UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

GUSTAVO ALONSO NUNEZ SEGURA

Cooperative Intrusion Detection for Software-Defined Resource-Constrained Networks

São Paulo

2022

GUSTAVO ALONSO NUNEZ SEGURA

Cooperative Intrusion Detection for Software-Defined Resource-Constrained Networks

Revised Version

Thesis presented to the Graduate Program in Electrical Engineering at Escola Politécnica da Universidade de São Paulo, to obtain the degree of Doctor in Science.

Advisor: Profa. Dra. Cíntia Borges Margi

Coadvisor: Profa. Dra. Arsenia Chorti

São Paulo

2022

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Segura, Gustavo Alonso Nunez
Cooperative Intrusion Detection for Software-Defined Resource Constrained-Networks / G. A. N. Segura -- versão corr. -- São Paulo, 2021.
120 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Redes de sensores sem fio 2.Redes definidas por software 3.Detecção de intrusos 4.Ataques de negação de serviço I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

NOME: Gustavo Alonso Nunez Segura

Título: Cooperative Intrusion Detection for Software-Defined Wireless Sensor Networks

Tese apresentada à Escola Politécnica da Universidade de São Paulo, para obtenção do título de Doutor em Ciências - Programa de Pós-Graduação em Engenharia Elétrica

Aprovado em:

Banca Examinadora:

Profa. Dra.: Instituição:	Cíntia Borges Margi Universidade de São Paulo
Julgamento:	_____
Prof. Dr.: Instituição:	Marcos Antonio Simplicio Junior Universidade de São Paulo
Julgamento:	_____
Prof. Dr.: Instituição:	Daniel Macêdo Batista Universidade de São Paulo
Julgamento:	_____
Prof. Dr.: Instituição:	Lukas Tobias Nepomuk Landau Pontifícia Universidade Católica do Rio de Janeiro
Julgamento:	_____
Prof. Dr.: Instituição:	Adrián Lara Universidad de Costa Rica
Julgamento:	_____

São Paulo
2022

ACKNOWLEDGEMENTS

I would first like to thank my supervisors, Professor Dr. Cíntia Borges Margi and Professor Dr. Arsenia Chorti, whose guidance was invaluable during the research formulation, questions and methodology. Their advices always helped me to bring my work to a higher level.

I would like to acknowledge my laboratory colleagues. Despite the last two years we were not physically at the SEMBEI, our whatsapp group was always the first place to discuss a new idea.

I would like to express my gratitude to Universidad de Costa Rica, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Finance Code 001 and ELIOT project (ANR-18-CE40-0030 / FAPESP 2018/12579-7) for their support and financial aid during this research.

Lastly, I would like to thank to my family, the one in Costa Rica and the new one I gained here in Brazil.

RESUMO

Segura, G.A.N. **Cooperative Intrusion Detection for Software-Defined Resource-Constrained Networks**. 2022. 120p. Tese (Doutorado) - Universidade de São Paulo, São Paulo, 2022.

Redes definidas por software são um paradigma que foi projetado para melhorar a programabilidade e o gerenciamento de redes. Os benefícios deste paradigma motivaram sua implementação em redes de sensores sem fio e aplicações da Internet das coisas, para abordar desafios relacionados a flexibilidade e reutilização de recursos. As redes definidas por software são vulneráveis a ataques de negação de serviço, e esta vulnerabilidade se torna crítica em redes com recursos limitados. Analisando o estado da arte, dois desafios principais foram identificados: escalabilidade e complexidade. Propostas com alto desempenho de detecção são em sua maioria abordagens centralizadas e precisam de recursos de comunicação incompatíveis com as limitaçoes das redes de sensores. Por exemplo, precisam de canais dedicados de comunicação para pacotes de controle e monitoramento constante em períodos curtos de tempo, o qual limita a escalabilidade das soluções. Existem propostas híbridas que reduzem o tráfego de pacotes e o gargalo das propostas centralizadas. Porém, estes trabalhos tem um desempenho menor que os centralizados ou precisam de dispositivos com características diferenciadas para aumentar o seu desempenho. Para abordar este desafio, foi projetada uma estratégia cooperativa de detecção de intrusos, onde todos os nós da rede participam ativamente. A proposta é composta por um detector de anomalias centralizado, mas ajustando os recursos de communicação de acordo com as características da rede. De forma paralela, todos os nós da rede estão monitorando seu próprio comportamento usando uma frequência de amostragem maior que a abordagem centralizada, para compensar o atraso na detecção. A detecção de intrusos é baseada na detecção de anomalias usando *change point* análise. A proposta é uma versão modificada de algoritmos de soma cumulativa modernos e é tão leve que roda em dispositivos TelosB, ocupando ao redor de 7.2 KB de memória. A proposta cooperativa foi simulada em redes com 36, 100 e 225 nós com um controlador só. Os resultados mostraram que, resolvendo o problema de complexidade na detecção distribuída é possível melhorar o desempenho na detecção sem reduzir o desempenho da rede quando o seu tamanho aumenta, tratando o aspecto da escalabilidade. A acurácia da detecção é comparável com outras propostas centralizadas que precisam de altas taxas de tráfego de pacotes ou dispositivos com características diferenciadas. Ainda, o sistema cooperativo permitiu a identificação de atacantes e tipo de ataque com probabilidades acima de 0.89.

Palavras-chave: Redes de sensores sem-fio. Redes definidas por software. Detecção de intrusos. Ataques de negação de serviço.

ABSTRACT

Segura, G.A.N. **Cooperative Intrusion Detection for Software-Defined Resource-Constrained Networks**. 2022. 120p. Thesis (Doctor) - Universidade de São Paulo, São Paulo, 2022.

Software-defined networking (SDN) is a paradigm that was meant to improve networks programmability and management facilities. These benefits motivated its implementation in Low-power and Lossy Networks (LLNs), such as Internet of Things and wireless sensor networks, to address challenges considering flexibility and resource reuse. SDN-based networks are vulnerable to denial of service (DoS) and Distibuted DoS (DDoS) attacks, and this vulnerability is critical in resource-constrained networks. Analyzing the state of the art for SDN-based LLNs, we identified two main challenges: scalability and complexity. Proposals with high detection performance are mainly centralized and require communication resources that are not compatible with LLNs, such as out-of-band communication and constant monitoring in short periods, restricting scalability. There are also hybrid proposals that reduced packets traffic and the bottleneck effect. These works reported inferior performance than centralized approaches or required specific nodes with high capabilities inside the LLN to support the detection. To address this gap, we propose a cooperative intrusion detection strategy where all the nodes have active participation. We use centralized monitoring to detect anomalies in the network behavior, adjusting the communication frequency to the network size and communication resources. At the same time, every LLN node is monitoring its behavior using a higher sampling frequency to compensate the delay of the detection from the centralized information. The intrusion detection is based on anomaly detection using change-point analysis. The algorithm proposed is a modified version of state-of-the-art CUSUM algorithms and is so lightweight that it can run on TelosB motes requiring around 7.2 KB of memory space only. The cooperative intrusion detection was simulated on networks with 36, 100 and 225 nodes with only one controller. The results showed that by solving the complexity issues of the distributed detection we were able to improve scalability without reducing detection and network performance, obtaining detection accuracy comparable to high-traffic centralized approaches without the need of high capabilities devices. Moreover, the cooperation among the nodes allowed us to identify nodes launching the attack and the type of the attack with a probability exceeding 0.89

Keywords: Wireless Sensor Networks. Software-Defined Networking. Intrusion Detection. denial of service attack.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CPD	Change Point Detection
CUSUM	Cumulative Sum
DT	Decision Tree
DoS	Denial of Service
DR	Detection Rate
DTM	Detection Time Median
DDoS	Distributed Denial of Service
ETX	Expected Transmission Count
FDFF	False Data Flow Forwarding
FFR	False Flow Request
FNR	False Negative Rate
FNI	False Neighbor Information
FPR	False Positive Rate
ID	Identifier
IT-SDN	Improved TinySDN
IETF	Internet Engineering Task Force
IoT	Internet of Things
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IRTF	Internet Research Task Force
K-NN	K-Nearest Neighbors
LLNs	Low-Power and Lossy Networks
MCU	Microcontroller
NFV	Network Function Virtualization

NSAL	Network Services Abstraction Layer
NN	Neural Networks
OS	Operating System
RF	Random Forest
RFC	Request for Comments
RPL	Routing Protocol for Low Power and Lossy Networks
SOM	Self-Organizing Maps
SDN	Software-Defined Networking
SDWSN	Software-Defined Wireless Sensor Networks
SVM	Support Vector Machine
TCP	Transmission Control Protocol
WSN	Wireless Sensor Networks

CONTENTS

1 INTRODUCTION

Software-defined networking (SDN) is a paradigm of programmable networks devised to control, change and manage network behavior dynamically. For this reason, SDN proposes to separate the control plane and the data plane and centralize the control and routing decisions in a software-based controller, or controllers, with a global view of the network. Then, the network devices become simple forwarding devices that are configured by the controllers through an application programming interface (Kreutz et al., 2015).

Back in the day, SDN first proposals were motivated by the inflexibility and low performance of the software platforms for programmable networks (MCKEOWN et al., 2008). Current results in the literature have shown that SDN can improve the programmability and simplify the administration and management of networks (BENZEKKI; FERGOUGUI; ELALAOUI, 2016; Rawat; Reddy, 2017).

Low-Power and Lossy Networks (LLNs) is a group of networks composed of embedded devices with processing, memory, energy and bandwidth limitations. This group of networks includes wireless sensor networks (WSN) and Internet of things (IoT) applications. WSNs are considered application-specific, which results in infrastructure underutilization, e.g., multiple WSNs for multiple applications deployed in the same physical space instead of one that is able to attain the same objectives. The implementation of SDN in LLNs has been explored as a solution to address flexibility and resource reuse challenges, but not limited to them. Kobo, Abu-Mahfouz and Hancke (2017) discuss the importance of SDN to address energy, scalability, mobility and interoperability challenges in WSN.

