO (Figure 5.19(b)). If the solution used is the one obtained from solving the SOO (Figure 5.19(a)), then the detection delay is much larger. More precisely, the mean detection delay for the SOO solution is 2.28, and for the MOO solution it is 0.58.

The results for the same attack scenarios against HA solutions (i.e., Figures 5.21(a) and 5.21(b)) are shown in Figure 5.23(b). The mean detection delay for the SOO solution is 2.37, and for the MOO solution it is 0.87, slightly larger than the GA solutions. Simulation results for the insider attacker scenario, for GA and HA solutions are presented in Figures 5.23(c) and 5.23(d), respectively. Similar with the outsider attacker scenario, a GA solution for the MOO problem is able to detect an insider attacker much quicker than the case when the solution is only for the SOO problem formulation. For the insider attacker scenario against GA solutions, the mean detection delay for the SOO solution is 2.3, and for the MOO solution it is 0.25. However, these values are respectively 2.12 and 0.58 in SOO and MOO solutions produced by HA, which means the SOO solution of HA is better than GA, but its MOO still has larger delay compared with GA solution.

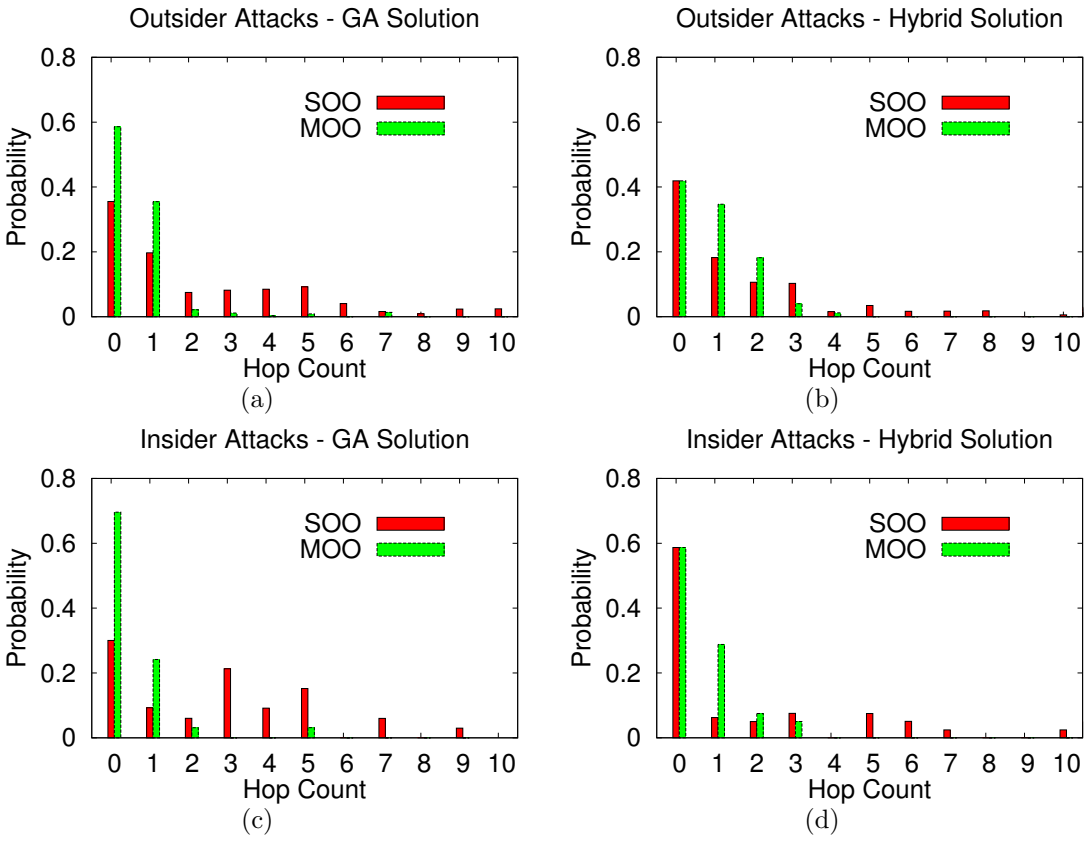


Figure 5.23: Probability mass function for delay in reporting attacks to the nearest leader for different cluster trees of a 50-node network produced by GA and HA.

5.2.6 System Evaluation

As a proof of concept of our proposed architecture for cooperative IDS, we developed an adhoc network, consisting of six laptops, and deployed it in an indoor area. The two laptop configurations were: 1.8 GHz Intel processor with 2 MB cache and 1 GB RAM running Linux (Ubuntu 6.10); and Intel Core 2 Duo 2 GHz, 2 MB cache, 4 GB RAM running Windows 7. These hardware configurations were used by both attacker and target nodes, to mainly show that our proposed solution is not OS dependent and that any detection engine and operating system could be used. For the intrusion detection engine we used Snort [78] in NIDS mode, the most complex

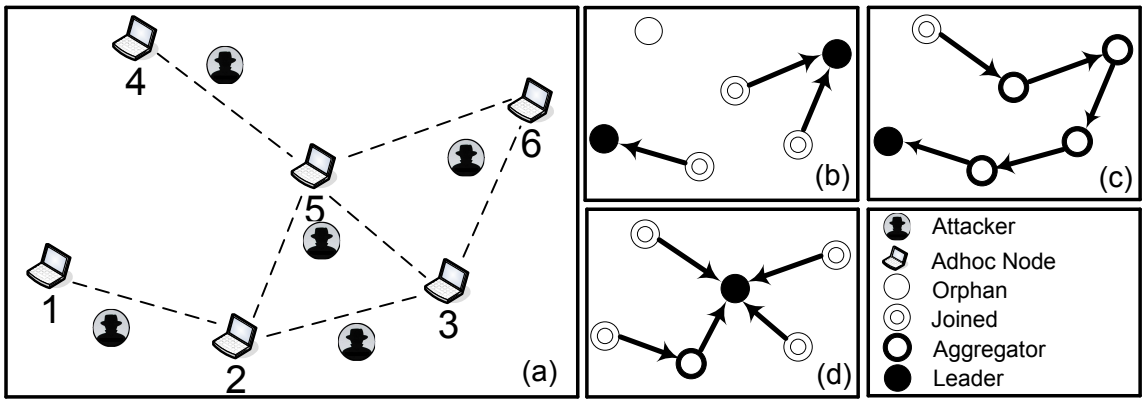


Figure 5.24: (a) Connectivity graph of adhoc network and random attacker locations. (b) Random solution. (c) SOO solution. (d) MOO solution.

and configurable mode, that analyzes captured network traffic to find any matches between them and activated rule sets [78].

To have different configurations of Snort with diverse power, memory and CPU consumption and detection power (i.e., LW-DS, RE-DS,and HW-DS), we disabled portions of rulesets. We disabled 2/3 of detection rules (excluding *bad-traffic* and *icmp*) in the LW-DS configuration (run by either orphan or joined nodes). Thus, nodes that had LW-DS version only detected flooding attacks. For aggregator nodes that used the RE-DS configuration, only 1/3 of rules (excluding *bad-traffic icmp* and *scan*), were disabled, so that flooding and port scanning attacks could be detected. Leader nodes ran the HW-DS version of Snort which used all rulesets for detecting any possible attack, among the three we consider in this research. Although we did not consider event reporting among nodes, the different types of detection engines we used (i.e, LW-DS, RE-DS and HW-DS) emulated such costs, if we had reporting capabilities available. The locations of the 6 adhoc nodes were randomly chosen, and are shown in Figure 5.24(a). We selected 5 random positions for the outsider attacker. Attackers were enabled to run the 3 types of attacks considered in this

research. Adhoc nodes had different configurations of Snort depending on their assigned role by our proposed cooperative IDS architecture.

To evaluate the effectiveness of our collaborative IDS, we evaluated 5 different cluster tree topologies. Two were obtained from the proposed GA for SOO and MOO problems. These solutions are depicted in Figure 5.24(c) and (d), respectively. A third solution we considered was a "Random" solution, shown in Figure 5.24(b), which assigns detection responsibilities in a random manner. The final two solutions we considered (not presented in a figure) were: "All LW", in which all 5 nodes execute the LW-DS; and "All HW", in which all 5 nodes execute the HW-DS detection engine.

The results show that the "All HW" scenario achieves a detection rate (DR) of 100%. In this solution, however, all nodes execute the most sophisticated IDS, and thus exhaust resources the fastest. At the other extreme, when all nodes execute the lightest IDS, the detection rate suffers - it was only 33%. The solution obtained by the proposed GA that solves both SOO and MOO achieve high detection rates of 73% and 93%, respectively, superior to a random assignment of roles to nodes, which achieves only a 60% detection rate. Consequently, the MOO solution achieves a higher intrusion detection rate, when compared with a random role assignment [41] and with the SOO solution (i.e., an indicator of state of art solutions [46, 47] that consider only one objective), while ensuring maximum Information. Both MOO and SOO solutions would achieve 100% detection rate if an event reporting/correlation mechanism is used. MOO, however, would still provide lower detection delay and lower power consumption.

5.2.7 Discussion

5.2.7.1 Cluster Reformation: Conditions and Solutions

In this research we consider static resource constrained wireless networks, e.g., static mesh or sensor networks or a combination of them. We assume that some nodes might be relocated for administrative reasons, that nodes may run out of power, become faulty or become compromised by attackers. Also, leader's residual energy might fall below a threshold. Despite all these network condition changes, *the network topology does not change often* (or not as often as in mobile adhoc networks). We remark that it is still necessary for the base station to periodically collect the network information in order to decide whether cluster reformation is required or not. Considering the expected infrequent nature of network topology (i.e., the information collected by base station most likely reflects the most recent network situation), the cluster reformation interval will be much longer than for mobile adhoc networks.

We learned from our performance evaluation results that the execution time of GA for large networks is very high, i.e., the algorithm execution time increases exponentially with the network size. Dividing a large network into smaller subnetworks, as HA does, helps the algorithm execute faster. Once a change in the network triggers cluster-reformation, it is desired to re-execute the GA or HA algorithms only in the subnetwork in which the change has occurred, instead of the whole network. For example, once a leader can no longer hold the leadership role, an efficient solution is to upgrade one of its children to leader role, if feasible (e.g., upgrading a child node to a leader role might be forbidden by a constraint in our problem formulation). We also note that some network changes may cause a cluster to be disconnected from neighboring clusters, which will require a rearrangement in clusters. In this case, for sake of efficiency, we can re-run the algorithm for a subset of neighboring clusters,

instead of the entire network.

5.2.7.2 Intrusion Detection Engine

From intrusion detection engine perspective, systems can be categorized in three classes; a) anomaly-based intrusion detection, b) misuse-based intrusion detection, and c) hybrid intrusion detection, which is a combination of the former two. One may argue that our proposed system can only detect known attacks since Snort is a signature-based detection engine. However, unknown attacks would also be detected if the intrusion detection engine would employ anomaly-based inspection (as claimed in Snort Website). An another example of an anomaly-based engine employed by an off-the-shelf IDS, is Bro [65], which is able to discern traffic anomalies and to detect unknown attacks. Thus, our proposed cooperative IDS is not limited to detecting only known attacks, but also stealth attack with appropriate *Local IDS* and *Cooperation* modules substitutions. The problem of dealing with unknown attacks through anomaly-based intrusion detection has received some attention from research community [12, 32, 51, 91].

5.2.7.3 Scalability of HA

The idea behind a hybrid algorithm (HA) is to divide the optimization problem such that the algorithm execution time for larger networks is reduced considerably. When partitioning a large network into smaller subnetworks, the size of each subnetwork is an important parameter to be considered. On one hand, the size of a subnetwork/cluster is limited by a predefined threshold, as constraint $|T_i| \leq |T|_{th}$ shows. The reasons for considering an upper bound for cluster size are: 1) to avoid a single point of failure in large clusters; 2) to limit the maximum Delay and Power; 2) to avoid scenarios where HA needs to handle large subnetworks, because of, potentially, high execution time. On the other hand, dividing the network into many

smaller subnetworks is not desired since a higher objective value for Information (i.e., primary objective in cooperative IDS - as we discussed in Section 5.2.4) is obtained from larger clusters. Consequently, the parameter r in HA has to be carefully chosen, so that a high objective value for Information can still be obtained, while cluster sizes are smaller than $|T|_{th}$. Considering the network sizes we used for performance evaluation in this research, $r = 2$ is the best choice for HA; as $r = 1$ produces less optimal solutions and $r \geq 3$ leads to infeasible solutions (because of large subnetworks/clusters). For larger networks, which we have not considered in this research (e.g., 100 nodes or larger), $|T|_{th}$ and consequently r might need to be set to larger values.

6. RESOURCEFUL AND TRAFFIC-AGNOSTIC IDS

This section* focuses on battery-powered WMN consisting of resourceful nodes that are able to run full IDS. We assume that the complexity of network topology and user traffic does not allow the security administrator to have up-to-date and accurate information about traffic paths and he/she has to monitor the entire network traffic on all communication links. Our motivating application is DistressNet [18,28], a system, developed for situation management in disaster response. In DistressNet, WMN are used for providing an infrastructure in triage areas for collecting physiological data from victims and in the disaster area for communication among emergency responders. Since in disaster areas electric power is almost always unavailable (see earthquake and tsunami disaster in Japan 2011, with energy blackouts going as far as 200+ miles away from the affected area), DistressNet needs to operate predominantly on batteries. Battery powered WMN pose major challenges given the typical high power consumption of mesh nodes. Despite the attention energy efficient operation in WMN has received [7, 24, 57], there is no provision in the 802.11s standard for power saving mode operation. This led to the absence of mesh node hardware that operates in a power saving mode. Given the urgent need for deploying DistressNet, we are proposing, as a first step for energy efficient operation, to allow mesh nodes, when feasible, to duty-cycle by turning on-off their wireless interfaces. As we uncover experimentally, the duty-cycling has an interesting effect, in that it allows the battery to recover some of its capacity, thus allowing for a longer total operation time.

*Parts of this section are reprinted with permission from "Energy efficient monitoring for intrusion detection in battery-powered wireless mesh networks" by Amin Hassanzadeh, Radu Stoleru, and Basem Shihada, In Proceedings of the 10th International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW), pages 44-57, Paderborn, Germany, 2011., Copyright 2011 by Springer.

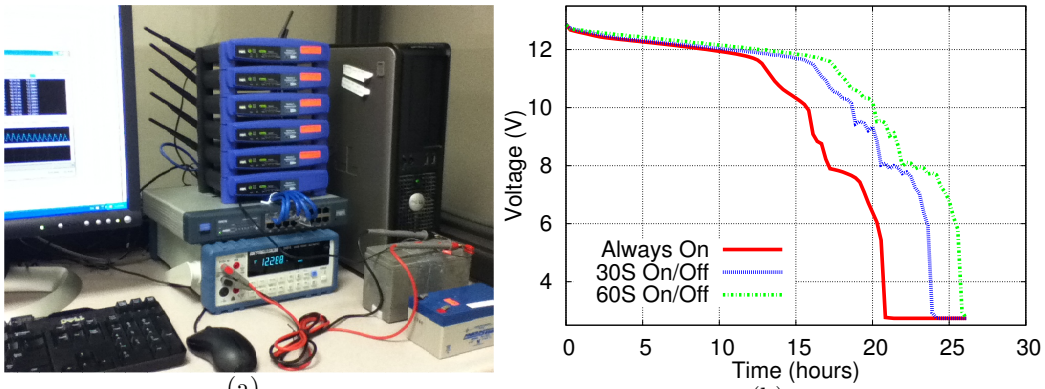
Duty-cycling, however, has adverse effects on the operation of intrusion detection systems, which are required to be on/awake at all times, to monitor network traffic. As proposed in the literature, in wireless networks, some nodes can be selected as "monitoring" nodes. It is obvious that duty-cycling mesh nodes are not suitable to be monitoring nodes, since they are not awake all the time. Consequently, the research challenge/problem we address in this section is how to reconcile energy efficient operation, which requires nodes to be asleep as much as possible, with an effective intrusion detection, which requires nodes to be awake, to monitor traffic. We define this problem as an optimization problem and propose centralized and distributed algorithms for solving it, algorithms that trade off communication and computation overhead for optimality of the solution.

6.1 Validation of Duty-Cycled Operation in WMN

DistressNet, being deployed in an environment where electric power is very limited (if at all available), needs to aggressively pursue energy efficient operation, including in the WMN. Unfortunately, no native procedure is included in IEEE 802.11s to allow mesh routers to work in power saving mode. Moreover, a power saving mode is not supported by current wireless routers available on the market. Consequently, we propose to use an application-layer controlled duty-cycling, as a means for saving energy on mesh routers.

We ran experiments involving Linksys WRT54GL wireless routers (we tested different OpenWrt firmware versions as well) powered by 12V-7Ah Power Sonic rechargeable lead acid batteries (as illustrated in Figure 6.1(a)) to investigate if duty-cycling affects connectivity between mesh routers and their clients and estimate an expected increase in the mesh router lifetime. A wireless client establishes an ssh session when the mesh router is initially turned on and starts a terminal appli-

cation. Then, the duty-cycling operation is initiated by turning the wireless interface of a mesh router on and off using "*iwconfig eth1 txpower on/off*", at different time intervals. When using duty-cycling the power consumption of a mesh router was reduced by 840mW (the current consumption drops from 250mA, to 180mA when the wireless interface is turned off). We have validated experimentally that the proposed duty-cycling does not close the ssh session - our terminal application continues to work despite the duty-cycled operation of the router.



(a) (b)

Figure 6.1: (a) Experimental setup. (b) Battery consumption for different on/off intervals.

Figure 6.1(b) depicts the battery lifetime when the mesh router has the wireless interface constantly on, and when it operates at a 50% duty-cycle, with different on/off periods (e.g., 30s on/off and 60s on/off). As expected, we observe that when the router operates in duty-cycle mode, its lifetime is extended. Surprisingly, different on/off periods (30s vs 60s) extend the lifetime of the router differently, despite operating at the same 50% duty-cycle. As shown in Figure 6.1(b) the router lifetime is prolonged by 5h when using the 60s on/off duty-cycling, and by 3h when using the 30s on/off interval. This experiment validated battery recovery effects [66], that

have been mentioned briefly in the context of WMN [57]. We used the data collected during these experiments to enhance a simulator we have developed so that it accounts for the new source of energy efficiency, namely battery recovery.

The proposed energy efficient operation based on duty-cycling, however, has adverse effects on solutions for monitoring network security in wireless networks. If a mesh router is assigned an intrusion detection/monitoring task or if it helps in relaying high network traffic, then the router has to be awake all the time. This implies that routers with higher available energy and with higher network traffic load should be better suited candidates for becoming monitoring nodes. Deciding which routers should be selected as monitoring nodes, for reducing total energy consumption, while not affecting intrusion detection functions is a challenging problem. In the sections that follow, we introduce our systems and security models and formulate mathematically our problem.

6.2 Problem Formulation and its NP-Hardness

We model a WMN as a graph $G = (V, E)$, in which V is the set of mesh nodes $\{v_1, v_2, \cdots, v_n\}$, and $E = \{e_1, e_2, \cdots, e_m\}$, is the set of links between them. We denote the residual energy and the network load of a mesh node v_i by b_i and l_i, respectively. Let $w : V \longrightarrow [0, 1]$ be a cost function that assigns a weight w_i to a node v_i based on l_i and b_i ($w_i = w(l_i, b_i)$), such that higher normalized l_i and b_i values result in lower weight being assigned to v_i.

Definition 8 *The* Covering Set $C_i = \{e_{ij} : j = 1..c\}$, $C_i \subseteq E$, *for a monitoring node v_i, contains any edge e_{ij} where either e_{ij} is incident to v_i or v_i is connected to the two end points of e_{ij}. (Figure 6.2).*

Considering our link coverage (as opposed to node coverage) and the desired effect of selecting mesh routers with higher residual energy and higher network load

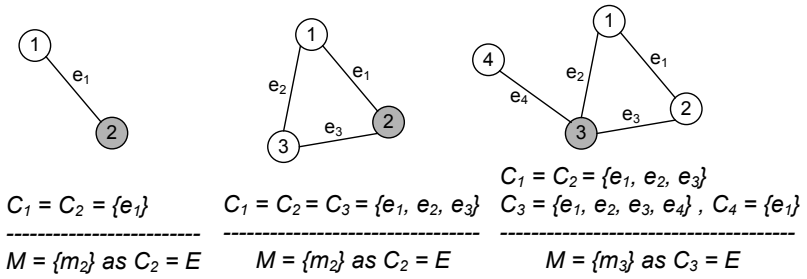


Figure 6.2: Examples of monitor nodes M and corresponding covering sets C.

as monitoring nodes, we define the Weighted Monitor Coverage (WMC) Problem as follows:

Definition 9 ***Weighted Monitor Coverage (WMC) Problem***

Given $G = (V, E)$ with a set of vertices in V and a set of edges in E, let w_i be the weight of v_i, find the set of monitors $M = \{m_1, m_2, \cdots, m_k\}$ with the minimum cost $\Sigma_{i \in M} w_i$, such that $\bigcup_{i \in M} C_i = E$, i.e., the monitors cover all edges in G and $b_i \geq b_{th}, \forall i \in M$, i.e., the residual energy of each monitor node exceeds a threshold b_{th}.