First SDN proposals for LLNs were architecture ideas that step by step became functional, and today there are protocols and frameworks (Alves et al., 2019) showing performance comparable to RPL (ALEXANDER et al., 2012), an established standard for LLNs. However, there are still challenges in wired SDN networks that are critical in LLNs, such as security (Pritchard; Hancke; Abu-Mahfouz, 2017). The planes separation and the global view of the network in SDN are characteristics that can help to countermeasure particular types of attacks by configuring the whole network to isolate malicious nodes, for example. On the other hand, the control plane centralization turns the network prone to denial of service attacks (DoS) and distributed DoS (DDoS). Since SDNs are entirely controller-based, this vulnerability compromises the whole network as well.

In this work, we study the main security vulnerabilities of SDN in LLNs, analyze the impact of **DoS** and **DDoS** attacks in the network performance and propose a solution

to detect them considering LLNs restrictions.

1.1 Motivation

First security proposals for SDN-based networks were totally dependent on the controller (SCOTT-HAYWARD; O'CALLAGHAN; SEZER, 2013), mostly based on centralized monitoring leveraged by the global view of the network. However, there were already some authors proposing to delegate some control back to network devices since the controller could become a bottleneck and is the most vulnerable element of the network (NAOUS et al., 2009; NAYAK et al., 2009). Using network devices as part of the attack detection aids to keep malicious traffic as much as possible in the data plane, reducing the chances to attain the controller.

Centralized approaches are effective for DoS and DDoS attacks detection and could use machine learning (ML) techniques to detect anomalies on the packets traffic. These approaches need constant monitoring and high processing resources to run the detection algorithms, consequently demanding requirements as out-of-band and wired connections between switches and controllers (Bhunia; Gurusamy, 2017; Ravi; Shalinie, 2020; Jia et al., 2020). These conditions limit the LLNs scenarios where these proposals could be applied.

One approach to reduce packets traffic in centralized DoS and DDoS detection is by delegating part of the responsibility to the network devices, i.e., the traffic behavior passing through them and their neighbors to detect the attack. There are hybrid proposals (i.e., centralized and distributed monitoring) in the literature that considered LLNs resources constraints but reported inferior performance than centralized approaches or required specific nodes with high capabilities inside the network to support the detection. The main problem of the first approach is that high control packets traffic compromise scalability, and in the case of less complex proposals, the main problem is the accuracy of the detection. *An effective DoS attack detection strategy for SDN-based LLNs networks that does not compromise scalability is still a gap in the literature.*

1.2 Hypothesis

We believe that a hybrid DoS detection is the right approach to solve the complexity and scalability challenges since the centralized and distributed monitoring provides diverse information and could cooperate to reduce resource usage. However, hybrid approaches in the literature have not presented strategies to maintain high detection performance without requiring high capabilities in the LLNs devices. Our belief is that, assuming a secure channel in the whole network, this can be solved following the next steps:

- the centralized communication frequency for security monitoring has to be defined according to the network size and performance metrics to improve scalability;

- the distributed detection has to be able to compensate the delay in the centralized monitoring in large networks;
- single nodes should be able to detect anomalies in the network by monitoring its own behavior instead of monitoring their neighbors. This could reduce resources usage.

1.3 Objectives

Our main objective is to design and evaluate a hybrid intrusion detector for DoS attacks in software-defined resource-constrained networks where all the elements cooperate to detect the attack and identify the attackers in large networks. The main objective is divided into the following specific goals:

- design and evaluate a centralized DoS attack detector for SDN-based resource constrained networks and evaluate its detection performance;
- design and evaluate a lightweight behavior anomaly detector for LLNs devices and evaluate its detection performance;
- design a centralized security manager for coordinating the cooperation among the elements of the network and taking security-wise decisions;
- evaluate the performance of the hybrid intrusion detector proposal in large networks

The reference to determine the size of a large network is based on works that studied scalability of SDN-based resource constrained networks. This topic is further discussed in Chapter 4.

1.4 Methods

First, the detection performance of software-defined wireless sensor networks (SD-WSN) under DoS attacks was analyzed (SEGURA; MARGI; CHORTI, 2019). For this, three different types of attacks were implemented: two of them were new-flow type and the other one based on the integrity of routing information. A detailed description of these attacks is discussed in Chapter 4. From this analysis, network vulnerabilities and the metrics where these attacks have a stronger impact were analyzed.

The performance metrics considered were: data and control packets delivery rate, data and control packets end-to-end delay, control packets overhead and energy consumption. The delivery rate is calculated by dividing the total number of packets successfully received by the total packets sent. The end-to-end delay is the average of the time the packets spent to reach the destination. The packets overhead is quantified as the total

amount of packets sent. The energy consumption is the average energy consumption of all nodes, excluding the sink and the controller nodes.

Next, the anomaly detection algorithm was implemented. We chose to use change point analysis since it fulfills the lightweight criteria and the previous evaluation (SEGURA et al., 2020) showed detection performance results similar to other works in the literature using ML techniques, among others. However, our proposal is independent of the anomaly detection technique while it is lightweight enough to fit and run in simple WSN or IoT devices.

Both centralized and distributed anomaly detection performance were evaluated based on detection rate (DR), false positive rate (FPR), false negative rate (FNR) and detection time median (DTM), indicating the median of the time instances elapsed from the launch of the attack to the instance it was identified. The experiments included simulations on fully-bidirectional grid topologies with 36 and 100 nodes (SEGURA; MARGI; CHORTI, 2020; SEGURA; MARGI; CHORTI, 2021).

Lastly, the scalability of our proposal was evaluated through simulations on 225-node fully-bidirectional grid topologies (SEGURA; CHORTI; MARGI, 2021). This number was based on state-of-the-art results about the scalability of SDWSN using only one controller (Alves et al., 2019), where it was shown that for topologies over 200 nodes the delivery rate falls under 90%. The timeline through these steps is depicted in Figure 1.

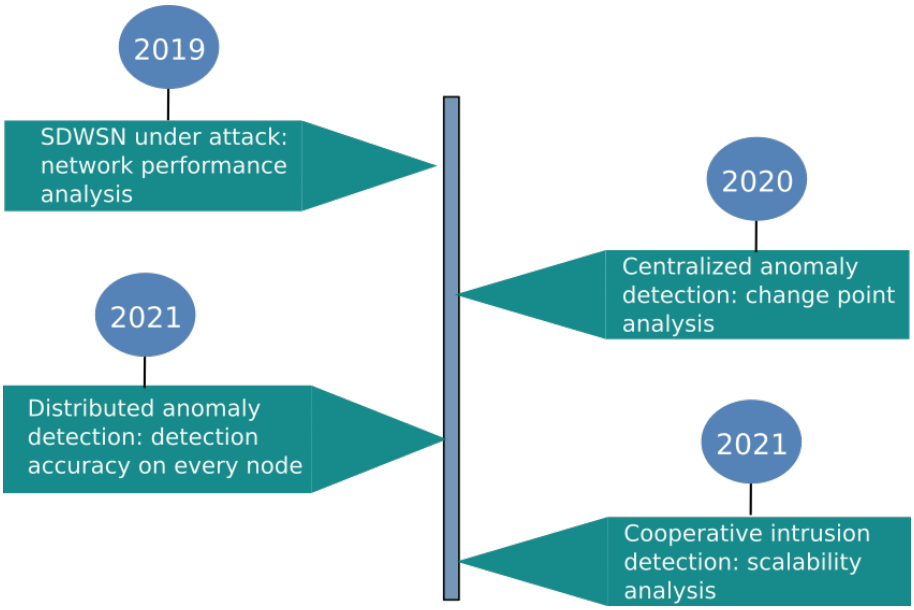


Figure 1 – Methods timeline

The experiments consisted of simulations using COOJA simulator and emulating TelosB motes. The SDWSN was implemented using IT-SDN (Alves et al., 2019) and Contiki-OS 3.0 (DUNKELS; GRONVALL; VOIGT, 2004).

1.5 Organization

This thesis is organized as follows:

- Chapter 2, Theoretical Background, explains the basic concepts related to SDN, LLNs, and security in SDN;
- Chapter 3, State of the Art, analyzes relevant research concerning DoS attack detection in SDN-based resource-constrained networks;
- Chapter 4, Problem Formulation, analyzes the impact of DoS attacks on the network performance and explains the scalability and complexity problem SDN-based resource-constrained networks;
- Chapter 5, Cooperative Intrusion Detection, describes the hybrid intrusion detection proposal;
- Chapter 6, Performance Evaluation, presents and discusses the most relevant results about the cooperative intrusion detection proposal;
- Chapter 7, Final remarks, includes the conclusions, future work and publications.

2 THEORETICAL BACKGROUND

This chapter presents basic definitions and information that are important throughout this document. First, Section 2.1 explains SDN concept and operation. Then, Section 2.2 presents main concepts about resource-constrained wireless networks and their implementation using SDN. Lastly, Section 2.3 studies SDN security vulnerabilities.

2.1 Software-Defined Networking

There are three main planes in computer networks: data, control, and management planes. The data plane involves all functions and processes to forward data from applications being executed. The control plane takes routing decisions and configures them. The management plane ensures the optimal operation of the network through control and monitoring functions.

SDN is a paradigm that decouples control and forwarding functions. This separation is meant to increase network programmability and improve management. Unlike traditional networks, where the control and data planes are bundled inside the network devices, in SDN the control plane is logically-centralized in one or multiple controllers. These controllers are in charge of taking routing decisions and have control over the data plane elements, i.e. network devices, through an application programming interface. Consequently, the network devices become simple forwarding elements with routing tables handling rules, commonly named as flow tables. These rules specify the action, or actions, to perform when there is a match between a flow entry in the table and the flow identifier in the incoming packet. The decision about the actions configured in each device depends on routing policies and topology information. Examples of these actions include: dropping, forwarding, and receiving.