We set b_{th} based on real battery profile; however, if it is not possible to cover all the links by monitor nodes with residual energy higher than the threshold, the threshold value is reduced by Δb until there exists a feasible solution. It is important to observe that our problem is different than the Maximum Coverage and 1-hop Dominating Set problems as proposed in earlier research. Similarly, it may seem that our problem is the same as the Weighted Vertex Cover problem, since both problems aim to cover all the network links, while minimizing the total weight assigned to the selected mesh nodes. It is key to observe that in the Vertex Cover problem when we pick a vertex, incident edges to the vertex are considered covered. In our problem, however, all edges in the communication range of the node are considered to be covered. As an

illustration of these key observations, consider Figure 6.2, which depicts the covering sets and monitoring set of different networks. As shown, only one node is enough to monitor all the edges of a 3-node network.

Theorem 1 *WMC is NP-hard even for* $w_i = 1$.

Proof 1 *First we assume that each node has a unit weight, so that the problem is to find minimum number of nodes to cover all the edges, i.e., Monitor Coverage (MC) problem. We show that even with this assumption MC is NP-hard, thus same proof is valid for WMC. To prove this, we reduce the Set-Cover to MC in polynomial time. Given a universe* $U = \{x_1, x_2, \cdots, x_n\}$*, subsets* $S_i \subseteq U$*, and a positive integer* k*, the Set-Cover is to determine if* $\exists$ *a collection* C *of at most* k *such subsets such that union of the* k *subsets cover all of* U*, i.e.,* $\exists C \subseteq \{1, 2, \cdots, m\}$ *s.t.* $|C| \leq k$ *and* $\bigcup_{i \in C} S_i = U$*. Given the instance of the Set-Cover, we attempt to construct the instance of MC. We let* $E = U$*, and for each* $v_i \in V$*, define the subset* $C_i \subseteq E$ *such that* $C_i = \{e | e \text{ is within communication range of } v_i, e \in E\}$.

Next we show that our construction is correct, i.e., we prove the claim, "Set-Cover has a valid instance if and only if MC has a valid instance." Suppose Set-Cover has a valid instance. By our construction, each S_i *corresponds to* C_i*. Since* $|S_i| = k$*, we have at most* k *monitors. Furthermore, since* $\bigcup_{i=1,\cdots,k} S_i = U$*, and we defined* $E = U$*, the* k *monitors cover all the edges in* G*. Therefore, MC has a valid instance. Next suppose that MC has a valid instance. This implies that there exists at most* k *monitors in* G*. By our construction, each subset* C_i *of covered links by monitor* m_i *corresponds to the subset* S_i*, so* $|S_i|$ *is* k*. And since the monitors cover all edges in* G*, and* $E = U$*, it is trivial to see that* $\bigcup_{i=1,\cdots,k} S_i = U$*, thus proving the claim. This proof is also valid for the case that weights are more than one unit.* $\square$

One other problem to consider is how to optimally choose duty-cycle values for non-monitoring nodes, to extend WMN lifetime, but to also ensure WMN availability to clients. Obviously, the longer a mesh router sleeps, higher the lifetime extension will be. WMN availability, however, limits the maximum time interval a mesh router can sleep. Therefore, the actual duty-cycle a non-monitoring mesh router will use trades off network availability for WMN lifetime. In this research, we assign the duty-cycle value for a non-monitoring nodes inversely proportional its network load. We leave the computation of an optimal duty-cycle value for a mesh router, for future work.

6.3 Proposed Solutions

In this section, we present centralized and distributed solutions for our WMC problem. As centralized solutions, we propose a greedy algorithm and an integer linear programming (ILP) algorithm. These algorithms are executed on the WMN gateway (i.e., base station). The base station collects information from WMN nodes (i.e., connectivity, communicating load, and residual energy), executes the monitoring node selection algorithm (either greedy or ILP) and distributes back in the network the decisions. The distributed algorithm, however, is executed by individual nodes using 1-hop neighbor information. It is notable that these algorithms have different time complexity, message complexity, and approximation ratios.

6.3.1 Greedy Algorithm

We propose a greedy algorithm, shown in Algorithm 7. The algorithm selects monitor nodes based on the number of links per unit weight a node covers and based on the remaining energy level b_i which needs to be above a threshold b_{th}. When a node v_i is selected, all the links in C_i are covered. Hence, they are removed from the uncovered set E'. This selection is repeated until all the links become covered. The

Algorithm 7 Greedy Monitor Coverage

```
1: M = {}
2: E' = E , V' = V
3: while E' ≠ ∅ do
4:    if ({m} = max_{i∈V'}{|C_i ∩ E'|/w_i}) ≠ ∅ then
5:       M = M ∪ m
6:       V' = V' − m
7:       E' = E' − C_i
8:    else
9:       b_th = b_th − Δ_b
10:   end if
11: end while
```

proposed algorithm runs in time polynomial of $|E|$ and $|V|$. Similar to the Set Cover problem, the approximation ratio of our greedy algorithm is $H(max_{i \in V}|C_i|)$, where $H(n) = \Sigma_{j=1}^{n}(\frac{1}{j}) \leq \ln n + 1$.

6.3.2 Integer Linear Programming

The second solution we propose is based on Integer Linear Programming (ILP). Let S_j be a set of selected monitor nodes out of all possible nodes that can monitor link j. The proposed WMC can be formulated as follows:

$$\text{minimize} \quad \sum_{i \in V} w_i m_i \tag{6.1}$$

$$\text{subject to:} \quad |S_j| \geq 1, \qquad \forall j \in E \tag{6.2}$$

$$b_i \geq b_{th}, \qquad \forall m_i \in M \tag{6.3}$$

$$m_i \in \{0, 1\} \tag{6.4}$$

where constraint (6.2) indicates that every link has to be covered, constraint (6.3) enforces the algorithm to select the nodes with residual energy greater than a thresh-

old. We reduce b_{th} by Δb and run the ILP again if there is no feasible solution for the given b_{th}.

6.3.3 Distributed Algorithm

We propose a distributed algorithm, shown in Algorithm 8. In our protocol each node periodically broadcasts a HELLO message containing its residual energy, network traffic it handles and the number of links it covers, and sets a local timer T_{BC}. When T_{BC} fires, every node builds an adjacency table *AdjTbl* using the collected HELLO packets. Then each nodes computes the weight per link for each neighbor and for itself. Based on this computed value, a node v_i will broadcast an IS-MONITOR message to announce itself as monitor or it will set another timer T_{Mon}, waiting to receive an IS-MONITOR message from a neighbor. If node v_i receives an IS-MONITOR message before T_{Mon} expires, it checks all its links to see whether the elected monitor(s) can monitor all of them. If there are still uncovered links, then v_i will also broadcast IS-MONITOR to its neighbors, indicating it will be a monitor. To avoid redundancy, the higher the weight (w_i) of a node, the longer timer T_{Mon} will be.

6.3.4 Solution Analysis

The proposed algorithms have different time complexities, message complexities, and approximation ratios. The Set Cover problem has a relatively high approximation ratio (i.e., O(ln $|C_i|_{max}$)). Improving this ratio has not been addressed by research. Our greedy algorithm has the same approximation ratio as Set Cover, while the ILP solution is considered near optimal. The distributed algorithm, however, has worse approximation ratio because the solution is locally optimal. On the other hand, the time complexity of the distributed algorithm is O($|V|$), which is smaller than that of the centralized algorithms; greedy algorithm has time complexity

Algorithm 8 Distributed Monitor Coverage

1: Broadcast (HELLO)
2: delay (T_{BC})
3: create ($AdjTbl_i$)
4: **if** $b_i \geq b_{th}$ and $\frac{w_i}{|C_i|} > \frac{w_j}{|C_j|}$ for all $j \neq i$ **then**
5: $m_i = 1$
6: Broadcast (IS-MONITOR)
7: **else**
8: delay (T_{Mon}) //should receive IS-MONITOR
9: **if** ($\{e_l\}$ = uncover-link(i)) $\neq \emptyset$ and $b_i \geq b_j, \forall v_j$ that can cover e_l **then**
10: Broadcast (IS-MONITOR)
11: **else**
12: duty-cycle(l_i)
13: **end if**
14: **end if**

of order O($|V||E|min(|V|, |E|)$) and the time complexity of ILP algorithm depends on the solver. The message complexity of the distributed algorithm is less than that of the centralized algorithms, since the distributed algorithm requires $|V| + |M|$ network-wide packet exchanges. The message complexity of centralized algorithms is O($|V|log|V|$).

Considering the above analysis, we expect that centralized algorithms produce a smaller set of monitors than the distributed algorithm. On the other hand, the distributed algorithm, with lower time and message complexities, produces larger set of monitors with higher average weight. Therefore, we expect that centralized algorithms will save more energy than the distributed one. The distributed algorithm, however, will select more monitoring nodes, improving the intrusion detection rate.

6.4 Performance Evaluation

All algorithms presented in this section are implemented in MATLAB except for ILP solver which is already implemented in *bintprog* function of MATLAB. We consider networks ranging in size from 10 to 90 nodes for all of our simulations while

maintaining the network density constant at 3 neighbors per radio range. The radio range is fixed at 50m.

6.4.1 Link Coverage vs. Node Coverage

We first evaluate the network coverage and resource consumption of link coverage and node coverage approaches. We show that a solution for node coverage problem which guarantees 100% node coverage, does not necessarily guarantee 100% link coverage. Motivated by the fact that traditional node coverage approaches are not suitable for detecting attacks against wireless communication links, we are curious to investigate what percentage of network links are usually left uncovered by the node coverage approach. It is also important to know how many extra monitoring nodes a link coverage imposes to the monitoring mechanism when comparing to node coverage approaches.

To compare with node coverage solutions, we implemented a greedy Maximum Coverage algorithm [75, 83] (called "MAX Coverage" for the remainder of the section). To fairly compare the results, we ran MAX Coverage for several upper bounds (maximum number of monitors), and found the minimum upper bound (called k) that guarantees 100% node coverage in a 50-node network. As depicted in Figure 6.3(a), roughly 35% of nodes have to be selected for guaranteeing 100% node coverage. We use this upper bound in all of our simulations.

Figure 6.3(b) shows that the number of uncovered links increases as network size grows. In contrast, link coverage approach always guarantee full link coverage. Hence, using traditional node coverage approaches has the risk of not monitoring all communication links (i.e., false negative rate in intrusion detection systems) and this risk increases to more than 10 % of the links for $N = 90$. However, one may argue that link coverage approach requires more monitoring nodes to cover all communication

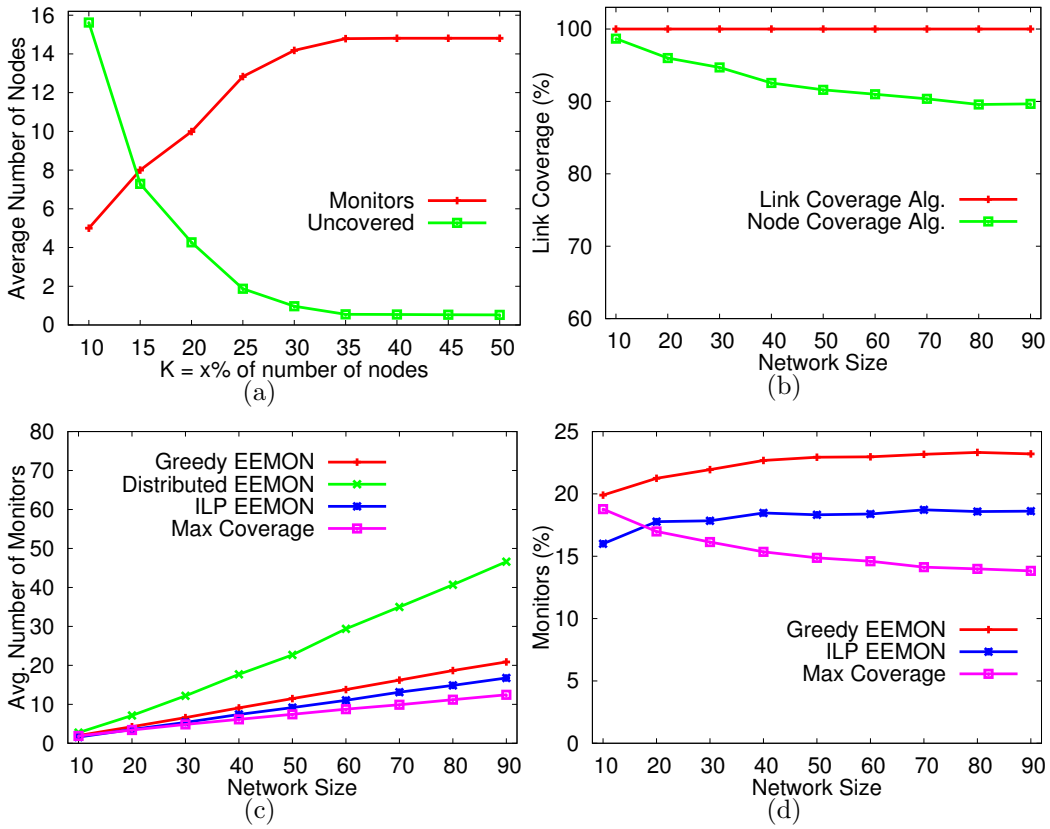


Figure 6.3: (a) Average number of monitor and uncovered nodes for different K values in Max Coverage of 50-node network. (b) Link coverage percent. (c) Average number of monitors in different algorithms. (d) Percentage of nodes selected as monitor in link coverage and node coverage approaches.

links. Figure 6.3(c) shows the average number of nodes selected as monitoring nodes in different EEMON algorithms and also in MAX Coverage algorithm. These results were obtained from 500 random networks for each given network size. As expected, Distributed EEMON has the worst results among other algorithms and requires a large number of monitoring nodes. However, the centralized algorithms of EEMON select only few monitoring nodes to cover all communication links.

To have a better comparison between Greedy EEMON and ILP EEMON with traditional node coverage approaches, Figure 6.3(d) depicts the percentage of the

nodes selected as monitoring nodes in different network sizes. As shown, for 10-node WMN, ILP EEMON achieves 100% link coverage with less monitoring nodes than Max Coverage algorithm. For mid-size networks (e.g., 20 to 50 nodes as the most common sizes in WMN [87]), ILP EEMON requires 1∼3% extra monitoring nodes than Max Coverage to achieve 100% link coverage (5∼9% greater than Max Coverage). Finally, for larger WMN, 60 to 90 nodes, ILP EEMON requires less than 5% extra monitoring nodes than Max Coverage to cover all communication links (i.e., 10% improvements in link coverage rate). Figure 6.3(d) also shows that Greedy EEMON requires less than 10% extra monitoring nodes than MAX Coverage to achieve 100% link coverage. Hence, it is quite reasonable to have few more monitoring nodes than traditional solutions for covering all communication links and consequently increasing the intrusion detection rate.

6.4.2 Comparison of Different EEMON Algorithms

In this section, we evaluate the performance of all EEMON algorithms used for link coverage and show how different algorithms select optimal sets of monitoring nodes with maximum residual battery charge and communication load (i.e., to keep the average cost per monitoring nodes as low as possible). All experiments in this section are performed for 500 random networks for each given network size. The average residual charge and communication load among all nodes in each network is always 50%. The metrics we evaluate for each algorithm are the average residual charge among monitoring and non-monitoring nodes, the average communication load among monitoring and non-monitoring nodes, the threshold reduction in B_{th} required for covering all communication links, and finally, the time complexity of monitoring node selection algorithms. According to the WMC formulation, we expect EEMON algorithms to select nodes with higher residual charge and also higher

communication load as monitors. It is also worth mentioning that traditional node coverage solutions [46, 47, 75, 83, 98] do not necessarily consider both residual charge and communication load together in their optimal monitoring formulation, as we do for Max Coverage.

Figures 6.4(a) and 6.4(b) depict the average residual charge among selected nodes, i.e.,$(1/|M|)\Sigma_{v_j \in M} b_j$, and unselected nodes, i.e., $(1/N - |M|)\Sigma_{v_j \notin M} b_j$, respectively, in all EEMON algorithms and the Max Coverage algorithm. Considering that the average residual charge among all nodes is always 50%, the results shown in these two figures confirm that the proposed algorithms always select monitoring nodes among those with higher residual charge. As depicted in Figure 6.4(b), the set of non-monitor nodes includes nodes with low residual charge.

Greedy EEMON and Distributed EEMON select nodes with minimum weight per links first (higher residual energy and communication load). Therefore, selected nodes in the last iterations are most likely among low-battery nodes since we may have to select nodes that cover a single wireless link; simply because all the links must be covered. When considering Figures 6.4(a) and 6.4(b) together, one can observe that distributed EEMON has the lowest average residual charge among selected nodes because 1) distributed solution is locally optimal, and 2) as network size increases more monitoring nodes are required (see Figure 6.3(c)). This increase in monitoring nodes results in selecting more monitors among low-battery nodes. In contrast, Max Coverage algorithm has the highest average among all since it does not need to cover all links, and thus, fewer monitoring nodes are always required. In conclusion, the more nodes selected as monitors by an algorithm, the higher is the chance of selecting low-battery nodes. It is worth mentioning that the higher residual charge of monitors in Greedy EEMON, compared to the ILP EEMON, forces the selection of more monitoring nodes as previously shown in Figure 6.3(c).

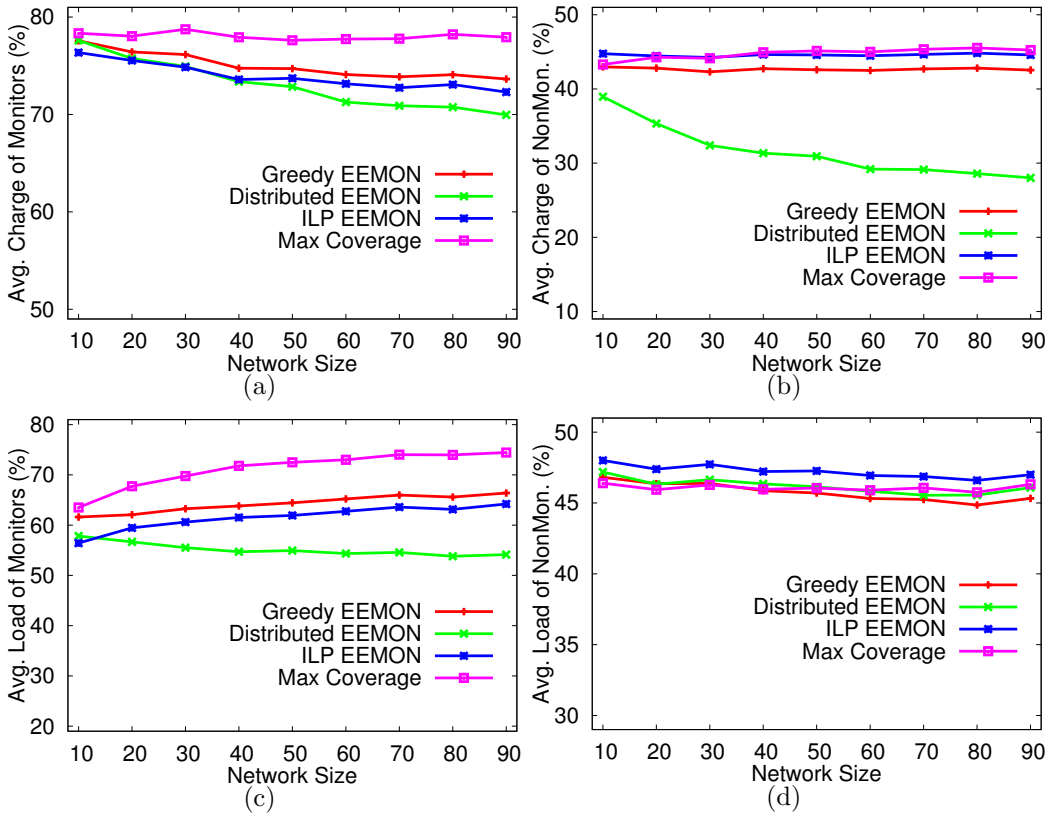


Figure 6.4: Average residual battery charge of (a) Nodes selected as monitoring nodes. (b) Nodes NOT selected as monitoring nodes. Average communication load of (c) nodes selected as monitoring nodes. (d) Nodes NOT selected as monitoring nodes.