The controller needs topology information to be able to calculate and configure routing rules and the network devices need to know how to reach the controller to provide these information. In the case where network devices are not directly connected to the controller, an auxiliary controller discovery protocol is used. In parallel, a neighbor discovery protocol allows network devices to obtain neighborhood information, such as addresses and link quality metrics. Then, the network devices send this information to the controller. This could be periodically or when there is a significant change in the topology or routing metrics.

Some authors propose the management plane on top of the control plane (Kreutz et al., 2015) while other authors propose the management plane should be directly connected to the control plane, data plane and an application plane (CABAJ et al., 2014). In 2015,

the Internet Research Task Force (IRTF) published the RFC 7426 to define SDN layers and architecture terminology, since as they claimed, there was an increasing confusion as to what SDN is. According to the RFC 7426, and as shown in Figure 2, the application plane is at the top. Then, the control plane and the management plane are at the same level and the data plane (i.e. Device and Resource Abstraction Layer) is at the bottom. The application plane is a collection of applications and services that define the network behavior, excluding applications to support the forwarding plane that are part of the control plane. Also, these applications and services may use services from the control and the management planes.

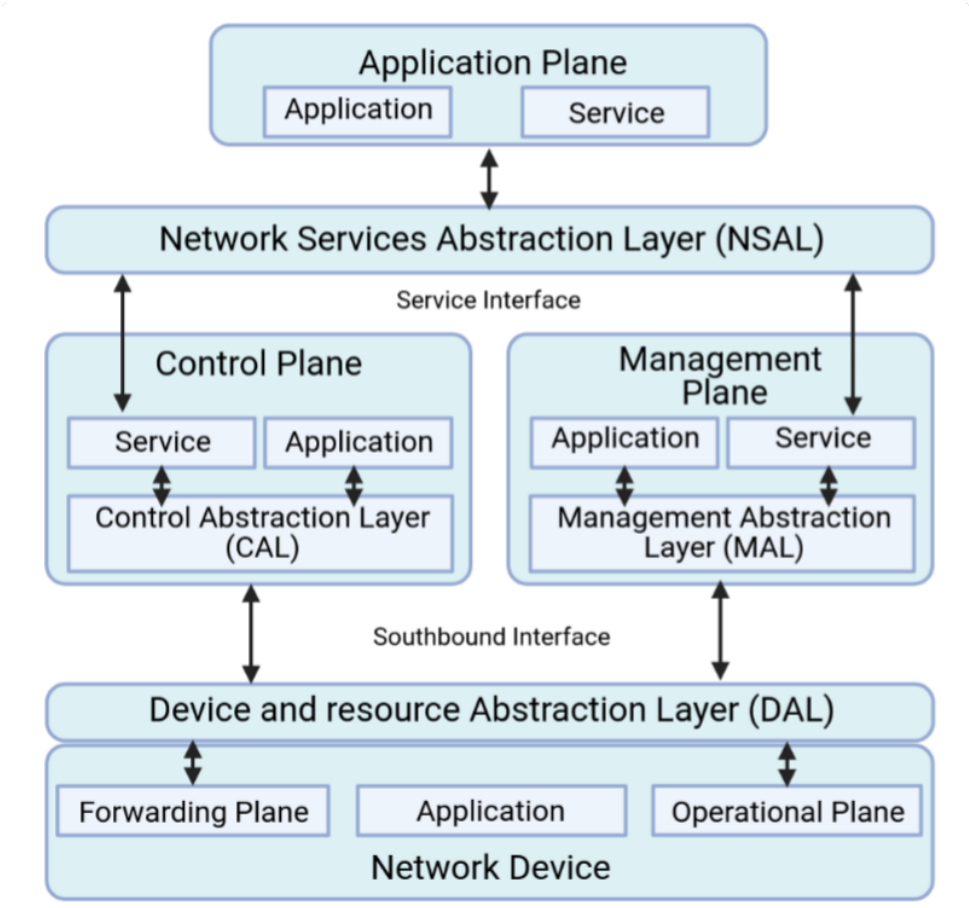


Figure 2 – SDN architecture according to RFC 7426

Source: Modified from Haleplidis et al. (2015)

The management plane is in charge of ensuring the optimal operation of the network. Such as the control plane, the management plane usually has a global view of the network, thus based on global or individual metrics the management plane is able to detect faulty or non-optimized operation (LUZ; MARGI; VERDI, 2021). Some metrics could be high delay, low delivery rate or high energy consumption. In some cases, the actions required to optimize the operation may need the intervention of the control plane to configure new routing rules.

The northbound interfaces are implemented in the Network Services Abstraction Layer (NSAL) and allow the communication among the control, management and application planes. Examples of service interfaces include, but are not limited to, RESTful APIs and open protocols such as NETCONF (ENNS et al., 2011). The southbound interfaces are implemented in the abstraction layer of the planes and allow the communication between the control and management plane with the network devices. OpenFlow (MCKEOWN et al., 2008) is the most commonly used southbound protocol and is considered as a standard in SDN (LATIF et al., 2020). Some other southbound protocols are: FORwarding & Control Element Separation, Open virtual Switch Database, Protocol Oblivious Forwarding, OpFlex, and OpenState (LATIF et al., 2020).

2.2 Low-Power and Lossy Networks

LLNs are composed of embedded devices with processing, memory, energy and bandwidth limitations. These group of networks include WSN and IoT applications. To understand the resource limitations in these devices, Table 1 compares some WSN and IoT platforms with the Raspberry Pi 3, a small single-board computer. The RE-Mote has better capabilities than the traditional TelosB and SensorTag devices, but in comparison with the Raspberry Pi 3, the last one has 32 times more clock speed and more than 30 000 times more RAM than the RE-Mote. Also, none of the WSN and IoT devices has 64-bit register such as the Raspberry Pi 3.

Table 1 – WSN and IoT devices specifications

Platform	**Microprocessor model**	**Clock speed (MHz)**	**Flash memory**	**RAM**	**Register size**
TelosB	MSP430	8	48 KB	10 KB	16-bit
SensorTag	ARM Cortex-M3	48	128 KB	20 KB	32-bit
RE-Mote	ARM Cortex-M3	32	512 KB	32 KB	32-bit
Raspberry Pi 3	4 x ARM Cortex-A53	1200	SD card	1 GB	64-bit

Source: Author

WSN are multi-hop/ad-hoc networks deployed to collect information from the environment and communicate it to specific nodes, known as sink nodes. Then, sink nodes communicate this information to the application servers or directly to the user. Commonly, this last part does not belong to the resource-constrained environment. These networks are used for monitoring hostile locations or whenever wiring is complicated or expensive. For this reason, they are found in a wide spectrum of applications, from military and industrial monitoring to home automation. Also, complex applications may consist of hundred or thousand of nodes.

Three communication patterns are found in WSNs: many-to-one, one-to-many, and many-to-many. The many-to-one prevails, since it is the case when many nodes send

the information sensed to a sink. The one-to-many case is commonly used to change the configuration of the sensor nodes, such as firmware or application parameters. In this case, the sink would work as a gateway between a management or controller entity and the sensor nodes. The many-to-many pattern is present in the networks with multiple sinks or peer-to-peer communication. In this case, one sensor node is able to send monitoring information to different destinations.

WSN and IoT use lightweight communication protocols specifically designed for their resource constraints, such as IEEE 802.15.4 (GUTIERREZ; CALLAWAY; BARRETT, 2003), Bluetooth Low Energy (BLE) (WOOLLEY, 2019), IEEE 802.11ah (BANOS-GONZALEZ et al., 2016) and LoRA (SORNIN et al., 2015). These protocols encompass lower layers, i.e. PHY and MAC, and their use depends on the application requirements. LoRA was designed for long distances while IEEE 802.15.4 and BLE are better for short distances. In terms of traffic intensity, 802.11 ah is optimal for large data sizes (MORIN et al., 2017).

The diversity of scenarios and applications with different service requirements incurred in the design of different routing protocols, which lead to a lack of standardization (WATTEYNE et al., 2011). This motivated the Internet Engineering Task Force (IETF) to specify a set of IPv6 protocols tailored for resource-constrained devices. One of these protocols was RPL (ALEXANDER et al., 2012), a distant vector routing protocol which became one of the most common used in IoT (KHARRUFA; AL-KASHOASH; KEMP, 2019; MARDINI; EBRAHIM; AL-RUDAINI, 2017; HASHEMI; ALIEE, 2019). Despite the popularity of RPL and their advances in routing standardization, there are still performance issues mainly concerning energy efficiency, mobility and security (KHARRUFA; AL-KASHOASH; KEMP, 2019).

WSN also faces challenges in terms of network management and resources utilization. WSNs are considered application specific, which results in infrastructure underutilization, for example, multiple WSNs for multiple applications deployed in the same physical space instead of one that is able to attain the same objectives. The lack of flexibility in management services and the environmental conditions in where these networks are deployed complicate their configuration, provisioning, maintenance and resource sharing (Kobo; Abu-Mahfouz; Hancke, 2017).

Such as mentioned in Section 2.1, SDN is a paradigm that aims to increase the programability and improve the management of networks. The implementation of the SDN paradigm in WSN and IoT applications has been studied as a solution to the routing and management issues in these networks, among others such as scalability, interoperability and energy efficiency. Up to this day there are promising results but new challenges to face. Subsection 2.2.1 presents more details about the Software-Defined Wireless Sensor Networks (SDWSN) concept.

2.2.1 Software-Defined Wireless Sensor Networks

Software-Defined Wireless Sensor Networks (SDWSN) is the common name used in the literature for the implementation of the SDN paradigm in WSN. How SDN is implemented vary among the authors, but a common approach is to use a southbound protocol to communicate the SDN controller with the WSN devices. Since usually the SDN controller is not a resource-constrained device, at least one WSN node should work as an interface. In this manner, the controller receives packets from the WSN and disseminates the configuration information.