Figures 6.4(c) and 6.4(d) depict the average communication load of monitoring nodes, i.e., $(1/|M|)\Sigma_{v_j \in M} l_j$, and non-monitoring nodes, i.e., $(1/N - |M|)\Sigma_{v_j \notin M} l_j$, respectively, an evidence that the proposed algorithms select monitors with higher values of communication load. As one can observe, the average communication load among selected nodes is always above 50% (the average among all) while it is below 50% for non-monitoring nodes. However, the results average communication load, especially for Distributed EEMON, are much better than those for average residual charge. This is because there is no exception in selecting nodes with lower

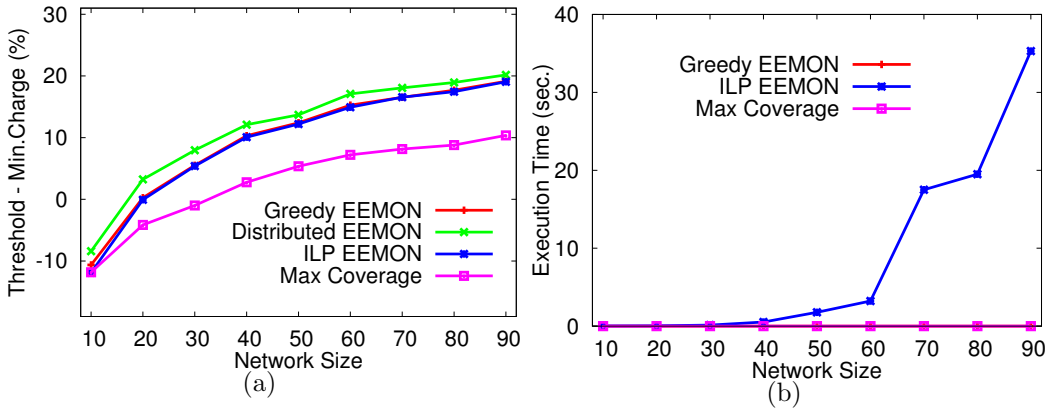


Figure 6.5: (a) $B_{th}-$Minimum Charge among selected nodes. (b) Average Execution Time.

communication loads, as is the case for residual charge (i.e., Δb).

Next, we show how different EEMON Algorithms require different Δb for selecting monitor nodes. Using the battery profiling data, we set a threshold of b_{th} =60%, slightly higher than the average among all nodes, for the residual energy of a node to be a monitoring node candidate. The reduction in the residual energy threshold is considered as penalty by EEMON algorithms since monitoring nodes with low residual energy most likely die in a short time. The reduction in the residual energy threshold (i.e., Δb) is shown in Figure 6.5(a). Negative values for Δb mean that the minimum residual energy among selected nodes is larger than b_{th} and no threshold reduction is required. As shown, larger networks are penalized more than smaller ones because more nodes are required to be selected as monitors. This increases the probability of choosing low-battery nodes and increasing the Δb. We can also see that Distributed EEMON is penalized more than Greedy and ILP EEMON. As expected, Max Coverage solution (that was shown to leave a considerable percentage of links uncovered) causes lower Δb than EEMON.

Finally, we compare the execution time of Greedy EEMON, ILP EEMON, and

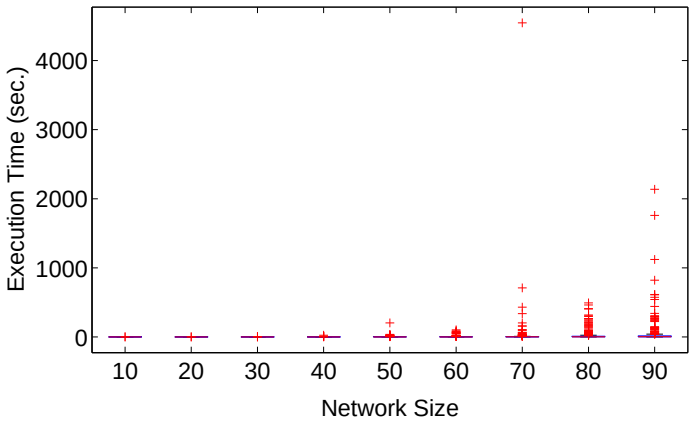


Figure 6.6: The distribution of ILP EEMON execution time.

Max Coverage. Figure 6.5(b) shows the average execution time of these algorithms for 500 random networks of each given network size. As depicted, Greedy EEMON and Max Coverage produce results in a few milliseconds while the execution time for ILP EEMON is in the order of a few seconds (e.g., 35 seconds for 90-node networks). We plot the distribution of the ILP execution time as a function of network size for all 500 random networks in Figure 6.6. As depicted, the higher average for ILP execution time is because of few outliers (e.g., one topology among 500 random 70-node networks) that rarely occur. Hence, considering the optimal solutions that ILP EEMON produces, an efficient approach for security administrators might be to set a time threshold for ILP solver and choose the greedy solution if the ILP exceeds the threshold.

6.4.3 Impact of Duty-Cycling on WMN Lifetime

We investigate the impact of duty-cycling on WMN lifetime through a system implementation on five Linksys WRT54GL routers. One router acts as an AC powered gateway. The other mesh routers are battery powered. We assigned a fixed random network load to each router as 62%, 49%, 33%, 67% of the maximum network load

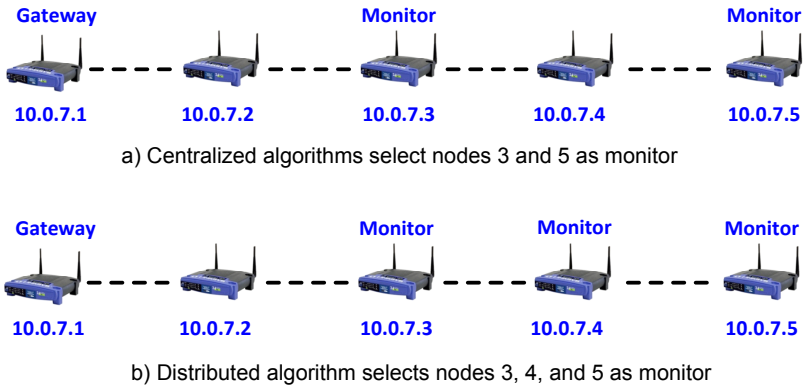


Figure 6.7: Five-node mesh network topology and different monitoring solutions.

a mesh router can handle. As depicted in Figure 6.4.2, we created a linear network topology to ensure that centralized and distributed algorithms produce different set of monitoring nodes. The Centralized algorithms (Greedy WMC and ILP WMC) selected nodes 3 and 5 as monitoring nodes, while the distributed algorithm selected nodes 3, 4 and 5 as monitoring nodes. We used 12V-3.4Ah Power Sonic rechargeable batteries for powering the mesh routers. We observed that the centralized solution prolonged the network lifetime (defined as the time when the first battery dies) by 8%, while the distributed solution did not increase it. The explanation for this is that battery attached to the router 4, that was the first one died, was not monitor in centralized solution, however, in the distributed solution it was selected as monitor.

7. RESOURCEFUL AND TRAFFIC-AWARE IDS

This section investigates the effect of traffic-awareness on the performance of intrusion detection systems proposed for battery-powered resourceful WMN. In fact, we will show how traffic-awareness, as an administrative assumption for some WMN applications, will reduce the amount of resources (e.g., energy) consumed by performing IDS on WMN nodes. We introduce TRAIN, as a resourceful and traffic-aware IDS that uses link-coverage mechanism as used in EEMON. Because of the similarities between EEMON and TRAIN detection systems, the performance of TRAIN will be compared to EEMON.

7.1 Problem Formulation

We model a WMN backbone (i.e., only WMN nodes and backbone links) as a graph $G = (V, E)$, in which V is the set of mesh nodes $\{v_1, v_2, \cdots, v_N\}$, and $E = \{e_1, e_2, \cdots, e_Q\}$ is the set of links between them. Considering the information about active traffic paths in the WMN, we denote the number of nodes and links actively participating in WMN routing paths by n ($n \leq N$) and q ($q \leq Q$), respectively. Thus, the set of active nodes and links in WMN, after removing *idle* nodes/links, is modeled as a reduced graph $G' = \{V', E'\}$, where V' is the set of n active nodes ($V' \subseteq V$), and E' is the set of q active links ($E' \subseteq E$). The set of selected monitoring nodes are denoted by $M = \{m_j \,|\, m_j \text{ is a monitoring node}\}$. We also denote the set of routing paths for the network traffic by $P = \{p_1, p_2, \cdots, p_l\}$, where $P_i^v = \{v_j \,|\, v_j \text{ is located on } p_i\}$ and $P_i^v \subseteq V'$. We also use $P_i^e = \{e_r \,|\, e_r \text{ is a link in } p_i\}$ for $P_i^e \subseteq E'$ to show the set of edges (links) on each path.

Additionally, we denote the residual energy and the communication load of a mesh node by b_j and l_j, respectively. Based on the maximum residual charge and

communication load a node can have, both b_j and l_j are considered normalized values in range $[0, 1]$. Let $w : V \longrightarrow [0, 1]$ be a cost function that assigns a weight w_j to a node v_j based on l_j and b_j ($w_i = w(l_j, b_j) = 1/(l_j \times b_j)$), such that higher normalized l_j and b_j values result in lower weight being assigned to v_j. Finally, we use vectors $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_N]$ (*i.e., Battery Threshold*) to represent the minimum energy charge required for being selected as monitoring node. This threshold is important parameters and typically set by a network administrator based on energy resources.

7.1.1 Optimal Monitoring Problem

In this section, inspired by EEMON's problem formulation, we present optimal monitoring problem statement in resourceful and traffic-aware IDS class.

Definition 10 *The **Path Covering Set** is a set of paths in WMN that a node can monitor, i.e., at least one link on the path is in the Covering Set of that node ($PC_j = \{p_i \,|\, \exists e_r \in P_i^e$ such that $e_r \in C_j\}$).*

This definition shows the difference between PRIDE (node coverage approach) and TRAIN (link coverage approach) in their traffic monitoring mechanisms. TRAIN uses *monitoring coverage* to benefit from every possible node in the network that can contribute in monitoring a path (as opposed to PRIDE that only uses those located on the path).

Definition 11 ***Path Monitoring Problem (PMP)*** *Given $G = (V, E')$ with a set of vertices in V and a set of edges in E' and a set of paths P in WMN, let w_j be the weight of v_j, find the set of monitors $M = \{m_1, m_2, \cdots, m_k\}$ with the minimum cost $\Sigma_{j \in M} w_j$, such that $\bigcup_{j \in M} PC_j = P$, i.e., the monitors cover all paths in G, and $b_j \geq \beta_j, \forall j \in M$, i.e., the residual energy of each monitoring node exceeds a battery threshold.*

In order to formulate PMP as an ILP, very similar to WMC, let S'_i be a set of selected monitor nodes out of all possible nodes that can monitor *path* p_i. We formulate PMP Problem as follows:

$$\text{minimize} \quad \sum_{v_j \in V} w_j m_j \tag{7.1}$$

$$\text{subject to:} \quad |S'_i| \geq 1 \quad , \forall p_i \in P \tag{7.2}$$

$$b_j \geq \beta_j \, (or \, b_{th}) \quad , \forall m_j \in M \tag{7.3}$$

$$m_j \in \{0, 1\} \tag{7.4}$$

where constraint (7.2) indicates that every path has to be covered; constraint (7.3) enforces the algorithm to select the nodes with residual energy greater than a threshold.

EEMON/TRAIN and PRIDE formulations are compared in Table 7.1. Let consider these pseudo-formulations as representatives for optimal monitoring problem in intrusion detection systems of infrastructureless wireless networks. Such a consideration is because a variety of system and security assumptions (generally placed in two resourceful and resourceless categories) are already encompassed in these two examples: 1) the amount of *resources* (i.e., processing, storage) and the energy available for monitoring activity; 2) the network *component* (i.e., link, node, path) that has to be monitored.

As shown in Table 7.1 for EEMON and TRAIN, when nodes can fully monitor a network *component* with all IDS functions (i.e., monitoring nodes), the objective functions is to **minimize** the number of them. This considers wireless device as a resourceful hardware, e.g., Meshlium [53], that can act as an IDS node detecting

Table 7.1: Objective functions and constraints in all formulations.

	PRIDE	EEMON/TRAIN
Objective	cover **paths** with *maximum* detecting modules	cover **links/paths** with *minimum* monitoring nodes
Constraints	every **path** has to be covered	every **link/path** has to be covered
	one detecting module per **path**	one monitoring node per **link/path**
	Resource Awareness: available memory/processing/energy resources for IDS is limited	

every possible attack. On the other hand, the objective function in PRIDE, that assumes resourceless wireless devices, is to **maximize** the number of IDS functions activated for monitoring a network *component*. Other than the difference in the objective function of resourceless and resourceful IDS classes, all of the constraints in the optimal monitoring problem are similar, as shown in Table 7.1.

7.2 TRAIN Protocol

TRAIN uses a centralized approach in which, similar to EEMON, the base station can either apply an ILP solver or use a greedy algorithm to solve PMP. The base station collects information from WMN nodes (i.e., connectivity, traffic information, communication load, and residual energy), executes the monitoring node selection algorithm (either greedy or ILP solver) and distributes the monitoring roles to the network.

Lessons learned from PRIDE show that a centralized approach can impose very large time complexity when dealing with large networks. Thus, inspired by PRIDE protocol, we consider a time-efficient approach for the centralized algorithms. In order to reduce the execution time, after collecting the local information from the nodes, the base station removes *idle* nodes from the network, i.e., those not contributing in the traffic routing, and optimally selects monitoring nodes that can cover all traffic paths. If the reduced graph is disconnected, each graph component is considered as a sub-problem and solved separately. Algorithm 9 presents TRAIN

Algorithm 9 TRAIN Protocol

1: $Data\_Collection(V, E, N, Q)$
2: $Relaxation(V', E', n, q)$
3: $Path\_Extract(V', E', P)$
4: $\mathcal{P} = P$
5: $g = 0$
6: **while** $\exists\, p_i \in \mathcal{P}$ **do**
7: $\quad g++$
8: $\quad U_g = \{p_i\}$
9: $\quad \mathcal{P} = \mathcal{P} \backslash \{p_i\}$
10: $\quad$ **while** $\exists\, p_j \in \mathcal{P}$ and
11: $\qquad \bigcup_{p_k \in U_g} (P_j^e \cap P_k^e) \neq \varnothing$ **do**
12: $\qquad U_g = U_g \cup \{p_j\}$
13: $\qquad \mathcal{P} = \mathcal{P} \backslash \{p_j\}$
14: $\quad$ **end while**
15: **end while**
16: **for** $\forall\, U_g$ **do**
17: $\quad V_g = \{v_j | v_j \in P_i^v \; and \; p_i \in U_g\}$
18: $\quad E_g = \{e_r | e_r \in P_i^e \; and \; p_i \in U_g\}$
19: **for** $\forall\, g$ **do** Solve-PMP(V_g, E_g, P)
20: $M = \bigcup_1^g M_g$

protocol.

Given the set of nodes, the algorithm first collects information from nodes and then produces the reduced sets V' and E' by removing idle nodes and links (Lines 1, 2). Next, the set of active routing paths P is extracted in Line 3. Given P, Algorithm 9 creates the set $\mathcal{P}$ of unvisited paths (Line 4) and then defines variable g as the number of sub-problems (Line 5). For every unvisited path p_i in set $\mathcal{P}$ (Line 6), Algorithm 9 first creates a new sub-problem U_g (Lines 7, 8) and marks it as a visited path (Line 9). Algorithm 9 then searches $\mathcal{P}$ to find any unvisited path p_j which is *connected* (two paths are *connected* if they have links in common) to at least one path in the current U_g (Lines 10, 11). If so, the corresponding path p_j will be added to the current sub-problem U_g and removed from $\mathcal{P}$ (Line 12, 13). When

no more paths in $\mathcal{P}$ can be added to the current U_g, Algorithm 9 increases g and creates a new sub-problem. This process repeats until there is no unvisited path in $\mathcal{P}$. Next, for every sub-problem U_g, the Algorithm creates the corresponding sets V_g and E_g as the set of nodes located on the paths and the set of links on the paths of component U_g, respectively (Lines 16, 17, 18). Finally, Algorithm 9 solves PMP (running either an ILP solver or the greedy algorithm on the nodes, links, and paths of each sub-problem U_g) in Line 19. The set of monitoring nodes M will be the union of all set of monitoring nodes selected for each graph component, i.e., M_g (Line 20).

Greedy TRAIN is shown in Algorithm 10. This Algorithm first creates an empty set of monitoring nodes for the corresponding graph components (Line 1) and then puts the set of all paths of that component in P_g (Line 2). Next, Algorithm 10 selects monitoring nodes based on the number of paths per unit weight which a node covers and based on the remaining energy level b_i that must be above threshold b_{th} (Lines 3, 4). When a node v_j is selected as a monitoring node (Lines 5, 6), all the paths in PC_j are covered. Hence, they are removed from the uncovered set of paths P_g (Line 7). This selection is repeated until all paths become covered. Similar to Greedy EEMON, Algorithm 10 may reduce the threshold by Δb to ensure that all paths are covered (Line 9). The proposed Algorithm runs in time polynomial of $|P_g|$ and $|V_g|$. The approximation ratio of Greedy TRAIN is $H(max_{v_j \in V_g}|PC_j|)$.

7.3 Simulation Results

We performed a thorough set of simulations to evaluate the performance of TRAIN in selecting monitoring nodes with respect to the network energy consumption and intrusion detection rates. The simulation results are presented in four sections: 1) evaluation of different TRAIN algorithms for complexity versus optimality; 2) comparison between the energy consumption of EEMON and TRAIN solutions;

Algorithm 10 Greedy TRAIN

1: $M_g = \{\}$
2: $P_g = \{p_i \,|\, \forall e_r \in P_i^e,\, e_r \in E_g\}$
3: **while** $P_g \neq \varnothing$ **do**
4: **if** $(\{m\} = max_{v_j \in V_g}\{|PC_j \cap P_g|/w_j\}) \neq \varnothing$ **then**
5: $M_g = M_g \cup m$
6: $V_g = V_g \backslash m$
7: $P_g = P_g - PC_j$
8: **else**
9: $b_{th} = b_{th} - \Delta_b$
10: **end if**
11: **end while**

and 3) evaluation of intrusion detection rates for all EEMON and TRAIN solutions and the traditional node coverage mechanisms. Similar to EEMON, TRAIN is also implemented in MATLAB.

7.3.1 Performance Evaluation of TRAIN Algorithms

In this section, we evaluate the performance of Greedy TRAIN and ILP TRAIN for solving the PMP problem. Given a random network topology (repeated for sizes of 10 to 90 nodes), we produce random communication paths and show how different TRAIN algorithms solve PMP and select monitoring nodes. Since the number of traffic paths can affect the number of required monitors, we consider three different settings for $|P|$ in this section: 1) $|P| = 0.1 \times N$ (i.e., 10% of network size); 2) $|P| = 0.3 \times N$; and 3) $|P| = 0.5 \times N$. The path length for each given network size is set to $\lfloor \sqrt{N} \rfloor$ and is constant for all different values of $|P|$.

Figure 7.1(a) shows the number of selected nodes as monitoring nodes for different number of paths in each given network size. These results are obtained from applying both Greedy and ILP TRAIN algorithms to 500 random networks (the same case studies used for evaluating EEMON solution). As depicted, the number of monitoring nodes increases as the network size or number of paths increase. However, the number

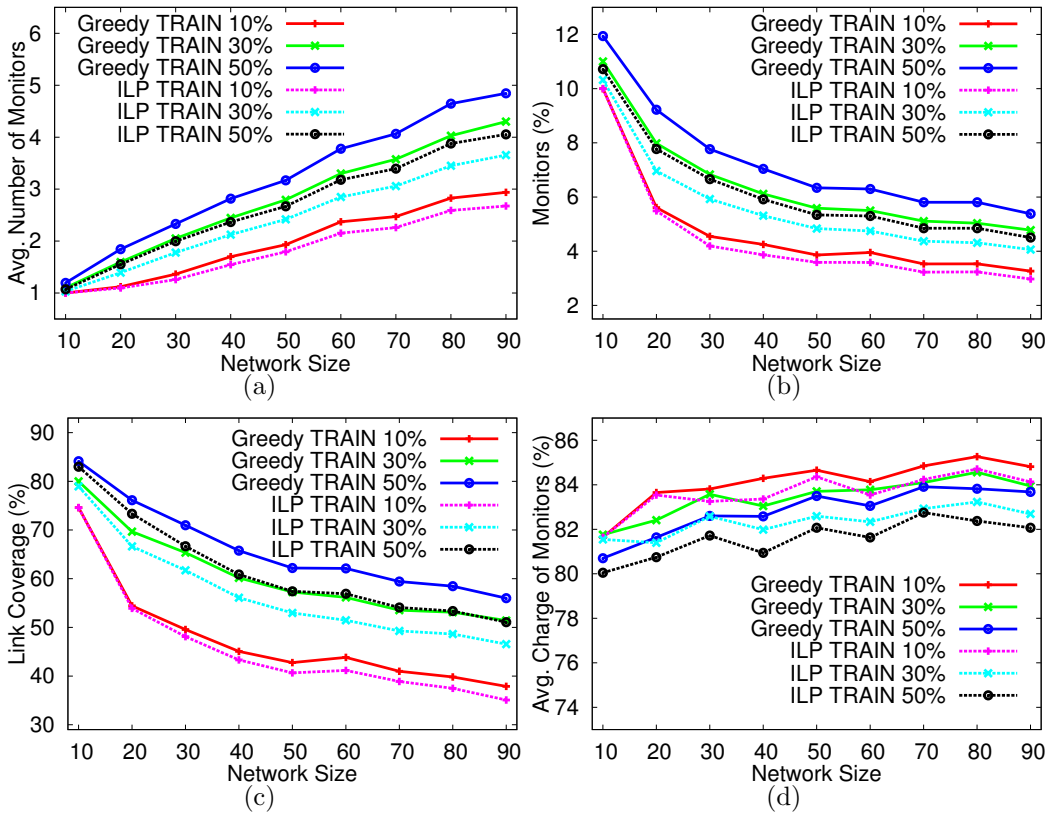


Figure 7.1: (a) Average number of monitors. (b) Percentage of nodes selected as monitors. (c) Average link coverage in the entire network. (d) Average battery charge of selected nodes.

of monitors in TRAIN (a traffic-aware solution) is considerably less than in EEMON (as EEMON has to cover all communication links). For example, given 90-node networks, Greedy TRAIN selects about 5 monitoring nodes on average to cover 45 paths (i.e., $|P| = 0.5 \times 90$) while ILP and Greedy EEMON select about 50 monitoring nodes. We emphasize here the smaller number of monitoring nodes selected by ILP TRAIN when compared to Greedy TRAIN.

Figure 7.1(b) shows the percentage of nodes selected as monitoring nodes for each network size and number of paths. As depicted, the number of monitoring nodes is usually less than 10% of the nodes in all of our considered settings. It is worth

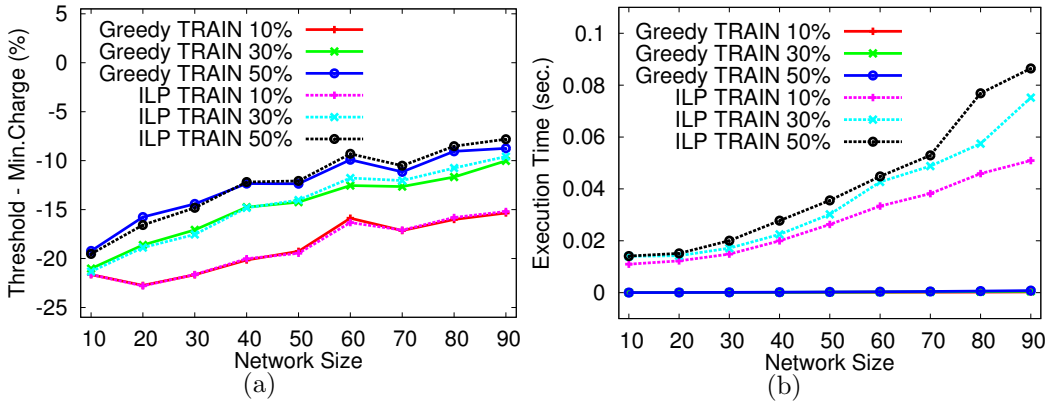


Figure 7.2: (a) $B_{th}-$Minimum Charge among selected nodes. (b) Average Execution Time.

mentioning that unlike EEMON solution, the percentage of selected nodes in TRAIN decreases as network size increases because of the number of paths we considered for each network size. This also has a negative effect on the link coverage percentage as shown in Figure 7.1(c). We note here that TRAIN, as proposed for traffic-aware networks, does not aim at covering every single network link but every path with traffic. Hence, when comparing TRAIN with EEMON, TRAIN is expected to have fewer nodes selected as monitoring nodes and more uncovered links. Figure 7.1(d) shows the average residual charge among selected nodes in TRAIN. As depicted, the average charge among monitoring nodes in TRAIN is slightly above those in EEMON because TRAIN requires less monitoring nodes and thtey are selected from those with very high residual charge.

The last two metrics we evaluate for TRAIN in this section are the threshold reduction in residual charge and the execution time. Figure 7.2(a) shows the threshold reduction in all TRAIN settings. As shown, Δb is always negative, which means Greedy and ILP TRAIN do not select nodes with the residual charge below b_{th}. For the same reason explained for Figure 7.1(d), the negative value of Δb in TRAIN

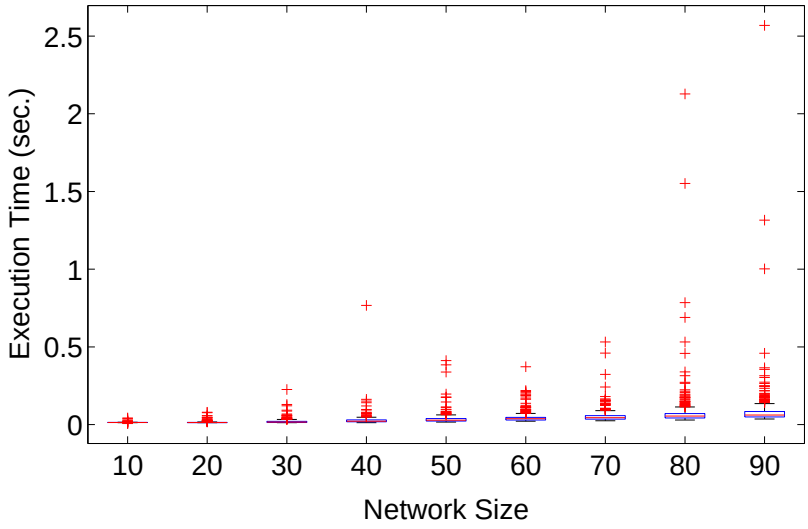


Figure 7.3: The distribution of ILP TRAIN execution time for $|P| = 0.5 \times N$.

is because it requires fewer monitoring nodes than EEMON to cover traffic paths. Hence, the probability of choosing a low-battery node as a path monitor is very low.

Figure 7.2(b) shows the average execution time of different TRAIN algorithms for different path settings and network sizes. As depicted, the results are always produced in less than 0.1 second, which is very fast. Similar to the execution time for ILP EEMON, we also plot the distribution of the execution time for ILP TRAIN ($|P| = 0.5 \times N$) as a function of network size. As shown in Figure 7.3, the execution time is always less than 3 seconds and that means ILP TRAIN, unlike ILP EEMON, is a practical approach to obtain the optimal set of monitoring node for large networks. We observe that even the execution time of the outliers is less than 3 seconds.

7.3.2 Energy Consumption of EEMON vs. TRAIN

In this section, we evaluate the energy consumption of EEMON and TRAIN when using duty-cycling for non-monitoring nodes. As we explained earlier, the network lifetime of a battery-powered WMN can be extended through duty-cycling. Thus,

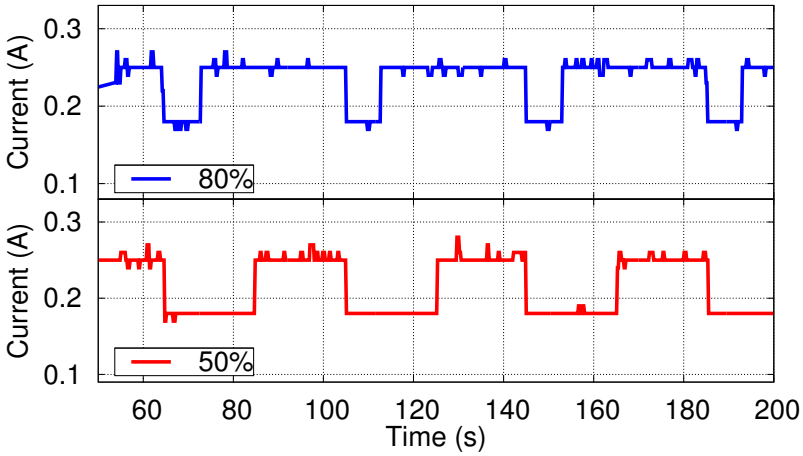


Figure 7.4: Duty cycling at ratios of 50% and 80% uptime in a 40s period.

considering the power that each monitoring node consumes and also the number of monitoring nodes selected for each given network, we evaluate the average energy consumption of our proposed algorithms during network lifetime.

The current consumption of Linksys routers is 250mA, which means each router consumes 3 Watts (12V×250mA). Thus, the regular energy consumed by each device during one minute working time (i.e., an epoch in our experiment) is 180 Joule. Given two different duty-cycling settings shown in Figure 7.4, the energy consumption in an epoch can be reduced to 154.8 J and 169.92 J for 50% and 80% duty-cycling settings, respectively. However, other duty-cycling intervals can also be considered by network administrator. If all nodes are selected as monitoring nodes and have to stay ON for an epoch, the average energy consumption of the nodes in that epoch is 180 J, i.e., the maximum energy consumption and consequently the minimum network lifetime. On the other hand, if there is no network traffic to be monitored (no monitor is required) and node can perform duty-cycling, e.g., 50%, the average energy consumption reduces, e.g., by 25.2 J for 50% duty-cycling, and the network lifetime is extended. Considering the two aforementioned intervals and the hardware

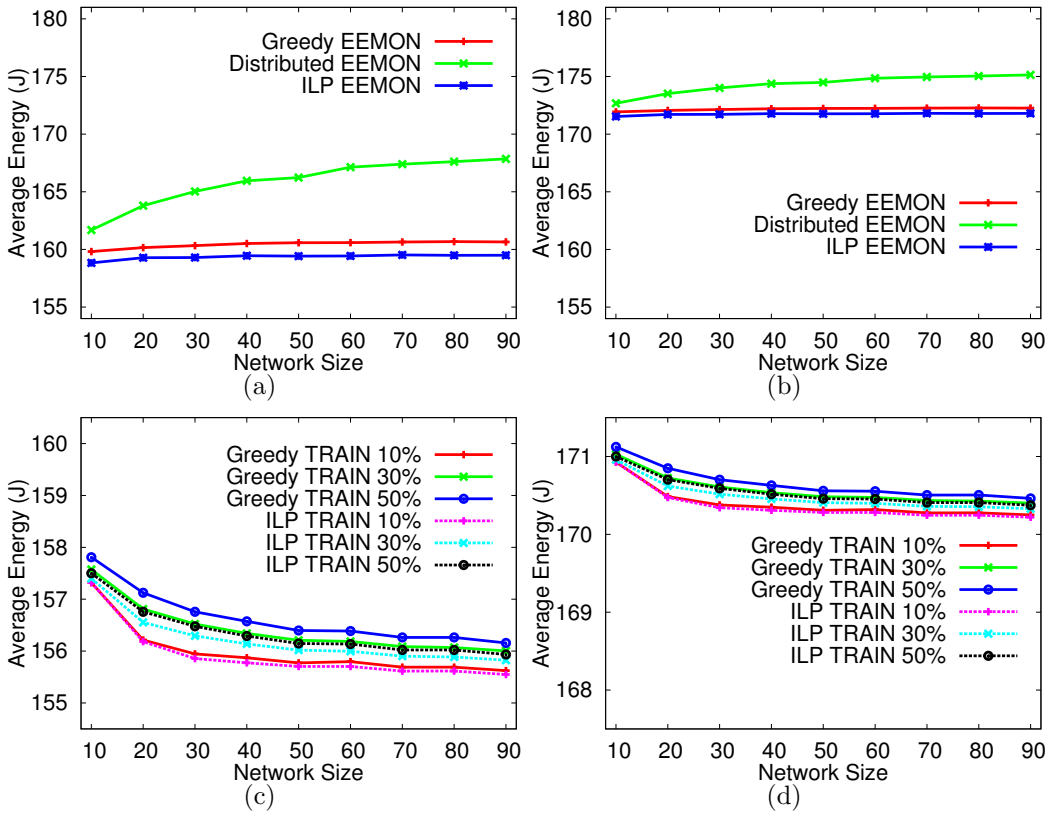


Figure 7.5: EEMON Energy Consumption (a) duty cycle 50%. (b) duty cycle 80%. TRAIN Energy Consumption (c) duty cycle 50%. (d) duty cycle 80%.

specifications, the average energy consumption in an epoch for a given network and given monitoring node mechanism (e.g., EEMON or TRAIN) varies between 154.8 J and 180 J. The larger the number of monitoring nodes, the larger the average energy consumption is.

Figure 7.5 shows the average energy consumption per node during an epoch (one minute working time) as a function of the network size. Figures 7.5(a), 7.5(b), show the average energy consumption of different EEMON algorithms for 50% and 80% duty-cycles, respectively. Figures 7.5(c), and 7.5(d), show the average energy consumption of different TRAIN algorithms and paths for 50% and 80% duty-cycles, respectively. The results show that those monitoring mechanisms and network set-

tings that select more monitoring nodes (e.g., Distributed EEMON) have higher energy consumption rates. Obviously, 80% duty-cycling consumes more energy than 50% duty-cycling when applying to the same networks and monitoring mechanisms.

We also emphasize here the possibility of having a small error in the amount of energy calculated for the given epoch. This is because the current consumption, as shown in Figure 7.4, slightly fluctuates, especially when the wireless interface is ON (i.e., the 250mA level). Thus, calculating the energy consumption for a longer period of time or even predicting the networks lifetime (based on the energy consumption measured for a small epoch) is less accurate than that for the epoch and also depends on the behavior of the batteries. Therefore, we do not evaluate the network lifetime through simulation.

7.3.3 Security Analysis

In this section, we evaluate the intrusion detection rates of all aforementioned monitoring mechanisms for the attack scenarios discussed in system and security models. We also consider some unexpected scenarios, e.g., single-hop attacks for TRAIN, to see if they can be detected even though they are not initially considered by the proposed mechanisms.

Let $Path_{ij}$ be the path between routers v_i and v_j. Also let $E'_i = \{e'_{ir} | e'_{ir}$ connecting node i to its client $r\}$ be a set of all local links between a router and its clients. Table 7.2 summarizes the paths for a client/host p (connected to router v_i) launching an attack against a single-hop or multi-hop target (either a router or a host). Let v_j be a multi-hop router and q be a client/host connected to router v_j. We will use this notation to explain and evaluate attacks we consider in this section.

Since in EEMON and TRAIN every traffic path is covered by at least one monitoring node, they ensure full coverage for any path that includes $Path_{ij}$, e.g., multi-hop

Table 7.2: Different attack scenarios and the corresponding attack paths.

Attack	**Target**	**Path**
Single-hop	router	$\{e'_{pi}\}$
	client/host	$\{e'_{pi}\} \cup \{e'_{iq}\}$
Multi-hop	router	$\{e'_{pi}\} \cup Path_{ij}$
	client/host	$\{e'_{pi}\} \cup Path_{ij} \cup \{e'_{jq}\}$

attacks in Table 7.2. We note here that $Path_{ij}$ in TRAIN has to be a path in the reduced graph G'. On the other hand, local links e'_{ir} for any client r, are only monitored by router v_i. Thus, detectability of the attacks on such local links depends on the role of the local router (monitor or non-monitor). For example, single-hop attacks in Table 7.2 are detectable if router v_i is a monitoring node.

To evaluate the intrusion detection rates of EEMON and TRAIN and compare them with Max Coverage, we simulated several attack scenarios. In order to evaluate each specific attack scenario, we launch $10 \times N$ random attacks for each given network size N. The attack scenarios we consider in this section include insider and outsider attackers, single-hop and multi-hop targets, severe and normal attacks, and also link-based and host-based attacks.

7.3.3.1 Normal Attacks

Normal attacks include any attack considered in the low severity group of attacks by the security administrator. Since the IDS running on every router (monitor and non-monitor) is able to detect attacks in this category, the administrator may put the most frequent attacks (e.g., port scan) reported in the WMN network in this category. This increases the chance of detecting most of the attacks launched by the attacker who is not aware of the attack categorization performed by the administrator. Moreover, there is no need to consider the attacks against unavailable services in

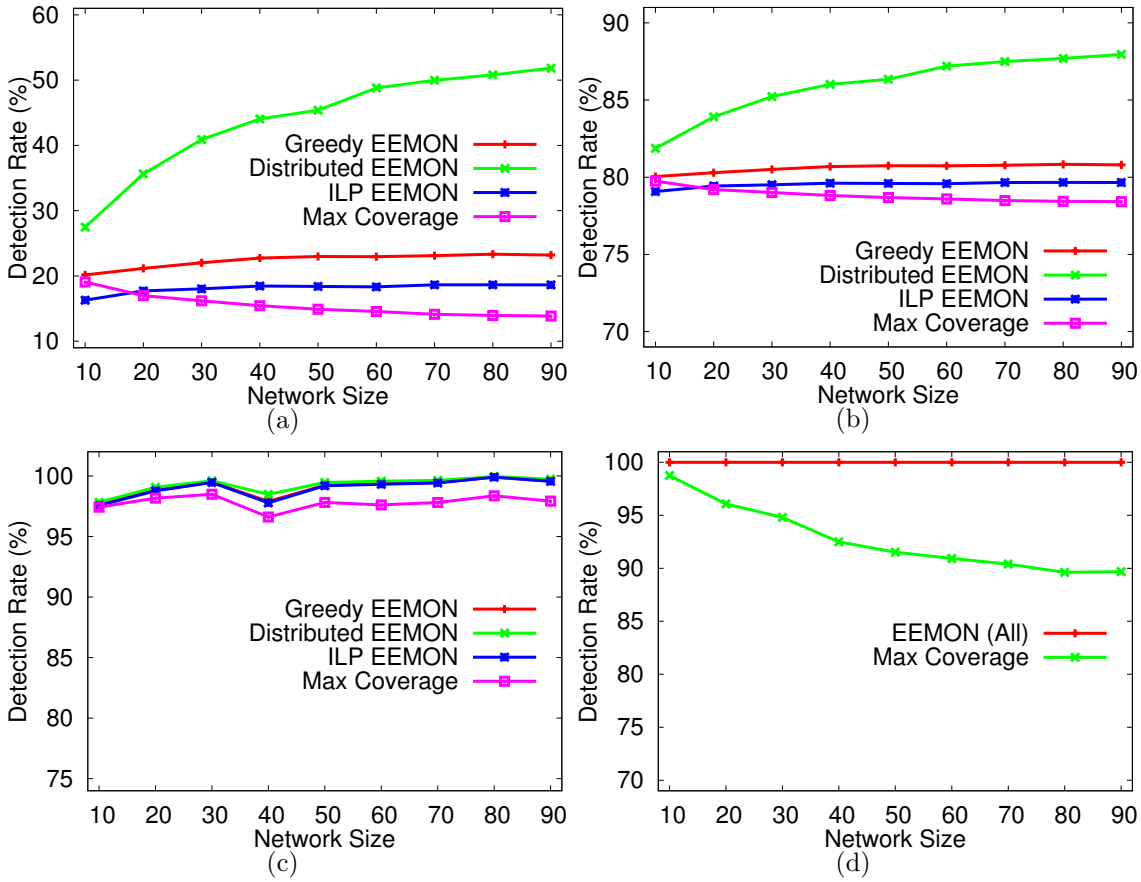


Figure 7.6: EEMON average Detection rate for (a) Severe Single-hop attacks. (b) All combinations of Normal/Severe and Single-hop/Multi-hop attacks. (c) EEMON-Aware attacks. (d) Jamming attack

the WMN as normal attacks. Hence, the detection rate of all monitoring mechanisms (i.e., TRAIN, EEMON, and Max Coverage) for Normal attacks, either single-hop or multi-hop, is 100% as the attack traffic is certainly monitored by a WMN router running at least LW-DS configuration. We note here that this detection rate is for those attackers either connected to the WMN network (insider) or communicating with a WMN host through the gateway (authenticated outsiders). Those attackers who are not connected to the WMN (unauthorized outsiders) are assumed to only threat communication links and will be evaluated at the end of this section.