The first initiatives to apply SDN to WSN and IoT are from 2011 and 2012 (MAHMUD; RAHMANI, 2011; COSTANZO et al., 2012; LUO; TAN; QUEK, 2012). Mahmud and Rahmani (2011) proposed to use SDN to improve reliability, routing performance and load balancing. Their simulation experiments showed a reduction in the packets traffic. Then, Costanzo et al. (2012) explored the implementation of SDN in WSN. An architecture was proposed by Costanzo et al. (2012), following three main requirements: support to duty-cycling, support to in-network data aggregation and flexible definitions of rules. Luo, Tan and Quek (2012) proposed preliminary solutions to port OpenFlow to WSNs. These works are based on OpenFlow (MCKEOWN et al., 2008), which is not adequate to the constraints imposed in WSN, such as limited frame sizes, memory constraints, and lack of dedicated control channel, which was reflected in the lack of simulations and experiments with real devices. OpenFlow is Transmission Control Protocol (TCP) based and TCP is able to receive segments up to 65K bytes. Comparing IEEE 802.3 (MTU of 1500 bytes) and IEEE 802.15.4 (MTU of 127 bytes) interfaces, using TCP in IEEE 802.15.4 would require 12 times more packets to reach the maximum segment size. Only Mahmud and Rahmani (2011) presented simulations results without specifying physical and link layer protocols or standards.

It took a few years to finally have complete proposals that considered resource constraints and were able to show performance results and experiments in real devices. In 2014, Oliveira, Margi and Gabriel (2014) proposed TinySDN and in 2015 Galluccio et al. (2015) proposed SDN-WISE. These frameworks are based on operating systems for WSN (TinyOS (LEVIS et al., 2005) and ContikiOS (DUNKELS; GRONVALL; VOIGT, 2004)), which allowed to test them on IEEE 802.15.4 compliant platforms, such as TelosB and EMB-Z2530PA. It is worth mentioning that TinySDN enables multiple controllers while SDN-WISE is based on SDWN (COSTANZO et al., 2012) architecture and focus on implementation and experiments. However, only small networks were used to evaluate their performances.

Coral-SDN (THEODOROU; MAMATAS, 2017), μSDN (BADDELEY et al., 2018) and IT-SDN (ALVES et al., 2017) are more recent proposals. Coral-SDN is an OpenFlow-like protocol intended to address mobility issues in SDWSNs through adaptive topology

control. According to Theodorou et al. (2019), Coral-SDN increases packet delivery rate and reduces control overhead in networks with mobile nodes. The proposal was evaluated in a 21-node network, with five mobile nodes. μSDN is a SDWSN framework that aims to address SDN control overhead issues in WSNs to increase scalability. To attain their objective, the authors proposed to eliminate fragmentation, reduce control packets frequency and use source routing, among other secondary optimization measures. Their results showed a significant reduction of control packets if compared to RPL. However, their experiments were limited to 30-node topologies, which limits the conclusions in terms of scalability.

TinySDN was a groundbreaking work, but its dependency on TinyOS and the use of active messaging limited its operation with other systems. IT-SDN is based on TinySDN concepts but it aimed to improve interoperability and performance. IT-SDN underlying architecture is independent of the operating system and also defines a clear separation among the southbound, neighbor discovery and controller discovery protocols. This separation simplifies the integration of new protocols that could improve network performance and interoperability. Simulation experiments (Alves et al., 2019) showed that IT-SDN is feasible for WSN even in networks over 200 nodes, performing as well as RPL. Also, it was evaluated on a small deployment using TelosB motes.

The experiments presented in this thesis were conducted using IT-SDN. For this reason, its architecture and protocols are explained in Subsection 2.2.2.

2.2.2 IT-SDN

As shown in Figure 3 within the dashed rectangle, IT-SDN framework (Alves et al., 2019) comprises the sensing layer, the control layer and three communication protocols: the Southbound protocol, the Neighbor Discovery protocol and the Controller Discovery protocol. The sensing layer is composed of the wireless sensor devices used to collect data from the environment and relay data to the sinks, while routing decisions are taken in the control layer. It is worth mentioning that all sensor nodes should be SDN-enabled.

The neighbor and controller discovery protocols are not interwined with the specification, which enables the implementation and replacement of them according to the application or scenario. The current IT-SDN implementation has been evaluated with different protocols (ALVES; MARGI; KUIPERS, 2020). The Southbound protocol defines the message formats for communication between the WSN and the SDN controller. In IT-SDN, this protocol is composed of seven packets: flow request, flow setup, flow ID register, neighbor report, data packet, acknowledgement, and underlying protocols packet.

The sensor nodes use the flow request packet to query the controller about an unknown route, for example, when the incoming packet has a flow identifier with no match in the flow table. The controller replies to the sensor nodes using the flow setup packet. This packet contains the information required to configure the routes. Moreover,

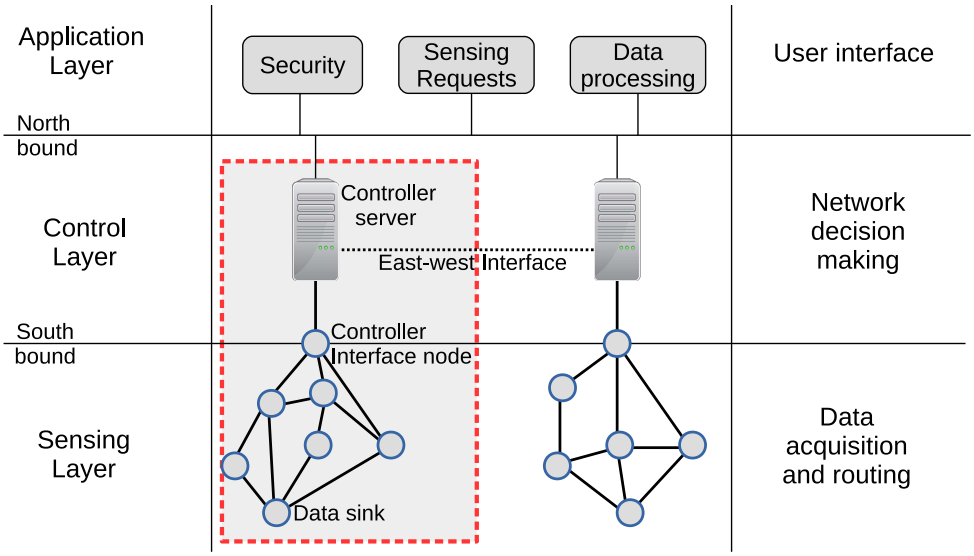


Figure 3 – IT-SDN architecture

Source: Alves et al. (2019)

Flow ID	Next hop	Action	# used	age
0	0x1111	forward	10	2
10	--	drop	5	1

Figure 4 – Flow table in IT-SDN SDN-enabled nodes

Source: Author

the controller could change already configured entries using such packet. SDN-enabled nodes use the flowid register to inform the controller that they are a destination candidate for a specific flow id. Sink nodes are common users of them. The neighbor report packet contains the source neighborhood information, which is used by the controller to update the network graph. The nodes send a neighbor report to the controller if: (i) the node detects one or more new neighbors, (ii) the node detects one or more nodes are no longer its neighbors, or, (iii) there is a significant change in one or more neighbors' routing metric. Data packets encapsulate application layer packets and the Acknowledgement confirms the delivery of control packets. Lastly, the neighbor and controller discovery protocols packets are encapsulated using the underlying protocols packet.

All SDN-enabled nodes have a flow table, such as the one in Figure 4. The "Flow ID" column is the matching criteria. The "Next hop" column contains the address of the next hop to reach the final destination. The "Action" column contains the action to handle the incoming packet. The "# used" column indicates the times this flow was used, which means the incoming packets with this flow identifier. Then, "age" column represents the times the information in the table for this flow was updated.

IT-SDN also supports source routed. In this case, the whole route is within the

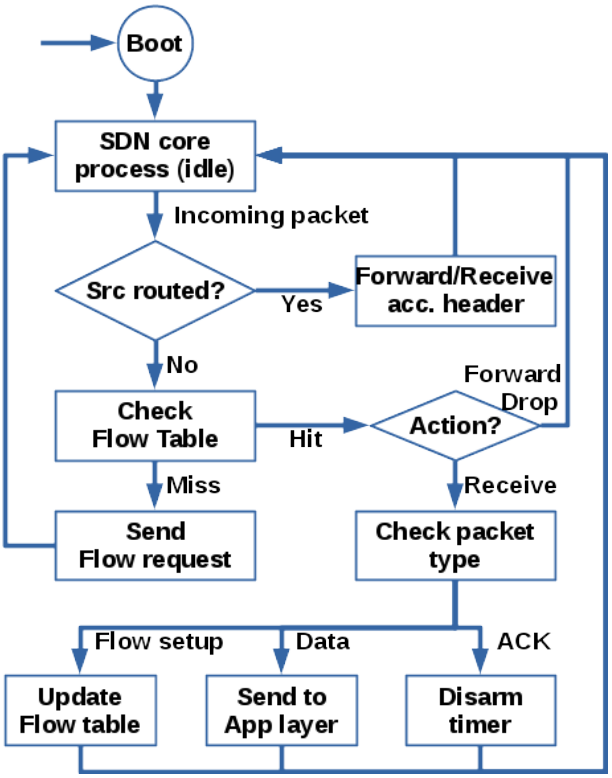


Figure 5 – IT-SDN node behavior

Source: Alves et al. (2019)

packet. Source routing is one option to reduce the requirement of flow table sizes. For example, the flow setup packets from the controller to a specific node are source routed since each node would need a specific flow to reach it.

In this manner, when the SDN core in the node detects an incoming packet, the next step is to check the type of the packet. If this is source routed, the node forwards it according to the route in the header or receives it if its address matches the final destination address. If the packet is flow routed, the node executes the action in the flow table. If there is no matching criteria, the packet is stored and a flow request is sent to the controller. The node waits the flow setup packet to configure the new rule and execute the action for the packet stored. Figure 5 summarizes the node behavior.