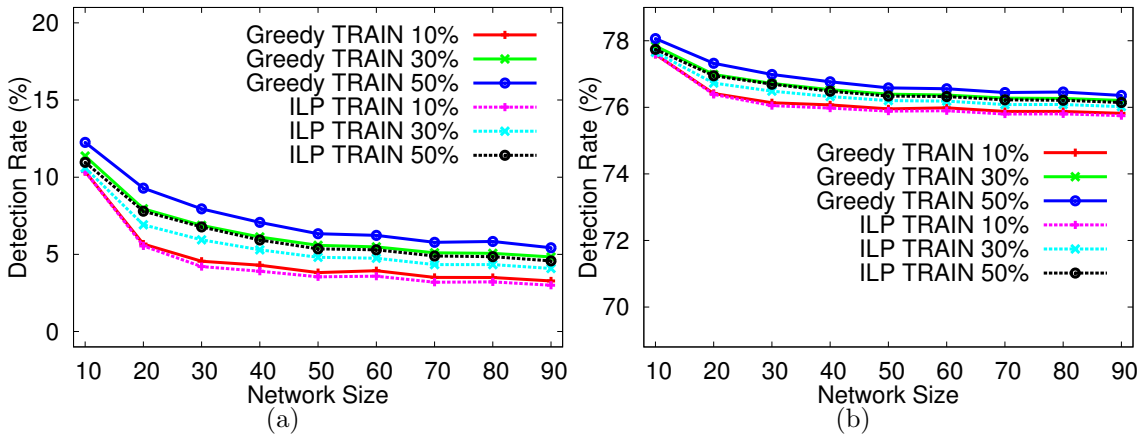


Figure 7.7: TRAIN average Detection rate for (a) Severe Single-hop attacks, (b) All combinations of Normal/Severe and Single-hop/Multi-hop attacks.

7.3.3.2 Severe Attacks

Severe attacks include those that are only detectable by monitoring nodes running CP-DS. Similar to the Normal attacks, Severe attacks are defined by security administrator. Since multi-hop attacks are always detectable by both EEMON and TRAIN, as they monitor all $Path_{ij}$, the administrator must configure network such that Severe attacks can only occur in a multi-hop manner. For example, the attacks against WMN file servers can be defined as Severe attacks since the location of file servers are always known (e.g., in DMZ) and the attacker has to run *multi-hop attacks* against them. Hence, Severe Multi-hop attacks are also considered to be always detected (100% detection rate) if they are properly defined by the administrator. However, there is always a possibility for considering some attacks as Severe while the attacker can launch them in a single-hop mode, e.g., an unknown vulnerability on the operating system of local router or local clients/hosts. Single-hop attacks are detectable if local router v_i is a monitoring node and monitors all e'_{ir} by CP-DS. Thus, the monitoring solutions that select fewer nodes as monitoring

nodes (e.g., ILP EEMON) are much more vulnerable to Severe Single-hop attacks. We evaluate the detection rate of all monitoring solutions for $10 \times N$ of each network size for Severe Single-hop attacks.

Figures 7.6(a) and 7.7(a) show the average detection rate for Severe Single-hop attacks in EEMON and Max Coverage algorithms and TRAIN algorithms, respectively. As depicted, the detection rates are very low since the number of monitoring nodes are minimal in all these mechanisms. These results emphasize the trade off between resource consumption and detection rate in WMN, although this type of attacks are supposed to occur rarely. We repeated this experiment for previously discussed attacks: $10 \times N$ Normal Single-hop, $10 \times N$ Normal Multi-hop, and $10 \times N$ Severe Multi-hop, although the probability of having these attacks in the network is not necessarily equal in real WMN. We then measured the average detection rate over all these $40 \times N$ different attacks for the aforementioned attacks. The average detection rate of these attacks are shown in Figures 7.6(b) and 7.7(b). As depicted, the average detection rates are much higher than those for Severe Single-hop attacks because the other attacks (e.g., all Normal attacks) are always detected.

7.3.3.3 EEMON and TRAIN Aware Attacker

EEMON and TRAIN aware attacker are those type of attacks where attacker knows which monitoring solutions is used by the administrator (e.g., traffic-agnostic or traffic-aware, link coverage or node coverage, etc.) but do not know which attack is considered to be Severe or Normal. It is worth mentioning that the attacker might be able to obtain more information about all other security settings through IDS localization and IDS module localization attacks. However, these types of attacks are out of the scope of this research and we assume that attacker only knows about the monitoring solution. For example, if the attacker knows that EEMON is used by

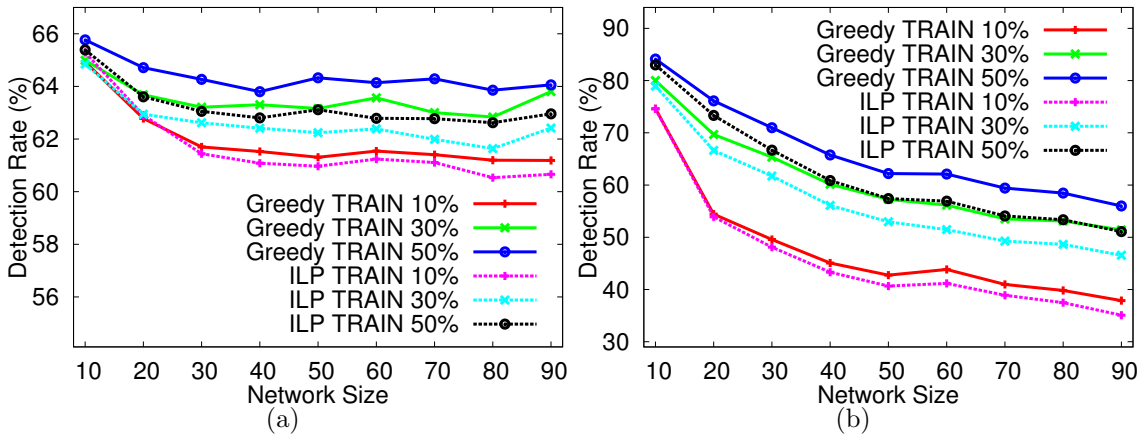


Figure 7.8: TRAIN average Detection rate for (a) TRAIN-Aware attacks, (b) Jamming attack.

the administrator, it only runs single-hop attacks, although he/she does not know which attack is considered as Severe or Normal. Similarly, if the attacker knows that TRAIN is used by the administrator, he/she tries to run attacks against intermediate nodes on WMN paths to avoid monitoring node on the route.

As shown in Figure 7.6(c) for $10 \times N$ attacks per each network size, an attacker who is aware of EEMON has yet a very low chance of success (more than 95% detection rates) since the only undetectable attacks are Severe Single-hop attacks and the severity of the attacks are considered to be unknown to the attackers. The detection rate for Max Coverage algorithm is slightly below EEMON algorithms as it leaves some links uncovered. Figure 7.8(a) shows the average detection rate for all TRAIN algorithms where the TRAIN-aware attacker may launch attack against some intermediate nodes. As depicted, TRAIN-aware attacks are harder to detect as there is only one monitoring node per path and if happen to be near the destination, it cannot detect the attacks against intermediate nodes.

7.3.3.4 Link-Based Attacks

Link-Based attacks are those that an unauthorized node (not connected to the WMN but physically located in WMN area) can launch against WMN communication links, e.g., Jamming attacks. We consider this attack as a Severe attack detectable by monitoring nodes. Depending on the total link coverage provided by a monitoring mechanism, these single-hop link-based attacks may or may not be detected. Figures 7.6(d) and 7.8(b) show the average detection rates for Jamming attacks launched at random locations in WMN. The results are obtained from $10 \times N$ attacks on 500 random networks for each given network size. As depicted, the monitoring node selected by all EEMON algorithms can detect all link-based attacks as they aim at monitoring all communication links. However, Max Coverage and TRAIN mechanisms may miss up to 10% and 40% of the link-based attacks, respectively. The high false negative rate in TRAIN is because it is a traffic-aware solution and is not designed for single-hop attacks.

8. ATTACK AND FAULT TOLERANT WIRELESS IDS

Although intrusion detection mechanism in WMN have received considerable focus, little attention has been paid to attacks-and-failures against/of IDS nodes. Undoubtedly, when an IDS node is compromised or faulty, it is unable to participate in intrusion detection process, thus, the intrusion detection rate will decrease and some malicious activities will remain undetected (i.e., high false negative rates). This research, inspired by similar efforts in other computer networking areas [11, 55, 56, 61, 96], investigates the attack-and-fault tolerance of IDS solutions presented in this research. In order to develop AFT mechanisms for the proposed IDS solutions, we first survey related works proposed for attack or fault tolerance in all networking areas (e.g., wired, ad hoc and sensor networks). Some of the proposed solutions use redundant/backup nodes [55] to increase the network/service availability after node compromise/failure while others concentrate on camouflaging mechanisms [61,96] to make monitoring/IDS nodes localization [11] very hard for the attacker. Furthermore, few other solutions propose fast and efficient fault detection mechanisms to detect compromised or faulty nodes [56] and recover the network from that situation [55, 61].

This research thrust proposes a classification for all AFT mechanisms and then concentrates on *preventive* solutions that use redundant IDS nodes to maintain high IDS availability ratio after IDS compromise/failure times. We will show that these mechanisms, at the price of higher resource consumption, increase the attack/fault tolerance level by: 1) increasing IDS availability; 2) reducing IDS compromise/failure detection time; and 3) eliminating the need for recovery actions (i.e., adopting backup nodes) [55, 61]. Taking into consideration the optimal monitoring mechanism em-

ployed by IDS solutions, we *reformulate* the optimal monitoring problem for intrusion detection in each class of IDS such that the solution becomes an AFT IDS. The performance (e.g., intrusion detection rate) and efficiency (e.g., resource consumption) of *redesigned* and *preventive* monitoring mechanisms proposed for each class of IDS are evaluated and compared to those of the original solutions. The results show how AFT design trades off attack-and-fault tolerance levels for the amount of resources consumed by intrusion detection systems.

8.1 AFT Mechanisms Diagram

The IDS mechanism presented in this research are not AFT (except for some special cases of the RAPID protocol, which we will explain in next sections). Therefore, if an IDS node fails (e.g., runs out of memory and crashes or its battery dies) or become compromised, part of the network will remain uncovered. This means that some WMN nodes/links become vulnerable against network attacks and that false negative rates will increase. Inspired by research in AFT design [11, 55, 56, 61, 96] we propose a classification which, to the best of our knowledge, is the first for AFT intrusion detection. Our proposed classification is based on the time of the action taken for AFT purposes. As shown in Figure 8.1, the actions are either taken before IDS attack or fault time (i.e., resulting in IDS compromise/failure) or after that.

8.1.1 Prevention Phase

As shown in Figure 8.1, prevention phase refers to the time while the IDS compromise/failure has not occurred yet. For example, a *preventive* AFT mechanism [61] may aim at increasing the risk of IDS node attack for the attacker (e.g., by using redundant monitoring node per link) or reducing the chance of node failure (e.g., by using high capacity storage or energy sources). Therefore, preventive solutions pay the AFT prices (i.e., redundant resources) at the design and implementation

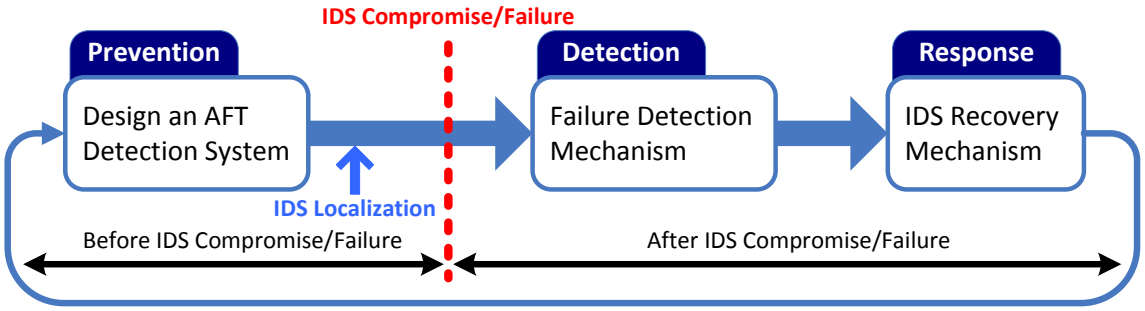


Figure 8.1: A multi-phase process for designing an AFT IDS mechanism.

phase so that the IDS availability (detection coverage ratio) will not be affected after IDS compromise/failure. *This research focuses on preventive mechanisms and evaluates the performance of preventive AFT designs and their costs.* It is important to mention that there exists solutions focusing on IDS node camouflaging, so that IDS localization (as shown in Figure 8.1) becomes very hard for the attacker [11, 96].

8.1.2 Detection Phase

If preventive mechanisms are not used, the monitoring system must be able to detect IDS compromise/failure immediately, so the security administrator can recover the IDS mechanism quickly. The time between IDS compromise/failure and its detection by security administrator is called *detection time*. A fast and accurate detection mechanism can remarkably reduce the detection time and increase the IDS availability time. Detection mechanisms [55, 56] can be either proactive or reactive. It is worth mentioning that a preventive AFT mechanism that uses redundant monitoring nodes is already a real-time detection system since every IDS node is monitored by at least another IDS node.

8.1.3 Response Phase

When the IDS compromise/failure occurs and it is detected, an appropriate action is to recover the node(s) from the compromise/fault. The time between the IDS

compromise/failure detection and its recovery is called *response time*. An optimal recovery mechanism minimizes the response time [55, 61]. Recovery mechanism and response time usually depend on the network topology, application, and IDS solution used in the network. We note here that although preventive solutions do not need detection and response mechanisms, it is very beneficial to consider these two mechanisms particularly for highly vulnerable WMN. This is because a preventive mechanism ultimately becomes non-preventive after a few IDS node compromises/failures.

8.2 AFT-Design for WMN IDS

We model a mesh network as a graph $G = (V, E)$, in which V is the set of WMN nodes $\{v_1, v_2, \cdots, v_N\}$, and $E = \{e_1, e_2, \cdots, e_Q\}$ is the set of links between them. For the traffic-aware solutions, we denote the number of nodes and links located on traffic routes by n ($n \leq N$) and q ($q \leq Q$), respectively. Therefore, the reduced graph $G' = \{V', E'\}$ represents the set of active nodes and links in traffic-aware WMN, where V' is the set of n active nodes ($V' \subseteq V$), and E' is the set of q active links ($E' \subseteq E$). The set of selected monitoring (IDS) nodes in the resourceful classes are denoted by $M = \{m_j \,|\, m_j \text{ is a monitoring node }\}$. We also denote the set of routing paths for the network traffic by $P = \{p_1, p_2, \cdots, p_l\}$, where $P_i^v = \{v_j \,|\, v_j \text{ is located on } p_i\}$ and $P_i^v \subseteq V'$. We denote by matrix $\mathbb{G}_{Q \times N}$ the mapping between nodes and links, i.e., $g_{hj} = 1$ iff node v_j can monitor link e_h. We also denote by matrix $\mathbb{T}_{l \times n}$ the mapping between nodes and paths, i.e., $t_{ij} = 1$ iff node j is located on path i.

We denote the residual energy and the communication load of a WMN node by b_j and c_j, respectively. Based on the maximum residual charge and communication load a node can have, both b_j and c_j are considered normalized values in range $[0, 1]$. Let $w : V \longrightarrow [0, 1]$ be a cost function that assigns a weight w_j to a node

v_j based on c_j and b_j ($w_i = w(c_j, b_j) = 1/(c_j \times b_j)$), such that higher normalized c_j and b_j values result in lower weight being assigned to v_j. We also denote the set of IDS functions by $\mathcal{F} = \{f_r \,|\, f_r$ is a set of detection rules $\}$ with size R (i.e., $|\mathcal{F}| = R$). Let $w^f : \{\mathcal{F}\} \longrightarrow [0, 1]$ be a cost function that assigns memory load w_r^f to IDS function f_r (as used in resourceless IDS solutions). Consequently, vector $W^f = [w_1^f, w_2^f, \cdots, w_R^f]$ represents memory loads for the IDS functions in $\mathcal{F}$, i.e., the amount of memory load each function imposes to the IDS node when activated on that IDS node. We use matrix $\mathbb{X}$ to show whether node v_j performs IDS function f_r (i.e., $x_{jr} = 1$) or not. Finally, vectors $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_N]$ (*i.e., Battery Threshold*) and $\Lambda = [\lambda_1, \lambda_2, \cdots, \lambda_N]$ (*i.e., Memory Threshold*) represent the minimum energy charge required for being selected as monitor and maximum allowable memory load by IDS functions, respectively.

8.2.1 Resourceful IDS

EEMON aims at covering all communication links while TRAIN aims at covering all traffic paths, both with minimum average cost per monitoring nodes. Let S_h ($S_h \subseteq M$) be the set of selected monitoring nodes out of all possible nodes that can monitor *link* e_h, and similarly S_i' be the set of selected monitoring nodes out of all possible nodes that can monitor *path* p_i.

$$\text{minimize} \quad \sum_{v_j \in V} w_j m_j \tag{8.1}$$

$$\text{subject to:} \quad |S_h| \geq 1(\text{EEMON}) \quad ,\forall e_h \in E \tag{8.2}$$

$$|S'_i| \geq 1\,(\text{TRAIN}) \quad ,\forall p_i \in P$$

$$b_j \geq \beta_j\,(or\; b_{th}) \quad ,\forall m_j \in M \tag{8.3}$$

$$m_j \in \{0, 1\} \tag{8.4}$$

Therefore, the optimal monitoring problem in a battery-powered resourceful WMN (both EEMON and TRAIN) can be formulated as an integer linear program (ILP), where Constraint (8.2) indicates that every *link/path* must be covered; Constraint (8.3) enforces the algorithm to select the nodes with residual energy greater than a threshold. Constraint (8.4) means a node is either selected as a monitoring node or not.

8.2.1.1 AFT Resourceful IDS

We define δ-AFT design as an AFT IDS mechanism in which each node is monitored by $\delta+1$ monitoring node(s) and the intrusion detection monitoring mechanism can tolerate at most δ IDS compromise/failures per link/path. Hence, in EEMON and TRAIN optimal monitoring formulations, δ-AFT design is achieved by modifying constraint (8.2) to $|S_h| \geq \delta$ for EEMON and $|S'_i| \geq \delta$ for TRAIN. It is worth mentioning that δ is bounded by maximum number of monitoring nodes that can potentially monitor a link/path, which is a function of network density.

8.2.2 Resourceless IDS

The main objective of resourceless IDS solutions is to monitor all links/paths with the maximum allowable number of IDS functions that can be performed on WMN nodes. A higher number of detection modules executed on node v_j means more attack traffic can be detected on the links/paths monitored by that node. Hence, the optimal monitoring problem in resourceless WMN is formulated as the following ILP (for both PRIDE and RAPID):

$$\text{maximize} \quad (1/l)(\mathbf{1}^T \cdot \mathbb{T})(\mathbb{X} \cdot \mathbf{1}) \quad \text{(PRIDE)} \tag{8.5}$$

$$(1/q)(\mathbf{1}^T \cdot \mathbb{G})(\mathbb{X} \cdot \mathbf{1}) \quad \text{(RAPID)}$$

$$\text{subject to:} \quad \mathbb{X} \cdot {W^f}^T \leq \Lambda^T \tag{8.6}$$

$$(\mathbb{T} \cdot \mathbb{X})_{ir} \leq 1 \quad \text{(PRIDE)} \quad ,\forall i, r \tag{8.7}$$

$$(\mathbb{G} \cdot \mathbb{X})_{hr} \leq 1 \quad \text{(RAPID)} \quad ,\forall h, r$$

$$x_{jr} \in \{0, 1\} \quad \text{(PRIDE)} \quad ,\forall v_j \in V', \forall f_r \in \mathcal{F}_j \tag{8.8}$$

$$x_{jr} \in \{0, 1\} \quad \text{(RAPID)} \quad ,\forall v_j \in V, \forall f_r \in \mathcal{F}_j$$

where Constraint 8.6 limits the IDS memory load on every node v_j to be less than its memory threshold λ_j. Constraint 8.7 ensures that only one copy of each function is assigned to the nodes for each link/path. Finally, Constraint 8.8 means a node either performs an IDS function or not.

8.2.2.1 AFT Resourceless IDS

This class of IDS may not be able to achieve 100% link/path coverage (i.e., every link/path is monitored by all R IDS functions) due to memory constraint Λ. Suppose $\lambda_j = \lambda \, \forall v_j$, the smaller the λ is, the lower the link/path coverage will be. Therefore,

if the memory threshold is very low that does not allow us to achieve 100% coverage, our IDS is always 0-AFT. When the memory threshold λ increases, it is most likely possible to achieve δ-AFT design for $\delta > 0$ in resourceless IDS. Hence, in PRIDE and RAPID, achieving higher link/path coverage rate is more important than achieving δ-AFT design.

In order to achieve δ-AFT design in this class of IDS, we have to remove Constraint 8.7 to ensure that more than one IDS function can be assigned to a link/path. In this case, since redundant IDS functions do not count for coverage ratio (ILP objective) [36], we need to modify the ILP objective function so that it accurately measures the link/path coverage ratio. Thus, we define function $BN : \{\mathbb{Y}\} \longrightarrow \{0, 1\}$ that converts y_{ij} to a binary value, i.e., if $y_{ij} = 0$, $BN(y_{ij}) = 0$, otherwise $BN(y_{ij}) = 1$. We reformulate the optimal monitoring problem for δ-AFT design of resourceless IDS classes as follows:

$$\text{maximize} \quad (1/q)(\mathbf{1}^T \cdot \text{BN}(\mathbb{T} \cdot \mathbb{X}) \cdot \mathbf{1}) \quad \text{(PRIDE)} \tag{8.9}$$

$$(1/q)(\mathbf{1}^T \cdot \text{BN}(\mathbb{G} \cdot \mathbb{X}) \cdot \mathbf{1}) \quad \text{(RAPID)}$$

$$\text{subject to:} \quad \mathbb{X} \cdot {W^f}^T \leq \Lambda^T \tag{8.10}$$

$$x_{jr} \in \{0, 1\} \quad , \forall v_j, f_r \tag{8.11}$$

The new objective function is no longer linear [36] and cannot be solved with ILP solvers. Therefore, we use Genetic Algorithm (GA), a popular and effective type of evolutionary algorithms, as used in RAPID [36] to solve the optimal monitoring problem proposed for δ-AFT design in resourceless WMN.

8.2.3 Solutions for AFT-Design of IDS

Although some of the solutions proposed for the optimal monitoring in our IDS solutions are implemented in both centralized and distributed manners, here, we only consider their centralized algorithms to compare with their centralized AFT designs. The system and attacker models considered in this research (for AFT-designs) are exactly the same as those in their original designs. Similar to the original centralized solutions, the AFT-design solutions consider a WMN including mesh routers (i.e., battery powered in EEMON and TRAIN and AC-powered in RAPID and PRIDE) and a computationally powerful base station. Each router in the WMN has some local information (e.g., its communication load and its residual energy, processing/memory loads and traffic information) and periodically sends it, via a middleware and secure communication links, to the base station. Based on the collected information and the δ and λ values chosen by the security administrator for resourceful and resourceless IDS, respectively, the base station then solves the optimization problem and assigns intrusion detection tasks to the nodes.

8.2.3.1 AFT-Design Resourceful IDS

Similar to original EEMON, upon collecting nodes' information, the base station uses an ILP solver (i.e., *bintprog* function of MATLAB) to find the optimal set of monitoring nodes that can monitor all WMN links with $\delta + 1$ monitors. AFT-design TRAIN, as a traffic-aware solution, first removes *idle* nodes from the network, i.e., those not contributing in the traffic routing, and then optimally selects monitoring nodes (using *bintprog*) to monitor all traffic paths with $\delta + 1$ monitors. If the reduced WMN graph after removing idle nodes is disconnected, each graph component is considered as a sub-problem (to reduce the execution time) and solved separately.

8.2.3.2 AFT-Design Resourceless IDS

The base station in this classes performs a Genetic Algorithm to find the optimal IDS function distribution that provides maximum average link/path coverage ratio. GA solutions are encoded as bitstrings (i.e., chromosomes) of specific length and tested for fitness. In AFT-design PRIDE and RAPID formulations, matrix $\mathbb{X}$ is a solution that can be encoded as a chromosome of length $n \times R$ and the fitness (objective) value of each solution is the average link/path coverage in the WMN. The genetic operations used in redesigned PRIDE/RAPID are based on operations explained in [34] that their details are omitted here.

8.3 Performance Evaluation

This section presents simulation results of the proposed AFT design solutions for both resourceful and resourceless classes.

8.3.1 Resourceful IDS

This section presents simulation results for AFT designs of two resourceful IDS solutions, EEMON and TRAIN. As shown in EEMON and TRAIN and by considering their problem formulations presented in Section 8.2, the metrics we evaluate in this section are: 1) average number of nodes selected as monitoring nodes; 2) average communication load and average residual energy charge among selected nodes as monitoring nodes, in addition to the battery threshold reduction; 3) average link coverage and intrusion detection rates; 4) time complexity and average energy consumption; and 5) a new metric called expected δ for a given δ-AFT design as we will explain it later in this section. The results are obtained from 100 random networks for each network size. We note here that 0-AFT design in simulation results means the original unmodified IDS design.

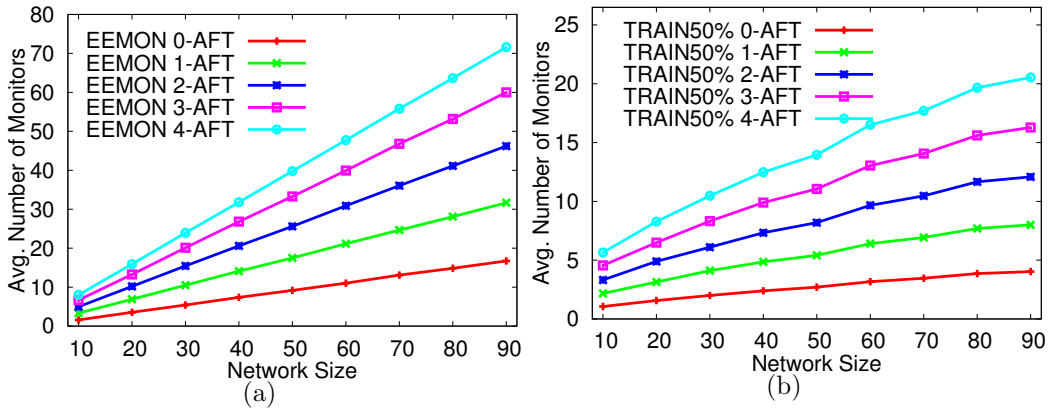


Figure 8.2: Average number of monitoring nodes for different δ in: (a) EEMON; (b) TRAIN 50%

8.3.1.1 Number of Monitors

The main objective in resourceful IDS solutions is to cover the entire network links/paths with minimum number of monitoring nodes and minimum total cost. Therefore, in a δ-AFT design, as δ increases, the number of nodes that must be selected as monitoring nodes will also increase (redundant monitoring nodes provide higher degree of attack and fault tolerance). Figures 8.2(a) and 8.2(b) show the average number of monitoring nodes for different δ and network sizes in EEMON and TRAIN, respectively. We note here that although TRAIN evaluates this metric for different number of paths (e.g., number of paths equals to 10%, 30%, and 50% of network size), we only consider the maximum case which is number of paths equals to $0.5 \times N$ and omit the other results due to space limitations. As shown, the number of monitoring nodes linearly increases (i.e., constant percentage of nodes are selected for different N) as δ increases in both traffic-agnostic and traffic-aware solutions to provide higher levels of attack and fault tolerance. For example, more than 80% of the nodes in EEMON are selected as monitoring nodes in 4-AFT design (i.e., higher costs to achieve larger δ).

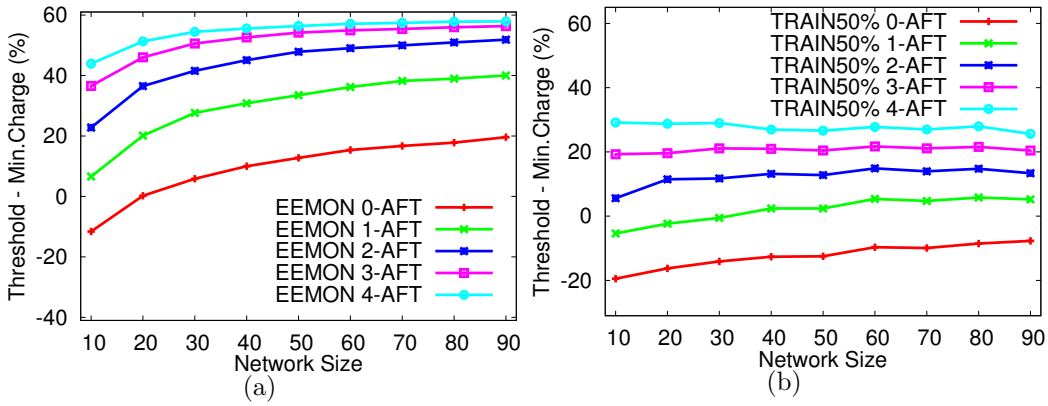


Figure 8.3: [B_{th} - Minimum Charge] among selected nodes for different δ in: (a) EEMON; (b) TRAIN 50%.

8.3.1.2 Properties of Monitoring Nodes

In EEMON and TRAIN, the cost per monitoring node is defined as a function of residual energy charge and the communication load. Therefore, the residual energy charge and the average communication load among selected node is expected to be higher than of those of non-monitoring nodes. In addition, it is possible that out of all possible nodes that can monitor a *link/path*, none of them has residual charge greater than threshold b_{th}. In this case, as mentioned in EEMON and TRAIN, the threshold decreases until at least one of the nodes is selected. Such a threshold reduction has to be as low as possible meaning that most of the selected nodes have residual energy charge above the threshold b_{th} resulting in longer network life time.

Figures 8.3(a) and 8.3(b) show the average value of [B_{th} - Minimum Charge] for different δ and network sizes in EEMON (TG-RF) and TRAIN (TW-RF), respectively. Negative values mean that the minimum residual energy charge among all selected nodes is larger than the threshold and no threshold reduction has occurred. As shown, the greater the δ is, the larger the threshold reduction will be. This is because selecting more monitoring nodes (required by large δ) increases the proba-

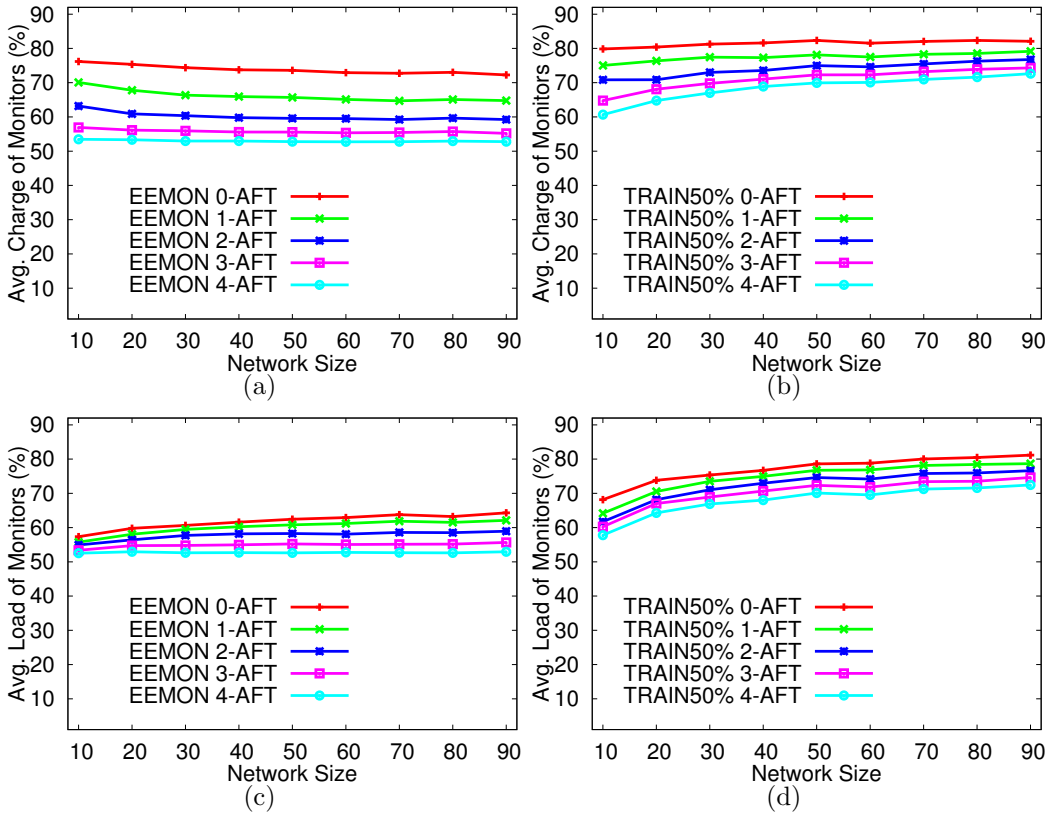


Figure 8.4: Average residual energy charge of selected nodes for different δ in: (a) EEMON; (b) TRAIN 50%; Average communication load of selected nodes for different δ in: (c) EEMON; (d) TRAIN 50%.

bility of selecting low battery nodes, and consequently increases the [B_{th} - Minimum Charge].

The next two metrics we consider are average residual charge and average communication load among selected nodes, as evaluated in both EEMON and TRAIN. Figures 8.4(a) and 8.4(b) depict the average residual energy charge among selected nodes (as monitoring nodes) for different δ and network sizes in EEMON and TRAIN, respectively. As depicted, the larger the δ is, the lower the average residual energy charge of monitoring nodes will be. This is because larger δ requires more monitoring nodes to achieve higher levels of attack and fault tolerance. Hence, the monitoring

node selection algorithms have to select monitors among low battery nodes that decreases the average value. Figures 8.4(c) and 8.4(d) show the average communication load of selected nodes for different δ and network sizes in EEMON and TRAIN, respectively. Similar to the average residual charge, the average communication load decreases as δ increases.

8.3.1.3 Intrusion Detection Rates

EEMON and TRAIN aim at covering all network links and paths respectively. Average link coverage in EEMON is always 100% but TRAIN only covers a subset of links located on active routing paths. Figure 8.3.1.2 shows the average link coverage provided by TRAIN 50% when δ increases. As shown, although the original TRAIN leaves some communication links uncovered, the AFT design of TRAIN increases the average link coverage as it selects more monitoring nodes than original TRAIN.

EEMON considers two types of attacks, Severe (detectable by only monitoring nodes) and Normal (detectable by monitoring and non-monitoring nodes). These two attacks can be launched in single-hop and multi-hop modes. The detection rate of EEMON and TRAIN for Normal attacks, either single-hop or multi-hop, is 100% as the attack traffic is certainly monitored by a node (either monitoring or non-monitoring). In addition, Severe multi-hop attacks are also considered to be 100% detectable as both EEMON and TRAIN have at least one monitoring node that monitors multi-hop traffic. The only attack that is hard to detect is Severe single-hop attack which is only detectable by monitoring nodes.

We performed $10 \times N$ random attacks for each of 4 types (i.e., 2 types and 2 modes) for different δ in EEMON and TRAIN 50% and measured the detection rates ($40 \times N$ random attacks for each network size). Figures 8.6(a) and 8.6(b) depict the average intrusion detection rates for all combinations of Severe/Normal and single-hop/multi-

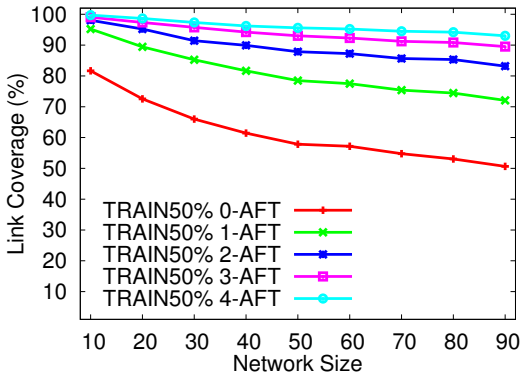


Figure 8.5: Average link coverage for different δ in TRAIN 50%.

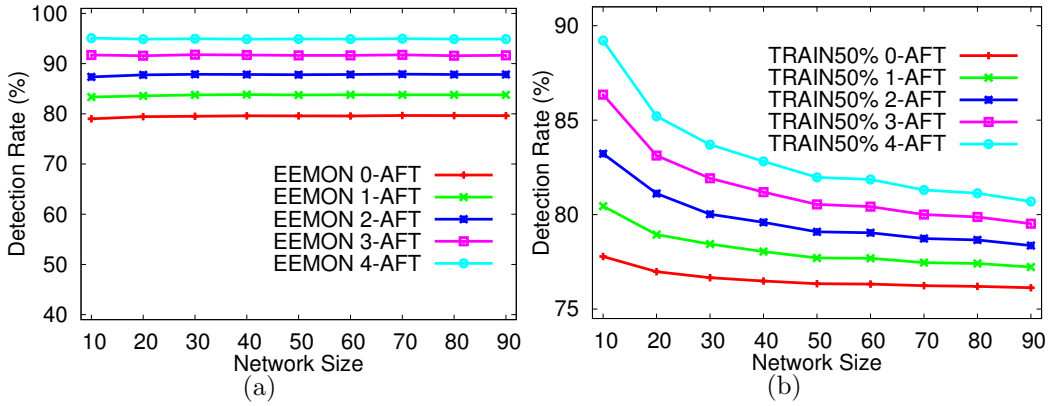


Figure 8.6: Average intrusion detection rate of all $40 \times N$ random Normal/Severe and Single-hop/Multi-hop attacks in: (a) EEMON; (b) TRAIN 50%.

hop attacks in EEMON and TRAIN, respectively. As depicted, larger δ increases the average intrusion detection rate since it results in selecting more monitoring nodes in the network that can detect Severe single-hop attacks and consequently increases the average detection rate. The lower detection rate in TRAIN (for similar δ-AFT designs as EEMON) is because it aims at only covering few paths (a subset of links) which results in selecting less monitoring nodes.