IT-SDN specifies the controller communication but does not specify internal procedures and policies, such as to decide between reactive or proactive routing configuration or which metric and routing algorithm use. In terms of implementation, current version available [1] is based on Contiki Operating System (DUNKELS; GRONVALL; VOIGT, 2004) with a custom controller implementation for x86 machines. IT-SDN has been compiled and tested on TelosB (Alves et al., 2019), Zolertia z1 (SAYJARI; SILVEIRA; MARGI, 2021) and CC2650 platforms.

[1] http://www.larc.usp.br/users/cbmargi/www/it-sdn/. Accessed: November 20th 2021

2.3 Security in SDN

In terms of security, SDNs have advantages and disadvantages. The control centralization, the controller global view, and the data and control planes separation are characteristics that inspired new security strategies (AHMAD et al., 2015). Braga, Mota and Passito (2010) proposed a mechanism based on Self-Organizing Maps (SOM) (KOHONEN, 1990) to find hidden relations among flows entering the network to detect anomalies in the network traffic. Giotis, Androulidakis and Maglaris (2014) improved legacy Remote Triggered Black-Hole (RTBH) routing approach for DDoS attack mitigation by filtering malicious traffic on a per-flow level. On the other hand, SDNs are entirely controller-based. This means that, if the controller is compromised, the control plane is compromised, therefore, the whole network is compromised as well. For this reason, the controller is tagged as a single point of failure, which turns SDN-based networks prone to DoS attacks (SINGH; BHANDARI, 2020; Rawat; Reddy, 2017).

In SDNs, the attackers can reach the control plane directly through a controller or from the applications and network devices through the northbound and southbound interfaces. The lack of standardized northbound protocols can lead to use insecure communication protocols that could be exploited by malicious applications to reach the controller (LATIF et al., 2020). For example, malicious applications on top of the controller execute a service interruption by manipulating packet handlers or network devices manipulate neighborhood information to mislead the controller.

In SDN networks, the controller takes the routing decisions and configures the routing rules on the network devices. When a network device is handling an incoming packet and there is no matching rule for it, the network device requests a rule to handle this packet to the controller. The controller calculates the rule and sends a reply to configure the device. Malicious nodes exploit this message exchange to attack the network in different ways. An attacker in the forwarding plane can flood the network with control packets targeting the controller, for example, querying routing rules. Aggressive attacks can exhaust the controller processing and communication resources. In the same way, an attacker can mislead other nodes in the network, inducing them to send packets to the controller at the same time. For example, an attacker sends several data packets tagged with an unknown flow identifier. The neighboring nodes receiving the packet will check on their routing table looking for a rule but without success, therefore the nodes will request a flow rule from the controller. These types of attacks are known as new-flow attacks and they impact both the controller and network devices resources, compromising the entire network.

New-flow attacks could be executed by new nodes, unknown by the controller, or by known nodes that were tampered. In the case the network had an authentication system, the controller or a Security application could detect an unknown node attacking

the network quickly, but if not, it is necessary another detection mechanism, for example, anomaly detection. During the time the attacker is active, the nodes in the route to reach the controller will spend resources forwarding the packets sent by the attacker, and the controller will spend resources calculating the rules and replying to the requests. Usually, this attack is executed by multiple attackers, which means that more nodes in the network will be affected and the resources usage increase. However, this condition allows the detection of anomalies on the network behavior.

In the case of wireless networks with resource constrained devices, this scenario is critical. In these networks, the devices have less resources to execute complex security algorithms and to deal with saturation attacks. Also, the in-band communication leads to undesirable consequences, such as an unreliable and less secure control plane (THEODOROU; MAMATAS, 2020).

To illustrate this concept, we conducted some experiments to measure the impact of a new-flow attack on a software-defined wireless sensor network (SDWSN) (SEGURA; MARGI; CHORTI, 2019). The main results showed this attack was able to increase the number of control packets per minute between 16% and 127% when there is only one attacker in the network, and the impact increases when increasing the number of attackers. After several repetitions, this attack saturates the neighbors' routing tables and packets buffer pool, disturbing packets transmission. Also, routing tables saturation inhibits the node to store new routing rules that could be necessary for the network normal operation. We also studied the case when malicious nodes manipulate neighborhood information to mislead the controller. We observed a significant impact on the control and data packets delivery rate. An extended analysis of these results is presented in Chapter 4. It is worth mentioning that we did not focus on running aggressive attacks since our objective was to evaluate the SDN architecture and IT-SDN framework vulnerability through the analysis of network metrics. Triggering the attack every sixty seconds was enough to observe these results.

2.3.1 Intrusion detection

Applying intrusion detection systems (IDS) is one approach to solve cyber-attacks, including virus, worms and DDoS (SCARFONE; MELL et al., 2007). IDSs are meant to detect threats from the analysis of the network behavior and have been implemented using different techniques, such as statistical methods and machine learning.

Intrusion detection techniques are classified into three categories: signature-based, anomaly-based and specification-based (ZARPELÃO et al., 2017). The signature-based approach detects attacks that match with patterns or signatures previously stored. This means this approach requires detailed information about the attack behavior to operate correctly. These patterns and signatures could be a specific message exchange or a specific

payload in a message (Butun; Morgera; Sankar, 2014). Signature-based approaches could be very effective at detecting known threats but have a poor performance with new attacks, such as zero-day attacks.

The anomaly-based approaches compare the monitored system against normal behavior profiles. Whenever a significant difference is detected, an alert is triggered. To obtain normal behavior profiles, they run training routines to obtain specific parameters and be able to compare behaviors. This detection method is effective against new attacks but could have high false positive rates (ZARPELÃO et al., 2017). The type of metrics monitored by these detection schemes is diverse. For example, some proposals monitor energy consumption (LEE et al., 2014) or number of control packets and neighbors (PONGLE; CHAVAN, 2015). Other proposals monitor more specific metrics, as packet size or data rate from one-hop neighbor packets (Thanigaivelan et al., 2016).

The specification-based approaches have elements of both signature-based and anomaly based approaches. The specification-based detection monitors the system behavior, such as the anomaly-based, but instead of running a training routine, the rules of each specification are manually defined, similar to signature-based detection. These characteristics aid to obtain lower false positive rates in comparison with anomaly-based detection, however, they turn the detection scheme less flexible (Mudzingwa; Agrawal, 2012).

This work concerns the detection of specific attacks for SDWSN that have been barely defined, that could have multiple variations, and that have high impact on the network behavior. These characteristics match with the profile of attacks commonly detected through detection of anomalies on the network behavior. For this reason, next subsections examine techniques used to detect anomalies in SDN-based networks.

2.3.1.1 Anomaly detection in SDN

Anomaly-based detection methods for intrusion detection are grouped in five categories: soft computing, data mining, knowledge-based, statistical and machine learning. Soft computing methods aim to automatically classify new packets through problem-solving technologies, such as fuzzy logic. Knowledge-based methods depend on predefined rules to check the legitimacy of connection events

Machine Learning (ML) is a wide group of computational algorithms conceived to learn and emulate human intelligence (YAO; LIU, 2014). One application of ML algorithms is to learn patterns and classify them. This application has been used to implement anomaly-based intrusion detection systems. In SDN, the controller access to traffic information from all the network motivated the implementation of ML, since it simplifies data collection and the analysis of multiple features (XIE et al., 2019).

ML techniques are classified into four categories: supervised, unsupervised, semi-supervised, and reinforcement learning. Supervised algorithms learn from labeled data to generate a model to classify new inputs, while unsupervised algorithms aim to find structures or patterns in unlabeled data (SULTANA et al., 2019). Semi-supervised algorithms use labeled and unlabeled data and are useful in scenarios where to obtain labeled data is more expensive than unlabeled data. Reinforcement learning aims to take suitable actions to maximize long-term reward (XIE et al., 2019).

Supervised learning algorithms fit well within the intrusion detection problem since this is commonly addressed as a classification task. According to Xie et al. (2019), classification ML algorithms used in SDN for anomaly detection are: Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (K-NN), Neural Networks (NN) and Decision Tree (DT). Qazi et al. (2013) proposed Atlas, a DT-based classifier for mobile applications in SDN. Atlas used known mobile agents to train the algorithm to then be able to classify flows from guests devices. The authors reported an average classification accuracy of 94%. Silva et al. (2016) proposed ATLANTIC for anomaly traffic detection, classification and mitigation using SDN and SVM. ATLANTIC obtained an average classification accuracy of 88.7%. More ML proposals for security in SDN are analyzed in Chapter 3.

Statistical techniques apply a statistical inference test based on a model to determine if a new instance belongs to it. Techniques as change point detection (CPD), t-test, covariance and entropy-based methods have been used to detect DDoS attacks in legacy and SDN networks (XU; LIU, 2016; KALKAN; GUR; ALAGOZ, 2017; FOULADI; ERMIS; ANARIM, 2020). In this work, we focus on CPD since it is a relevant part of the security strategy proposed. Subsection 2.3.1.2 presents some examples of how CPD has been used to detect attacks in WSN and SDN.

2.3.1.2 Change point detection for security applications

CPD is the process of searching abrupt variations in a time-series or stochastic process properties, such as mean, variance, correlation, and spectral density. CPD methods are commonly divided in real-time detection and retrospective detection, also known as on-line and off-line methods, respectively. Real-time methods are suited for applications requiring immediate response, while retrospective methods target applications that can deal with longer reaction periods but require more accurate detection.

Pioneer works compare probability distributions of time series over different intervals (past and present) obtaining promising results (BASSEVILLE; NIKIFOROV et al., 1993). The main idea of these works was to set an alarm when a significant difference between two distributions is detected. Examples of methods using this strategy are: cumulative sum (CUSUM)(BASSEVILLE; NIKIFOROV et al., 1993), generalized likelihood-ratio

method (GUSTAFSSON, 1996), and change finder (TAKEUCHI; YAMANISHI, 2006).

There is another group of methods that measure the difference between subspaces to detect dissimilarity between past and present time series intervals. Two relevant approaches are *subspace identification* methods, which can be interpreted as a singular value decomposition of a weighted matrix (QIN, 2006), and *Krislov subspace learning* based algorithms, which are convenient when working with indefinite and asymmetrical matrices (SHITAO, 2018).