The next type of attack we consider is *EEMON and TRAIN aware* attack where attacker knows which IDS solutions is used (e.g., traffic-agnostic or traffic-aware, link

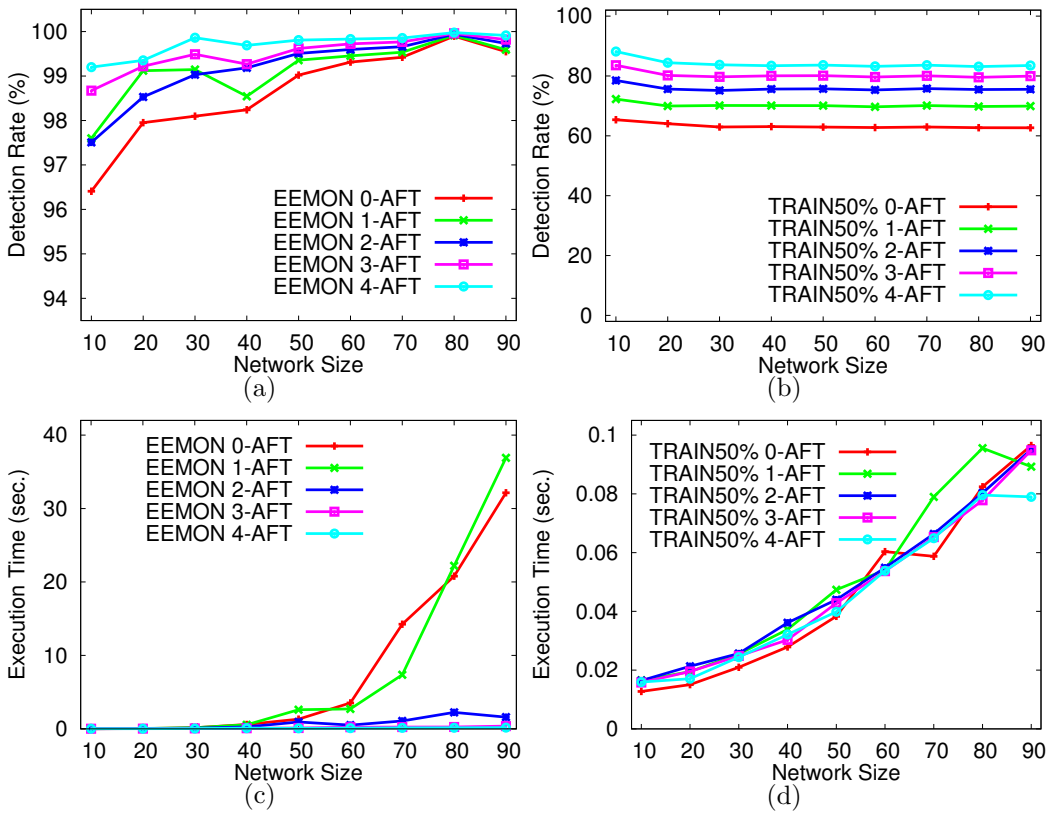


Figure 8.7: Average intrusion detection rate of EEMON/TRAIN aware attacks for different δ in: (a) EEMON; (b) TRAIN 50%. Average execution time of the ILP solver for different δ in: (c) EEMON; (d) TRAIN 50%.

coverage or node coverage, etc.) but do not know what type of attack is considered to be Severe or Normal. For example, if the attacker knows that EEMON is used, he will only run single-hop attacks and if TRAIN is used, he will try to run attacks against intermediate nodes on traffic paths to avoid monitoring node on the route. Figures 8.7(a) and 8.7(b) show the average intrusion detection rates of *EEMON and TRAIN aware* attacks ($10 \times N$ random attacks for each N) in EEMON and TRAIN 50%, respectively. It is worth mentioning that EEMON, at the price of using more monitoring nodes, achieves higher detection rates than TRAIN for a given network size. Also, as δ increases, the detection rate increases too because of selecting more

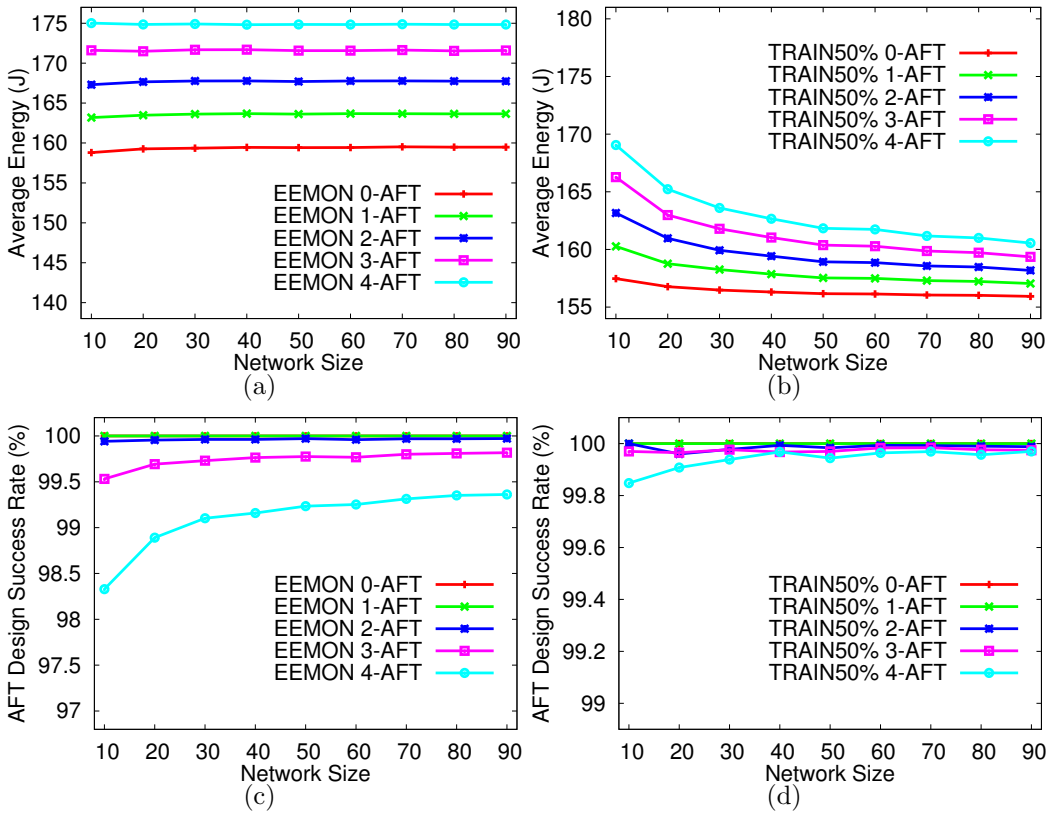


Figure 8.8: Average energy consumption of 50% duty cycling for different δ in: (a) EEMON; (b) TRAIN 50%. The ratio of number of selected monitors to the expected number of monitors for different δ in: (c) EEMON; (d) TRAIN 50%.

monitoring nodes in the network.

8.3.1.4 Time Complexity and Energy Consumption

Figures 8.7(c) and 8.7(d) show the execution time of the ILP solver when solving the optimization problem in EEMON and TRAIN, respectively. The results show the average execution time of different δ and network sizes. Generally, the execution time increases as network size (number of links/paths to be covered) increases. In addition, smaller δ increases the time complexity of the ILP solver since it reduces the solution space. As the results show, the execution time in TRAIN is always less than 0.1 seconds since it only considers traffic paths, however, the execution time in

EEMON is in the order of few seconds (as it considers all communication links). We note here that higher execution times for large networks in EEMON are also because of some outliers among 100 random networks.

In both EEMON and TRAIN, non-monitoring nodes work in duty-cycling mode to save energy. Thus, the set of monitoring nodes changes periodically (based on the problem formulation) to extend the network life time. The current consumption of devices used in EEMON and TRAIN (i.e., Linksys mesh routers) is 250mA, which means each device consumes 3 Watts (12V250mA). Thus, the energy consumed by each device during one minute working time (i.e., an epoch in our experiment) is 180 Joule. When duty-cycling, the energy consumption decreases depending on the duty-cycle interval. Figures 8.8(a) and 8.8(b) show the average energy consumption per node during an epoch for different δ in EEMON and TRAIN, respectively. As shown, the larger the δ is, the higher the average energy consumption will be. This is because larger δ means more nodes will work in monitoring mode and less nodes can save energy through duty-cycling.

8.3.1.5 Success Rate of δ-AFT Design

The last metric we evaluate in resourceful IDS class is the success rate of δ-AFT design in assigning $\delta + 1$ monitoring node(s) to each communication link/path. Since the number of monitoring nodes assigned to each link/path is limited by the maximum number of nodes that can cover the link/path, it is sometimes impossible to achieve δ-AFT for a given δ and network topology. In fact, the success rate of δ-AFT design in assigning $\delta + 1$ monitoring node(s) to a link depends on the network topology. We performed simulations for 100 random networks of each given network size and different δ and measured the average number of monitoring nodes per links/paths divided by δ. Figures 8.8(c) and 8.8(d) depict the success rates of δ-

AFT design for different δ in EEMON and TRAIN, respectively. As one can observe, the success rate is always near 100% specially for TRAIN as it monitors less links than EEMON.

8.3.2 Resourceless IDS

This section evaluates the performance of resourceless IDS solutions for AFT design. As we discussed in Section 8.2, the main parameter in designing resourceless IDS for traffic-agnostic and traffic-aware networks is memory threshold (λ). The larger the λ is, the higher the link/path coverage will be. This is because larger λ allows nodes to execute more IDS functions which also increases the IDS function redundancy (i.e., higher levels of attack and fault tolerance). Consequently, it increases intrusion detection rates and average memory load on the nodes. Hence, in resourceless IDS, unlike resourceful IDS, we cannot change δ as a tuning parameter for AFT design, however, δ is a function of λ and network density. In other words, the security administrator gives a higher priority to link/path coverage than AFT design because for example, having two identical (redundant) IDS functions on a path is not as useful as executing two different IDS functions on the nodes along the paths. Obviously, the later provides higher path coverage (and consequently higher detection rates) than the former (i.e., lower path coverage but higher level of attack and fault tolerance).

Figures 8.9(a) and 8.9(b) show the average number of IDS functions per links in RAPID for 6-module and 12-module configurations, respectively. As shown, this metric is a function of memory threshold (λ) and network density. The larger the λ and network density are, the more IDS function per link (i.e., the level of attack and fault tolerance) will be. Similarly, Figures 8.9(c) and 8.9(d) depict the average number of IDS functions per paths in PRIDE for 6-module and 12-module configu-

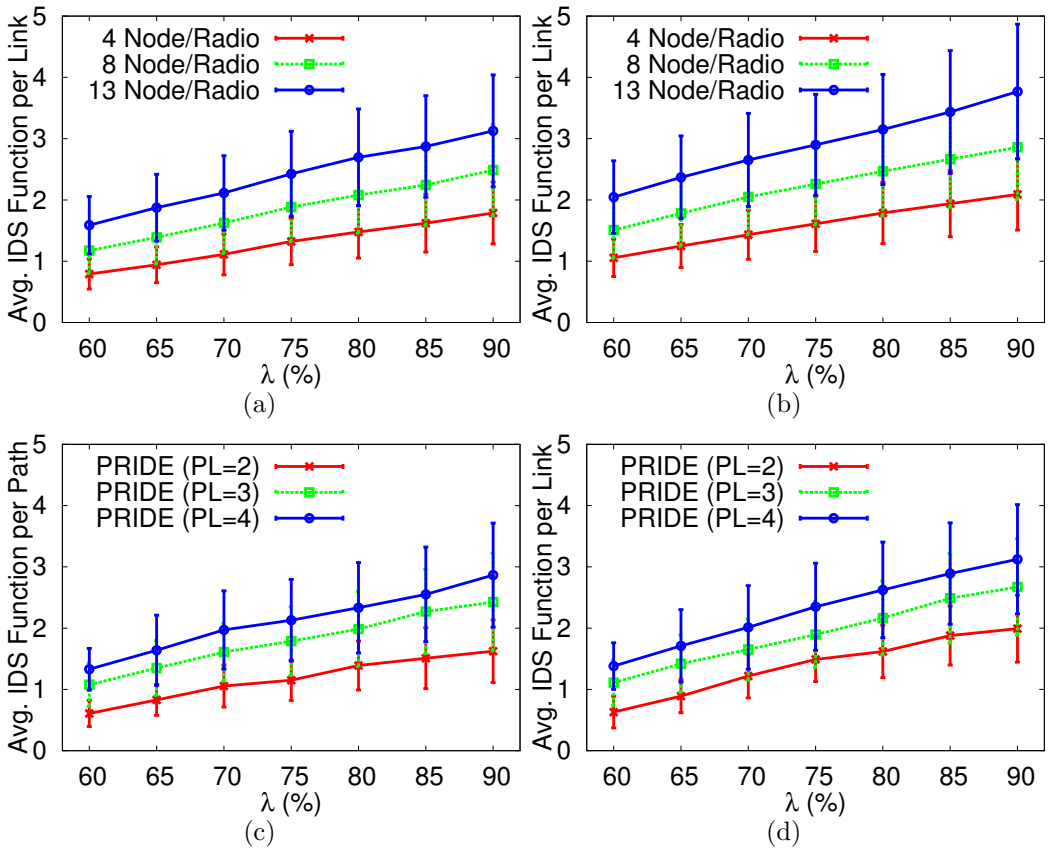


Figure 8.9: The average IDS functions per link for different memory threshold (λ) and network densities in: (a) 6-Module Configuration; (b) 12-Module Configuration RAPID. The average IDS functions per path for different memory threshold (λ) and path lengths (PL) in: (c) 6-Module Configuration; (d) 12-Module Configuration PRIDE.

rations, respectively. In PRIDE, since only the nodes located on the path participate in path monitoring, the level of attack and fault tolerance is a function of path length (PL) and λ. The higher the λ and PL are, the higher the attack and fault tolerance level will be. We note here that other metrics such as intrusion detection rates and average memory loads (omitted here) in RAPID and PRIDE are exactly the same as those shown in [36, 39].

9. CONCLUSIONS AND FUTURE WORK

In this section, we summarize the research, review the contributions, and discuss the future work.

This research investigates the problem of intrusion detection in wireless mesh networks (WMN). When compared to traditional intrusion detection systems (IDS) proposed for wired networks, intrusion detection in WMN is more challenging due to the limited resources (e.g., memory, CPU, and energy) available on WMN nodes and also lack of single vantage points where traffic can be analyzed (e.g., gateways and routers in wired networks). Inspired by thorough research on intrusion detection in wireless ad hoc and sensor networks, researchers have proposed decentralized monitoring mechanisms for intrusion detection in WMN. Decentralized solutions propose to distribute IDS responsibility to all nodes such that the entire WMN traffic is inspected. However, the major challenges that hinder the performance of these mechanisms are resources (e.g., *energy, processing*, and *storage* capabilities) accompanied by the adhoc-dynamic communication flows.

In light of these challenges, we proposed a taxonomy for different monitoring mechanisms proposed for intrusion detection in WMN. In fact, our objective in this research is not to propose a new intrusion detection rule, but to propose optimal approaches of applying proposed intrusion detection rules and engines to different WMN applications. In order to do that, we investigated the feasibility of applying state-of-the-art decentralized IDS solutions to different WMN applications. Our proposed classification is based WMN nodes' characteristics in different applications, types of services that the WMN provides, and also the administrative knowledge about network traffic. For each class of IDS, we propose optimal IDS role assignment

that aims at providing maximum intrusion detection rates with respect to all network characteristics and constraints. We also investigate the problem of IDS attack/failure for all of the proposed IDS solutions in this research.

9.1 Contributions

We proposed five intrusion detection systems categorized in four classes where each class of IDS is suitable for specific type of WMN. The first class of IDS solutions considers resource-constrained WMN where network administrator has knowledge about network traffic. We proposed PRIDE, a practical and traffic-aware intrusion detection mechanism, for this class of IDS that optimally distributes IDS functions along WMN nodes located on the traffic paths. We presented this problem as an integer linear program (ILP) and used ILP solver to solve it. We evaluated the performance of PRIDE solutions in a real-world department-wide testbed. Our experimental results show that PRIDE can achieve high detection rates in multi-hop attack scenarios even for small path lengths and low memory thresholds.

The second class of intrusion detection we studied in this research focuses on traffic-agnostic and resourceless WMN. This research thrust is motivated by the fact that traffic-awareness is a strong assumption in many WMN application in which traffic paths change very often. We proposed RAPID to monitors all communication links, instead of only few paths. It was shown that the complexity of this problem, i.e., monitoring all WMN links with resource-constrained nodes, is more than traffic-aware solutions that only consider few WMN link. Each node in RAPID, depending on its available resources, is assigned a subset of IDS functions and investigates the entire network traffic on the set of communication *links* it can monitor (i.e., in its coverage area). We proposed two algorithms to solve optimal IDS function distribution problem; an optimization algorithm based on evolutionary algorithms

and a random distributed mechanism. The results surprisingly showed that RAPID can achieve high detection rates in memory-constrained WMN. We also considered WMN application with extremely resource-constrained nodes where RAPID cannot provide reasonable intrusion detection rates. In this case, we use cooperative IDS where nodes optimally create cooperative intrusion detection groups with respect to multiple characteristics and constraints. The problem of creating optimal cooperative IDS clusters formulated as a multi-objective optimization problem and then an evolutionary algorithm was proposed to solve it.

The other two IDS solutions we proposed in this research consider battery-powered WMN consisting of resourceful nodes. The research challenge/problem we addressed by these IDS solutions was how to reconcile energy efficient operation, which requires nodes to save energy as much as possible, with an effective intrusion detection, which requires nodes to perform intrusion detection tasks resulting in extra energy consumption. The problem of energy-efficient monitoring was studied in these two classes of IDS while EEMON focuses on traffic-agnostic application and TRAIN studies the effect of traffic-awareness on such IDS solutions.

The last contribution in this research is to investigate the problem of attack and fault tolerance of the proposed IDS solutions. We first surveyed a series of administrative mechanisms for attack-and-fault tolerant (AFT) IDS design and proposes a classification for all AFT mechanisms and then concentrated on *preventive* solutions. We proposed redesigned IDS solutions that were attack and fault tolerant and then showed that AFT mechanisms, at the price of higher resource consumption, can increase the attack/fault tolerance level of IDS solutions in WMN.

9.2 Future Work

Taking into consideration that intrusion detection in resource-constrained wireless networks is a new promising research area and only few research efforts have been devoted to it, we envision several interesting future directions:

9.2.1 Intelligent Routing for Traffic-Aware IDS

In two research thrusts of this dissertation, we concentrated on traffic-aware IDS mechanisms where the knowledge about routing paths could help security administrators to optimally assign detection modules to WMN nodes (PRIDE) or optimally select monitoring nodes (TRAIN). In PRIDE, we showed that for a given memory constraint, the longer the path was, the higher the detection rates were. In addition, TRAIN aims at selecting nodes with higher residual energy charge and communication loads as monitoring nodes. Looking to the future, we also plan to modify WMN routing protocols (e.g., OLSR) and propose an intelligent routing mechanism that aims at redirecting WMN traffic through longer paths (to increase the number of nodes participating in traffic investigation) when using resource-constrained mechanisms, or also redirecting WMN traffic towards nodes that impose less costs to the set of monitoring nodes (those that have higher residual charge). This intelligent routing mechanism will tend to increase detection rates and lower monitoring costs at the price of some routing delays. The tradeoff between routing delays and intrusion detection rates would be an interesting research direction.

9.2.2 Intrusion Detection in Resource-Constrained and Dynamic Networks

This dissertation concentrates on static WMN where network topology does not change very often. However, recent advances in mobile networks show that these type of resource-constrained wireless networks are becoming more popular. Consequently

security of these networks will be of paramount importance. RAPID proposed a traffic-agnostic IDS for resource-constrained WMN, however, its performance in mobile network, where there is no central administrator and network topology changes very often, has to be evaluated. In such networks, only distributed algorithms can be considered, but it is possible that some selfish node do not participate in intrusion detection mechanism to conserve more energy.

9.2.3 Load-Awareness in IDS Role Assignment

PRIDE uses a fixed upper bound for number of concurrent network sessions that the IDS on each node can investigate. Considering the ever increasing networking services and consequently traffic loads, this setting will drastically degrade the IDS performance. Looking to the future, we also plan to investigate how dynamic memory allocation, e.g., Stream5 parameters such as "max_tcp," can be implemented based on traffic loads on each part of the network. Dynamic memory allocation and IDS function assignment This way, two nodes with the same memory space available but different traffic rates, will dedicate their memory spaces to the static and dynamic loads differently.

9.2.4 Intrusion Detection in Resource-Constrained IoT

The new trend in Internet is the network of interconnected wireless physical devices, sensors, and object where they interact through a worldwide communication infrastructure to provide different services. Internet of Things(IoT) has the potential to be the next evolution in the area of information technology. For such a ubiquitous technology, security is one of the top concerns. In order to provide a strong security foundation for IoT, several parameters has to be taken into considerations. Recently, intrusion detection and fault tolerance in IoT have received some attention from research community. We believe that lessons learned from intrusion detection

in resource-constrained wireless networks can help researchers in developing IDS mechanisms for IoT.

9.2.5 Cyber Physical Systems Security

Recently, the boundary between cyber systems and physical systems has been quite blurring. In fact, the number of cyber devices (e.g., phones) being able to interact with physical world is increasing. This interaction is done through different sensors (e.g., GPS, accelerometer, gyroscope, etc.) and help people to apply these systems to different applications, e.g., medical, transportation, home-security, and many other critical services. Taking into account the popularity of CPS application, their security is among the top concerns. Securing CPS against malicious activities or physical faults is of utmost importance. As a very new and promising research area, intrusion and fault detection is considered as another future direction.

REFERENCES

[1] Michael Adeyeye and Paul Gardner-Stephen. The Village Telco project: a reliable and practical wireless mesh telephony infrastructure. *EURASIP Journal on Wireless Communications and Networking*, 2011:78, 2011.

[2] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer Networks and ISDN Systems*, 47(4):445–487, 2005.

[3] Jamal N. Al-Karaki, Raza Ul-Mustafa, and Ahmed E. Kamal. Data aggregation and routing in wireless sensor networks: Optimal and heuristic algorithms. *Computer Networks*, 53:945–960, May 2009.

[4] Yair Amir, Claudiu Danilov, Raluca Musăloiu-Elefteri, and Nilo Rivera. The SMesh wireless mesh network. *ACM Transactions on Computer Systems*, 28(3):6:1–6:49, September 2008.

[5] Nils Aschenbruck, Jan Bauer, Raphael Ernst, Christoph Fuchs, and Jonathan Kirchhoff. Poster: Deploying a mesh-based command and control sensing system in a disaster area maneuver. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 335–336, Seattle, Washington, USA, 2011.

[6] Jonathan Backens, Gregory Mweemba, and Gertjan Van Stam. A rural implementation of a 52 node mixed wireless mesh network in Macha, Zambia. *EInfrastructures and EServices on Developing Countries*, pages 32 – 39, 2010.

[7] G.H. Badawy, A.A. Sayegh, and T.D. Todd. Energy provisioning in solar-powered wireless mesh networks. *IEEE Transactions on Vehicular Technology*, 59(8):3859–3871, 2010.

[8] David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms : Part 1, fundamentals. *University Computing*, pages 58–69, 1993.

[9] Naouel Ben and Salem Jean pierre Hubaux. Securing wireless mesh networks. *IEEE Wireless Communications*, 13(2):50 –55, April 2006.

[10] Giacomo Bernardi, Peter Buneman, and Mahesh K. Marina. Tegola tiered mesh network testbed in rural scotland. In *Proceedings of the ACM Workshop on Wireless Networks and Systems for Developing Regions (WiNS-DR)*, pages 9–16, San Francisco, California, USA, 2008.

[11] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping internet sensors with probe response attacks. In *Proceedings of the 14th Conference on USENIX Security Symposium (SSYM)*, pages 13–13, Baltimore, MD, USA, 2005.

[12] Vijay Bhuse and Ajay Gupta. Anomaly intrusion detection in wireless sensor networks. *High Speed Networks*, 15:33–51, January 2006.

[13] Levente Buttyn and Pter Schaffer. Panel: Position-based aggregator node election in wireless sensor networks. In *Proceedings of the 4th IEEE International Conferenceon Mobile Ad Hoc and Sensor Systems (MASS)*, pages 1–9, Pisa, Italy, 2007.

[14] M.E.M. Campista, P.M. Esposito, I.M. Moraes, L.H.M.K. Costa, O.C.M.B. Duarte, D.G. Passos, C.V.N. de Albuquerque, D.C.M. Saade, and M.G. Rubinstein. Routing metrics and protocols for wireless mesh networks. *IEEE Network*, 22(1):6–12, 2008.

[15] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. WCA: a weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, April 2002.

[16] H. Chenji, A. Hassanzadeh, M. Won, Y. Li, W. Zhang, X. Yang, R. Stoleru, and G. Zhou. A wireless sensor, adhoc and delay tolerant network system for disaster response. Technical report, LENSS-09-02, 2011.

[17] Harsha Chenji, Wei Zhang, Radu Stoleru, and Clint Arnett. DistressNet: a disaster response system providing constant availability cloud-like services. *Ad Hoc Networks*, 11(8):2440–2460, November 2013.

[18] Harsha Chenji, Wei Zhang, Myounggyu Won, Radu. Stoleru, and Clint Arnett. A wireless system for reducing response time in urban search and rescue. In *Proceedings of 31st IEEE International Performance Computing and Communications Conference (IPCCC)*, pages 215–224, Austin, Texas, USA, 2012.

[19] Arun Chhetri, Huy Nguyen, Gabriel Scalosub, and Rong Zheng. On quality of monitoring for multi-channel wireless infrastructure networks. In *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 111–120, Chicago, Illinois, USA, 2010.

[20] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991.

[21] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the first ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, pages 16–23, Montreal, Quebec, Canada, 2005.

[22] Rodrigo do Carmo and Matthias Hollick. DogoIDS: a mobile and active intrusion detection system for IEEE 802.11s wireless mesh networks. In *Proceedings of the 2nd ACM workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec)*, pages 13–18, Budapest, Hungary, 2013.

[23] Jakob Eriksson, Sharad Agarwal, Paramvir Bahl, and Jitendra Padhye. Feasibility study of mesh networks for all-wireless offices. In *Proceedings of the 4th International Conference on Mobile Systems, Applications, And Services (MobiSys)*, pages 69–82, Uppsala, Sweden, 2006.

[24] Amin Farbod and Terence D. Todd. Resource allocation and outage control for solar-powered wlan mesh networks. *IEEE Transactions on Mobile Computing*, 6(8):960–970, 2007.

[25] Alexandros G. Fragkiadakis, Ioannis G. Askoxylakis, Elias Z. Tragos, and Christos V. Verikoukis. Ubiquitous robust communications for emergency response using multi-operator heterogeneous networks. *EURASIP Journal on Wireless Communications and Networking*, 2011:13, 2011.

[26] Laura Galluccio, Sergio Palazzo, and Andrew T. Campbell. Efficient data aggregation in wireless sensor networks: An entropy-driven analysis. In *Proceedings of the 19th IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, Cannes, French Riviera, France, 2008.

[27] Meijuan Gao, Fan Zhang, and Jingwen Tian. Environmental monitoring system with wireless mesh network based on embedded system. In *Proceedings of the Fifth IEEE International Symposium on Embedded Computing (SEC)*, pages 174–179, Beijing, China, 2008.

[28] S.M. George, Wei Zhou, H. Chenji, Myounggyu Won, Yong Oh Lee, A. Pazarloglou, R. Stoleru, and P. Barooah. DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Communications Magazine*, 48(3):128 –136, 2010.

[29] Stephan Glass, Vallipuram Muthukkumarasamy, and Marius Portmann. Detecting man-in-the-middle and wormhole attacks in wireless mesh networks. In

Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA), pages 530–538, Bradford, UK, 2009.

[30] Qijun Gu, Wanyu Zang, Meng Yu, and Peng Liu. Collaborative traffic-aware intrusion monitoring in multi-channel mesh networks. In *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 793–800, Liverpool, UK, 2012.

[31] Amin Hassanzadeh, Ala Altaweel, and Radu Stoleru. Traffic-and-resource-aware intrusion detection in wireless mesh networks. Technical report, Texas A&M University 2014-1-2, 2014.

[32] Amin Hassanzadeh and Babak Sadeghiyan. A data correlation method for anomaly detection systems using regression relations. In *Proceedings of International Conference on Future Information Networks (ICFIN)*, pages 242–248, Beijing, China, 2009.

[33] Amin Hassanzadeh and Radu Stoleru. Towards optimal monitoring in cooperative ids for resource constrained wireless networks. In *Proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, Maui, HI, USA, 2011.

[34] Amin Hassanzadeh and Radu Stoleru. On the optimality of cooperative intrusion detection for resource constrained wireless networks. *Computers & Security*, 34:16 – 35, 2013.

[35] Amin Hassanzadeh, Radu Stoleru, and Jianer Chen. Efficient flooding in wireless sensor networks secured with neighborhood keys. In *Proceedings of the 7th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 119–126, Shanghai, China, 2011.

[36] Amin Hassanzadeh, Radu Stoleru, Michalis Polychronakis, and Geoffery Xie. RAPID: A traffic-agnostic intrusion detection for resource-constrained wireless mesh networks. Technical report, Texas A&M University 2014-1-3, 2014.

[37] Amin Hassanzadeh, Radu Stoleru, and Basem Shihada. Energy efficient monitoring for intrusion detection in battery-powered wireless mesh networks. In *Proceedings of the 10th International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW)*, pages 44–57, Paderborn, Germany, 2011.

[38] Amin Hassanzadeh, Zhaoyan Xu, Radu Stoleru, and Guofei Gu. Practical intrusion detection in resource constrained wireless mesh networks. Technical report, Texas A&M University 2012-7-1, 2012.

[39] Amin Hassanzadeh, Zhaoyan Xu, Radu Stoleru, Guofei Gu, and Michalis Polychronakis. PRIDE: Practical intrusion detection in resource constrained wireless mesh networks. In *Proceedings of 15th International Conference on Information and Communications Security (ICICS)*, pages 213–228, Beijing, China, 2013.

[40] Guido R. Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann, and Bernhard Walke. IEEE 802.11s: the WLAN mesh standard. *IEEE Wireless Communications*, pages 104–111, Feb 2010.

[41] Yi-an Huang and Wenke Lee. A cooperative intrusion detection system for ad hoc networks. In *Workshop on Security of AdHoc and Sensor Networks (SASN)*, pages 135–147, Fairfax, VA, USA, 2003.

[42] Fabian Hugelshofer, Paul Smith, David Hutchison, and Nicholas J.P. Race. OpenLIDS: a lightweight intrusion detection system for wireless mesh networks. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 309–320, Beijing, China, 2009.

[43] Sahid Hussain, Abdul W. Matin, and Obidul Islam. Genetic algorithm for energy efficient clusters in wireless sensor networks. In *Proceedings of the Fourth International Conference on Information Technology (ITNG)*, pages 147–154, Las Vegas, NV, USA, 2007.

[44] Johnathan Ishmael, Sara Bury, Dimitrios Pezaros, and Nicholas Race. Deploying rural community wireless mesh networks. *IEEE Internet Computing*, 12(4):22–29, July 2008.

[45] David Johnson. Evaluation of a single radio rural mesh network in south africa. In *Proceedings of International Conference on Information and Communication Technologies and Development (ICTD)*, pages 1–9, Bangalore, India, 2007.

[46] Oleg Kachirski and Ratan Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS)*, pages 57 – 64, Hawaii, USA, 2003.

[47] Hyunwoo Kim, Dongwoo Kim, and Sehun Kim. Lifetime-enhancing selection of monitoring nodes for intrusion detection in mobile ad hoc networks. *AEU - International Journal of Electronics and Communications*, 60(3):248–250, March 2006.

[48] Mihui Kim, Varagur Karthik Sriram Iyer, and Peng Ning. MrFair: misbehavior-resistant fair scheduling in wireless mesh networks. *Ad Hoc Networks*, pages 299 – 316, 2012.

[49] Ioannis Krontiris, Zinaida Benenson, Thanassis Giannetsos, Felix C. Freiling, and Tassos Dimitriou. Cooperative intrusion detection in wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN)*, pages 263–278, Cork, Ireland, 2009.

[50] Adrian P. Lauf, Richard A. Peters, and William H. Robinson. A distributed intrusion detection system for resource-constrained devices in ad-hoc networks. *Ad Hoc Networks*, 8(3):253–266, 2010.

[51] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science (ACSC)*, pages 333–342, Newcastle, Australia, 2005.

[52] Guorui Li, Jingsha He, and Yingfang Fu. A distributed intrusion detection scheme for wireless sensor networks. In *Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS)*, pages 309–314, Beijing, China, 2008.

[53] Libelium Comunicaciones Distribuidas S.L. *Meshlium Xtreme Technical Guide*, September 2011.

[54] Changlei Liu and Guohong Cao. Distributed monitoring and aggregation in wireless sensor networks. In *Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM)*, pages 2097–2105, San Diego, CA, USA, 2010.

[55] Hai Liu, Amiya Nayak, and Ivan Stojmenovi. Fault-tolerant algorithms/protocols in wireless sensor networks. In Subhas Chandra Misra, Isaac Woungang, and Sudip Misra, editors, *Guide to Wireless Sensor Networks*, Computer Communications and Networks, pages 261–291. Springer London, 2009.

[56] Xuanwen Luo, Ming Dong, and Yinlun Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70, Jan 2006.

[57] Chi Ma, Zhenghao Zhang, and Yuanyuan Yang. Battery-aware scheduling in wireless mesh networks. *Mobile Network Applications*, 13:228–241, 2008.

[58] Dwight Makaroff, Paul Smith, Nicholas J.P. Race, and David Hutchison. Intrusion detection systems for community wireless mesh networks. In *Proceedings of the 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 610–616, Atlanta, GA, USA, 2008.

[59] Devu Manikantan Shila and Tricha Anjali. Load aware traffic engineering for mesh networks. *Computer Communications*, 31(7):1460–1469, 2008.

[60] Fabio Martignon, Stefano Paris, and Antonio Capone. A framework for detecting selfish misbehavior in wireless mesh community networks. In *Proceedings of the 5th ACM symposium on QoS and security for wireless and mobile networks (Q2SWinet)*, pages 65–72, Tenerife, Canary Islands, Spain, 2009.

[61] Peter Mell, Donald Marks, and Mark McLarnon. A denial-of-service resistant intrusion detection architecture. *Computer Networks*, pages 641–658, 2000.

[62] Anderson Morais and Ana Cavalli. A distributed and collaborative intrusion detection architecture for wireless mesh networks. *Mobile Networks and Applications - Springer*, 2013.

[63] Dries Naudts, Stefan Bouckaert, Johan Bergs, Abram Schouttcet, Chris Blondia, Ingrid Moerman, and Piet Demeester. A wireless mesh monitoring and planning tool for emergency services. In *Workshop on End-to-End Monitoring Techniques and Services (E2EMON)*, pages 1–6, Munich, Germany, 2007.

[64] Georgios Parissidis, Merkourios Karaliopoulos, Rainer Baumann, Thrasyvoulos Spyropoulos, and Bernhard Plattner. Routing metrics for wireless mesh net-

works. In *Guide to Wireless Mesh Networks*, Computer Communications and Networks, pages 199–230. Springer London, 2009.

[65] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, pages 2435–2463, 1999.

[66] Daler Rakhmatov and Sarma Vrudhula. Energy management for battery-powered embedded systems. *ACM Transactions on Embedded Computing Systems*, pages 277–324, August 2003.

[67] S.A. Razak, S.M. Furnell, N.L. Clarke, and P.J. Brooke. Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks. *Ad Hoc Networks*, 6(7):1151 – 1167, 2008.

[68] Leon Reznik, Gregory Von Pless, and Tayeb Al Karim. Intelligent protocols based on sensor signal change detection. In *Proceedings of Systems Communications*, pages 443–448, Montreal, Canada, 2005.

[69] Nikhil Saxena, Mieso Denko, and Dilip Banerji. A hierarchical architecture for detecting selfish behaviour in community wireless mesh networks. *Computer Communications*, 34(4):548 – 555, 2011.

[70] Vyas Sekar, Ravishankar Krishnaswamy, Anupam Gupta, and Michael K. Reiter. Network-wide deployment of intrusion detection and prevention systems. In *Proceedings of the 6th International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT)*, pages 18:1–18:12, Philadelphia, PA, USA, 2010.

[71] Sevil Sen, John A. Clark, and Juan E. Tapiador. Power-aware intrusion detection in mobile ad hoc networks. *Ad Hoc Networks*, 28:224–239, 2010.

[72] Sevil Sen and John Andrew Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *Proceedings of the Second ACM Conference on Wireless Network Security (WiSec)*, pages 95–102, Zurich, Switzerland, 2009.

[73] Devu M. Shila and Tricha Anjali. Defending selective forwarding attacks in WMNs. In *Proceedings of IEEE International Conference on Electro/Information Technology (EIT)*, pages 96–101, Ames, IA, USA, 2008.

[74] Devu M. Shila and Tricha Anjali. A game theoretic approach to gray hole attacks in wireless mesh networks. In *Proceedings of IEEE Military Communications Conference (MILCOM)*, pages 1–7, San Diego, CA, USA, 2008.

[75] Dong-Hoon Shin and Saurabh Bagchi. Optimal monitoring in multi-channel multi-radio wireless mesh networks. In *Proceedings of the Tenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 229–238, New Orleans, LA, USA, 2009.

[76] Dong-Hoon Shin, Saurabh Bagchi, and Chih-Chun Wang. Distributed online channel assignment toward optimal monitoring in multi-channel wireless networks. In *Proceedings of the 31st IEEE Conference on Computer Communications (INFOCOM)*, pages 2626–2630, Orlando, FL, USA, 2012.

[77] Peter W. Shor. A new proof of cayley's formula for counting labeled trees. *Journal of Combinatorial Theory, Series A*, 71(1):154 – 158, 1995.

[78] The Snort Project. *SNORT Users Manual v2.9.2*, September 2011.

[79] T. Srinivasan, V. Mahadevan, A. Meyyappan, A. Manikandan, M. Nivedita, and N. Pavithra. Hybrid agents for power-aware intrusion detection in highly mobile

ad hoc networks. In *Proceedings of International Conference on Systems and Networks Communications (ICSNC)*, pages 1 – 6, 2006.

[80] Brian Steckler, Bryan L. Bradford, and Steve Urrea. Hastily formed networks for complex humanitarian disasters after action report and lessons learned from the naval postgraduate schools response to hurricane katrina. Technical report, Naval Postgraduate School, 2005.

[81] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C-Y. Tseng, T. Bowen, K. Levitt, and J. Rowe. A general cooperative intrusion detection architecture for MANETs. In *Workshop on Information Assurance (IWIA)*, pages 57–70, College Park, MD, USA, 2005.

[82] Wei-Tsung Su, Ko-Ming Chang, and Yau-Hwang Kuo. eHIP: an energy-efficient hybrid intrusion prohibition system for cluster-based wireless sensor networks. *Computer Networks*, 51(4):1151–1168, 2007.

[83] Dhanant Subhadrabandhu, Saswati Sarkar, and Farooq Anjum. A framework for misuse detection in ad hoc networks-part I. *IEEE Journal on Selected Areas in Communications*, 24:274 – 289, Feb. 2006.

[84] Bo Sun, Kui Wu, and Udo W. Pooch. Alert aggregation in mobile ad hoc networks. In *Proceedings of the 2nd ACM workshop on Wireless Security (WiSe)*, pages 69–78, San Diego, CA, USA, 2003.

[85] Bo Sun, Kui Wu, and Udo W. Pooch. Zone-based intrusion detection system for mobile ad hoc networks. *International Journal of Ad Hoc & Sensor Wireless Networks (2006)*, 2(3), 2006.

[86] Ling-Yi Sun, Wei Cai, and Xian-Xiang Huang. Data aggregation scheme using neural networks in wireless sensor networks. In *Proceedings of 2nd International*

Conference on Future Computer and Communication (ICFCC), pages V1–725–V1–729, Wuhan, China, 2010.

[87] Suleyman Uludag, Tom Imboden, and Kemal Akkaya. A taxonomy and evaluation for developing 802.11-based wireless mesh network testbeds. *International Journal of Communication Systems*, 25(8):963–990, 2012.

[88] Marco Valero, Sang Shin Jung, A. Selcuk Uluagac, Yingshu Li, and Raheem A. Beyah. Di-Sec: A distributed security framework for heterogeneous wireless sensor networks. In *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM)*, pages 585–593, Orlando, FL, USA, 2012.

[89] Sudarshan Vasudevan, Brian DeCleene, Neil Immerman, Jim Kurose, and Don Towsley. Leader election algorithms for wireless ad hoc networks. In *Proceedings of DARPA Information Survivability Conference and Exposition*, pages 261–272, 2003.

[90] Bo Wang, Sohraab Soltani, Jonathan K. Shapiro, Pang ning Tan, and Matt Mutka. Distributed detection of selfish routing in wireless mesh networks. Technical report, Michigan State University, MSU-CSE-06-19, 2006.

[91] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 203–222, Sophia Antipolis, French Riviera, France, 2004.

[92] Daniel Wu, Dhruv Gupta, and Prasant Mohapatra. QuRiNet: A wide-area wireless mesh testbed for research and experimental evaluations. *Ad Hoc Networks*, 9(7):1221–1237, sep 2011.

[93] Fan Yang, V. Gondi, J.O. Hallstrom, Kuang-Ching Wang, G. Eidson, and C.J. Post. Wireless infrastructure for remote environmental monitoring: Deployment and evaluation. In *Proceedings of International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, pages 68–73, Montreal, Canada, 2013.

[94] Hao Yang, J. Shu, Xiaoqiao Meng, and Songwu Lu. SCAN: self-organized network-layer security in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):261–273, 2006.

[95] Ping Yi, Yiping Zhong, and Shiyong Zhang. A novel intrusion detection method for mobile ad hoc networks. In *Proceedings of Advances in Grid Computing (EGC)*, pages 1183–1192, Amsterdam, Netherlands, 2005.

[96] Wei Yu, Nan Zhang, Xinwen Fu, R. Bettati, and Wei Zhao. Localization attacks to internet threat monitors: Modeling and countermeasures. *IEEE Transactions on Computers*, 59(12):1655–1668, Dec 2010.

[97] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 275–283, Boston, MA, USA, 2000.

[98] Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, 2003.