There are proposals in the literature that use CPD to detect anomalies in WSNs. Changlong Chen, Song and Hsieh (2010) proposed a detection mechanism of sinkhole attacks in WSN monitoring CPU usage in every node. Their main hypothesis is that CPU usage in malicious nodes is 80% higher than in benign node. Shakhov (2013) models WSN energy consumption when the network is under an energy exhaustion attack and proposes some methods to detect it. Part of these methods is to use change point analysis to detect an intrusion by monitoring the bit error probability. The main hypothesis is that bit error probability change when the network is under attack. Liu et al. (2011) introduces the self-diagnosis concept in WSN as a way to decentralized fault detection decisions, that are commonly taken in the sink, and encourage the sensor nodes to join this process. The authors used CPD algorithms to decide if a node should join the fault detection process.

There are also some proposals in the literature for wired SDN using CPD. Conti, Gangwal and Gaur (2017) proposed a distributed DoS (DDoS) detection attack and their main objective was to provide a unique solution to detect several fundamentally different DDoS attacks in SDN. The authors used CUSUM technique and incorporate an adaptive threshold to reduce false alarms. Hirayama, Kentaroh and Sasase (2015) and Mahrach et al. (2013) propose CPD mechanisms to detect flooding attacks. The first work used a detection scheme called *ChangeFinder* (TAKEUCHI; YAMANISHI, 2006) and the latter proposed a simple CUSUM algorithm with one threshold. Hirayama, Kentaroh and Sasase (2015) main objective was to detect the attack before the network congestion. To address this issue, they used multiple detection servers running the *ChangeFinder* mechanism to reduce false alarms caused by sudden increase of traceroute packets.

Finally, we explored the application of change point analysis to detect anomalies in SDWSNs. First, we applied Skaperas, Mamatas and Chorti (2018) algorithm to evaluate DDoS detection accuracy (SEGURA et al., 2020), obtaining similar results than ML approaches. Skaperas' algoritm has an offline phase to determine the best detection parameters and an online phase for a real-time CPD. Thus, we proposed a variation of Skaperas' algorithm (SEGURA; MARGI; CHORTI, 2020), eliminating the offline phase to reduce the memory and processing requirements without significant impact on the detection accuracy. This implementation is described in Section 5.2.

2.4 Summary

In this chapter, the basic concepts about SDN and LLNs are explained. The concept of SDWSN and a brief review of its evolution is presented, focused on IT-SDN, the framework used for our proposal implementation. Lastly, SDN-based networks security vulnerabilities are analyzed. This includes an explanation about how DoS attacks operates and how some anomaly detection techniques are used to detect them.

3 STATE OF THE ART

This chapter analyzes works that propose security solutions for resource constrained SDN-based networks. *The focus is on proposals targeting DoS and DDoS attacks detection and identification*; the analysis is based on detection and identification accuracy, type of DoS attacks detected, and the level of consideration of resource constraints. This last point concerns network structure, communication protocols and hardware specifications used or considered for experimentation.

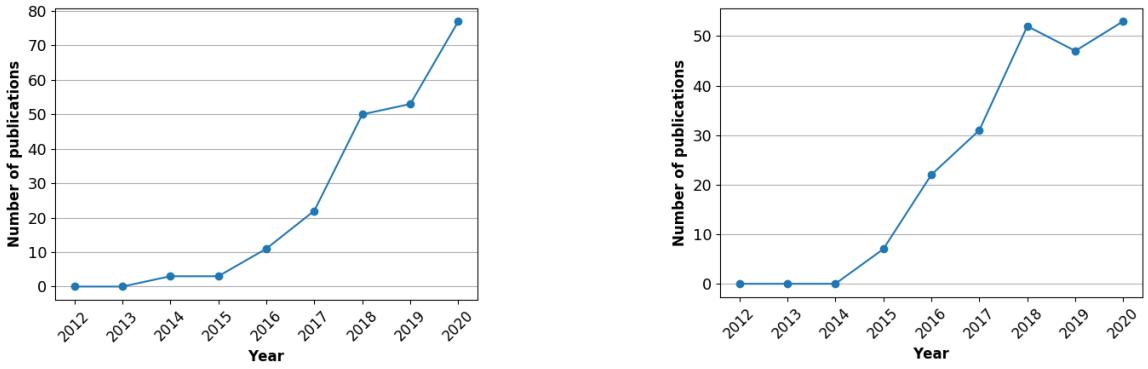
According to the information in the website dimmensions.ai[1], as shown in Figure 6, the number of publications about security in resource constrained SDN-based networks had a significant increase since 2016. This increase is related to the previous publication of groundbreaking works that showed that was possible to implement the SDN paradigm in resource-constrained networks.

First security proposals for SDN-based were totally dependent on the controller (SCOTT-HAYWARD; O'CALLAGHAN; SEZER, 2013). Eventually, some authors proposed to delegate some control back to network devices since the controller could become a bottleneck and is the most vulnerable element of the network (NAOUS et al., 2009; NAYAK et al., 2009). These authors stated that network devices as part of the attack detection aid to keep malicious traffic as much as possible in the data plane, reducing the chances to attain the controller. Also, that is an strategy used to address scalability and response issues during the attack (AHMAD et al., 2015). These insights were followed for SDN-based resource constrained networks as well, expanding the catalog of proposals. For this analysis, we split the literature in two groups: i) centralized approaches, i.e., works where the attack detection resides only in the control plane and ii) hybrid approaches, i.e., works where the network devices have an active role in detecting the attack.

3.1 Centralized approaches

Bagaa et al. (2020) propose a highly centralized OpenFlow-based security framework architecture for SD-IoT. Their objective was to provide protection at network and cloud levels combining SDN, NFV (Network Function Virtualization) and ML. However, in terms of DoS detection, this work follows previous proposals where a centralized entity collects information from the network devices to detect patterns or anomalies using ML. Their framework was deployed in a testbed with sensors distributed in three rooms, but there is no information about the number of devices and platforms models. Lastly, the results

[1] commercial platform for scientific research on publications https://app.dimensions.ai/discover/publication. Accessed: October 1st, 2021



(a) Keyword: SDWSN security

(b) Keyword: SDIoT security

Figure 6 – Number of publications about security in resource constrained SDN-based networks since 2012

Source: Author

show a detection accuracy exceeding 98%. A similar approach was proposed by Zarca et al. (2018).

Another highly centralized approach was proposed by Choi and Kwak (2016). Their security strategy is based on authentication and access control, using secure password authentication (SPA) and certification-based authentication, and traffic analysis to detect anomalies. However, Choi and Kwak (2016) were concerned about the processing overhead in the SDN controller caused by the traffic analysis, thus they proposed to include collector agents in the network. Despite this characteristic, the works is classified as a centralized approach since the controller is the responsible for the detection.

In terms of implementation and performance, we know the controller was implemented using the Floodlight Controller (MORALES; MURILLO; RUEDA, 2015) and the network environment using Mininet 2.0. Also, the authors described a simulation experiment used to analyze the framework performance, but they presented results only about resource usage, thus is not possible to evaluate the performance in terms of security. However, a weak point of this work is about the information provided of communication among the security agents. This is important to understand and evaluate resource usage.

Özçelik, Chalabianloo and Gür (2017) proposed to combine SDN and fog computing to bring the defense closer to the edge to avoid a disconnection between the IoT networks and the cloud in case of a DDoS attack. The detection is based on Threshold Random Walk with Credit-Based Rate Limiting and Rate Limiting. The first algorithm classifies the nodes based on the likelihood of a connection attempt being successful. This is based on the argument that a malicious node is continually trying to connect to random nodes to increase the infection. The second algorithm enqueue connections requests that are "not common" and processes them in order to wait a delay time between each processing.

When the number of connections requested in the queue surpass a threshold defined, the algorithm triggers an alarm. The malicious nodes are detected based on both algorithms and then the controller updates the routing rules to drop all the packets coming from these nodes. This paper shows interesting results about DDoS attacks detection and performance recovery but the cases of study do not represent the type of constrained network in our scope because of the following characteristics. The network is composed by eight IoT devices and 30 wired hosts, thus most of devices in the network have no energy constraints. IoT devices in the simulations are IEEE 802.11 compliant which means a significant difference in terms of bandwidth if compared to IEEE 802.15.4 compliant radios.

Bhunia and Gurusamy (2017) (*SoftThings*) and Ravi and Shalinie (2020) (*LEDEM*) proposed OpenFlow-based security architectures for IoT environments. Both are a two-tier organization with local multiple controllers and one universal or master controller. Local controllers monitor its corresponding network and report to the universal controller in case of detecting an attack. The universal controller can pass this information to other controllers and take actions in higher levels.

In Bhunia and Gurusamy (2017) proposal, the universal SDN controller orchestrates the whole network but the detection and mitigation are conducted by the local controllers. The anomaly detection is based on Support Vector Machine (SVM), a ML tool. Then, the local controller blocks suspicious flows and informs the actions taken to the Master controller. The performance evaluation showed detection rates between 90% and 98%, and recovery time between two and three seconds. On the other hand, this proposal was designed for home or small establishments networks, scenarios with low number of devices, without energy constraints and reachable in case of hardware problems of physical tampering. In terms of experimentation, the authors showed their proposal is able to detect and mitigate attacks from external and internal devices but considering scenarios with no more than five devices.

Ravi and Shalinie (2020) use a detection algorithm based on deep machine learning and it is executed in all the controllers. The authors state that their solution requires out-of-band communication to avoid saturation, which means it may not fit in all IoT environments. In terms of performance, the authors reported a detection accuracy of 96.28%. IDSIoT-SDL is a similar approach that uses deep learning (WANI; S; KHALIQ, 2021) as well. It is composed of an activity monitor, an activity analyzer and a classifier. Their results showed a detection rate exceeding 99% but tested on IEEE 802.11 networks.

3.2 Hybrid approaches

Wang et al. (2018) proposed ETMRM (Energy-efficient Trust Management and Routing Mechanism), an SDWSN trust management and routing mechanism to detect

selective forwarding and new-flow attacks. Trust, in this context, is a belief level that one node has on another. In this work, each node obtains the trust values of its neighbors based on their communication behavior. The controller uses local trust values to calculate a global trust value for each node. Nodes with a trust value under a threshold defined are classified as attackers. Then, the controller reconfigures the network to isolate these nodes. The mechanism was tested in simulations with 100 nodes, varying the number of attackers between 5 and 20 nodes. Their results showed a detection rate over 90% only for the case when five nodes are attackers, and this rate drops to values between 60% and 79% when 20 nodes are attackers. From this, we infer that their proposal could be not very effective on DDoS. Moreover, they considered CC2530 devices (IEEE 802.15.4 wireless MCU from Texas Instruments) for communication and energy consumption modeling. Thus we acknowledge this work as resource-constraints compatible.

Then, Bin-Yahya, Alhussein and Shen (2021) continues the proposal of Wang et al. (2018) but separating data and control traffic evaluation to increase detection sensitivity. Moreover, they include a hierarchical scheme to increase granularity and improve response time, which is supported by their results. Their performance analysis is based on simulations of a network with 40 nodes. In general, their results show their proposal improves ETMRM. Trust management approaches are effective over different attacks, such as black hole, selective forwarding and DoS. In turn, all the nodes remain more time active to monitor their neighbors' behavior. Additionally, for the proposal of Bin-Yahya, Alhussein and Shen (2021), a heterogeneous network is required to implement the clustering approach without reducing network lifetime.

Miranda et al. (2020) proposed a security framework for SDWSNs that combines intrusion prevention and anomaly detection divided in three layers: first, an IPS-based authentication process running in the data plane; second, an energy consumption prediction running in the sink; and third, a Support Vector Machine (SVM) algorithm running in the control plane that, using the information from the second layer, classifies nodes in trusted and malicious nodes. In this proposal, sensor nodes, sinks and the controller are an active part of the detection strategy, which alleviates the controller bottleneck. In turn, this overhead is passed to the data plane. In terms of performance, it was evaluated through simulations on a large network (i.e. 400 nodes), achieving a detection accuracy of 84.75%.

Yin, Zhang and Yang (2018) developed the OpenFlow-based framework SD-IoT, which includes a security system for DDoS attacks detection, based on the cosine similarity of packets received by switches. If the cosine similarity of the traffic received by a port exceeds a certain threshold, it means this port is being attacked. This work is not clear about how the system reacts when detecting the attack, but based on the results presented, it seems that the switch blocks the traffic in this port being attacked. This is another example about how some proposals choose to delegate some control back to network

devices. The authors evaluated their proposal through simulations using Mininet. The network size is not explicitly specified, but on the graphics and figures, it seems to be around 50 to 60 nodes.

Grigoryan et al. (2018) proposed the cooperation among SDN controllers to facilitate DoS attack mitigation. In the scenario of a DDoS attack targeting many devices of different SDN networks, the first controller detecting and confirming the attack shares the information with the controllers of other networks through a cloud. Using this information, the controllers can try to stop the attack by reconfiguring the network. Within the network, attacks are detected by the IoT devices with better capabilities. These devices should be registered to avoid malicious nodes to misinform the controller.

Grigoryan et al. (2018) evaluated their proposal using GENI testbed (GENI, 2018) and a real hardware environment. The experimental topology was composed by two controllers, three OpenFlow switches, two IoT devices and one attacker. Experiments were designed to test the delay during the whole detection and mitigation process sharing information between two SDN controllers. Results show that using this configuration one network is protected in around 550 ms and the second network is protected after 1 s the attack was triggered in average. Results show a faster attack detection and mitigation in comparison with Özçelik, Chalabianloo and Gür (2017) and Bhunia and Gurusamy (2017), but without testing scalability.

3.3 Analysis

From the works reviewed, we identified two approaches in the literature: centralized and hybrid. Centralized approaches obtained a high detection rate (i.e., over 95%) using ML techniques in most of cases. However, they impose high traffic overhead, requiring out-of-band communication in some cases (Ravi; Shalinie, 2020), and large datasets for training. Moreover, all of them are OpenFlow-based and assume a wired connection between the switches and the controller. These solutions do not apply to multi-hop/ad-hoc scenarios, such as WSN, where all nodes behave as switches, forwarding the packets to its destination. In such case, to apply such approach, one should monitor the traffic going through every node in the network.

In general, scalability was barely studied. Despite the experiments with 400 nodes in Miranda et al. (2020) with a detection rate lower than 90%, most works tested their proposals in networks with less than 100 nodes. Considering the centralized approaches characteristics in terms of communication resources and security requirements, what we expect from a scalability analysis is a decrease in the network performance. As shown by Alves et al. (2019), the delivery rate of data packets can be below 90% in SDWSN networks over 200 nodes. A security proposal that increases packets traffic could reduce even more this metric which turns important to analyze the viability of their implementation in these

sizes.

Hybrid approaches address high packet overhead issues but are limited by scarce processing, communication and energy resources in IoT and WSN devices. This restricts the complexity of security solutions that could run in these devices, understanding "complexity" as the tradeoff between resources usage and detection performance. In terms of results, Wang et al. (2018) and Miranda et al. (2020) did not obtain high detection performance. Bin-Yahya, Alhussein and Shen (2021) improved Wang et al. (2018) results but overloading the processing and communication resources usage in certain WSN nodes. In homogeneous networks, this could compromise network lifetime thus complexity is still a challenge (OLIVEIRA; MARGI, 2018).

Lastly, Table 2 summarizes the main characteristics of the works analyzed. We considered if they obtained high detection rate (represented as High DR), the ability to detect multiple attacks (represented as Multiple types), the ability to identify the type of attack (represented as Attack type ident.), if resources constraints were considered (represented as Resour. const.), the ability to identify the attacker or attackers (represented as Attacker ident.), if present scalability analysis (represented as Scal.) and if solve the complexity to run detection algorithms on resource-constrained devices (represented as complex.). From all the works, only Bin-Yahya, Alhussein and Shen (2021) fullfil the first five metrics. The hybrid approaches presented strategies to solve the control overhead traffic, instrumental to solve scalability, however, more experiments are required. Also, from all the hybrid approaches, only Bin-Yahya, Alhussein and Shen (2021) obtained high detection performance but at the cost of increasing communication resources usage and a hybrid infrastructure. For all this, we consider that scalability and complexity problems are still a challenge to solve. To cover these both problems, a hybrid approach that lays on the cooperation of all the elements in an SDN-based network is proposed. Also, our proposal is designed to fullfil the seven metrics presented in Table 2.

Table 2 – Related work

Reference	High DR	Multiple types	Attack type ident.	Resour. cons.	Attacker ident.	Scal.	Complex.
Centralized Approaches							
(CHOI; KWAK, 2016)	✓						
(Özçelik; Chalabianloo; Gür, 2017)					✓		
(Bhunia; Gurusamy, 2017)	✓						
(ZARCA et al., 2018)				✓			
(BAGAA et al., 2020)	✓	✓		✓			
(Ravi; Shalinie, 2020)	✓				✓		
(FARHIN et al., 2020)		✓					
(WANI; S; KHALIQ, 2021)	✓	✓	✓				
Hybrid Approaches							
(WANG et al., 2018)		✓	✓	✓	✓		
(BIN-YAHYA; ALHUSSEIN; SHEN, 2021)	✓	✓	✓	✓	✓		
(Miranda et al., 2020)		✓		✓		✓	
(GRIGORYAN et al., 2018)					✓		
(LI et al., 2019)	✓	✓					
(Yin; Zhang; Yang, 2018)				✓	✓		

Source:Author

4 PROBLEM FORMULATION

In this chapter, a security vulnerability analysis is presented. Then, the scalability and complexity issues in hybrid security proposals for SDN-based resource-constrained networks are described. The whole analysis is based on IT-SDN, an SDWSN framework previously described in Subsection 2.2.2. IT-SDN (ALVES et al., 2017) was developed by our research group and is completely open, free and available, which facilitates reproducibility of experiments and results. Moreover, IT-SDN performance was evaluated in networks up to 289 (Alves et al., 2019) nodes and tested in resource-constrained devices as TelosB and SensorTag, appropriate characteristics to evaluate the security proposal in this work.

4.1 Security Vulnerability analysis

In SDN, network devices are not able to make decisions about handling packets. When a network device has no rule for an incoming packet, it requests one to the controller. This simple behavior is the basis of new-flow attacks. The attackers flood the network with rule request packets to overload the control plane, and eventually, interrupt SDN controller communication. In the case of WSN, this attack is critical. Aggressive attackers can exhaust devices energy, leading to a permanent disconnection of the network.

Attackers are also able to reach the control plane through the network devices. Attackers can flood their neighbors with packets with unknown rules. In this case, the nodes flooding the network with rule requests packets are benign nodes. This attack can multiply the number of packets flooding the network by the number of nodes in the neighborhood of the attacker. This increases the radio and processing modules usage of the nodes around the attacker. Also, each packet with an unknown flow means a new rule, saturating routing tables after several requests. The saturation of routing tables can impact on the network performance since valid rules would be replaced or new valid rules will not have space, reducing delivery rate.

As explained in Subsection 2.2.2, a discovery protocol is executed to obtain topology information and share it with the controller. The controller uses this information to configure routing rules according to the policies programmed. The relevance of this information turns discovery protocol and neighbor information message exchange prone to attacks. Attackers in the route to reach the controller could identify these packets and disrupt their integrity to misinform the controller, for example, manipulating nodes addresses or routing metric values. Cryptography is a common proposal used to solve integrity issues but results show that its use in SDWSNs is still a challenge. Alves et al. (2018) proposed WS^3N, a Security Framework to handle node admission and symmetric key distribution in SDWSNs. However, their proposal has delivery rate issues even for

networks around 40 nodes, which are small for our approach. This occurs due to packet size increase, the time necessary to process messages, and the need to establish keys with a centralized entity. Toledo et al. (2020) followed a similar approach improving several aspects but increasing security level from 80 to 128 bits. According to their results, less than 30% of the nodes were able to complete the authenticated key agreement in networks of 36 nodes.

To understand the vulnerability of IT-SDN, several simulation experiments were executed setting up an SDWSN under three DoS attacks: false flow request (FFR), false data flow forwarding (FDFF) and false neighbor information (FNI). For the FFR and FDFF attack, a secure channel is assumed (authenticity of nodes and integrity of packets). On the other hand, the FNI attack targets networks without a secure channel. The implementation of the secure channel is not part of this work.

The experiments include grid topologies between 25 and 121 nodes, varying the number of attackers from one attacker to 10% of the total of nodes as attackers. The results were compared against the case where the network was not under attack but maintaining all the other configurations equal, i.e., the baseline case. Next, the description of each attack is presented.

1. The FFR attack targets the controller and its main goals are to increase the controller processing overhead and the packets traffic in order to increase the number of collisions. To attain these goals, the attacker sends multiple flow rule requests to the controller using different flow IDs. The controller processes the packets, calculates the rules, and sends the replies to the attacker. Figure 7 depicts the packets exchange during this attack.

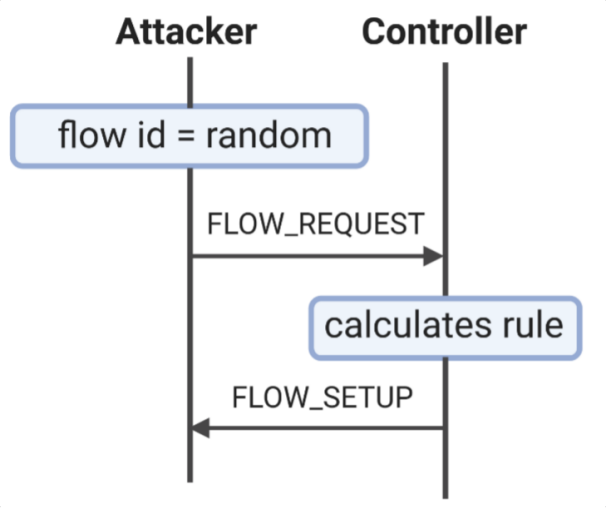


Figure 7 – False flow request message exchange for one iteration of the attack

Source: Author

2. A FDFF attack targets the controller via network devices. First, the attacker sends data packets with unknown flow IDs to its neighbors. The neighbors receive the packet and check the flow table to determine the action required, without success, thus ask a rule to the controller by sending a flow rule request packet. The controller receives this packet, calculates the rule and replies sending a flow setup packet. Figure 8 shows the packets exchange (one iteration) during this attack, which aims at increasing the network packet traffic and the controller and neighbors' processing overhead (SEGURA; MARGI; CHORTI, 2019).

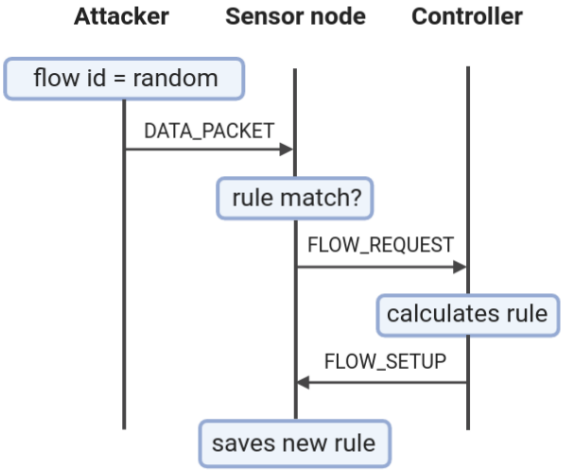


Figure 8 – FDFF attack, one iteration: the attackers send data packets to their neighbors using unknown IDs.

Source: Author

3. A FNI attack modifies the packets that contain neighbor information. The attackers do not intercept the neighbor information packets but modify the ones that use them to reach the controller. When receiving a neighbor information packet, the attacker modifies either the routing metric or the node identification number, then the packet continues its normal route to reach the controller. The packets exchange for this attack is depicted in Figure 9. This attack leads the controller to mistreat false information as true and to send erroneous routing rules to the nodes.

The FFR attackers were configured to send one false rule request every sixty seconds. Likewise, the FDFF attackers send one false data packet to each of their neighbors using this same frequency. This frequency could not be enough to disconnect the network rapidly, but can provide information about what metrics they affect still for weak attacks. The metrics analyzed are: data packets delivery rate, control packets delivery rate, control packets overhead, flow usage, control and data packets end-to-end delay and nodes energy consumption.

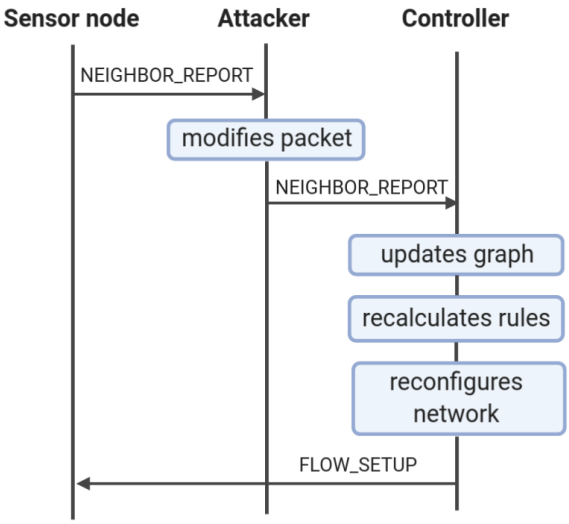


Figure 9 – FNI attack, one iteration: the sensor node sends a neighbor report to the controller and the attacker in the route (in the case there is one) modifies the neighborhood information before forwarding the packet to the controller.

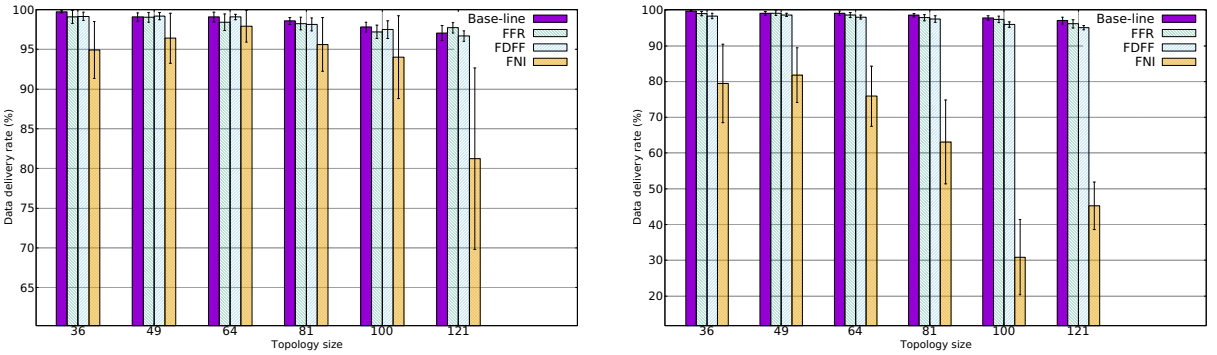
Source: Author

The delivery rate is the result of dividing the number of packets successfully received by the total number of packets sent. The end-to-end delay is the average time the packets spent to reach their destination. The control packets overhead is the total amount of control packets per minute. Finally, the energy consumption is the average energy consumption of all nodes during one hour of simulation, but excluding the sinks, the controller, and the attackers. We excluded the sinks and the controller because we assume those nodes have no energy restrictions and we excluded the attackers because we consider their energy consumption information could also be compromised.

4.1.1 Results analysis

The data packet delivery rate when there is only one attacker in the network is depicted in Figure 10a. The results show that in these conditions the FFR and FDFF attacks do not have a significant impact on the delivery rate, unlike the FNI attack does. In all the scenarios when the network is under a FNI attack, the average delivery rate is lower than the delivery rate in the baseline and the other attacks. Furthermore, we note a considerably higher variability (shown here as the standard deviation of the measurements) in the data packet delivery rates when compared to the baseline reference performance.

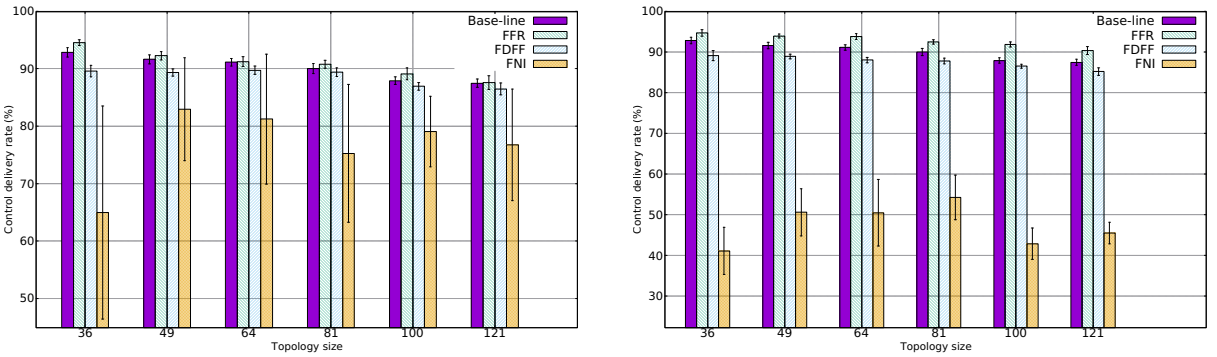
For the case when ten percent of nodes are attackers (Figure 10b), the results show that the networks under the FFR attack, and the networks from 36 to 81 nodes under the FDFF attack, maintain the baseline delivery rate. The networks with 100 and 121 nodes under the FDFF attack and all the network scenarios under the FNI attack show a decrease in the delivery rate with respect to the baseline results. In the case of the FDFF



(a) One attacker (b) 10% of nodes are attackers

Figure 10 – Comparison of the impact of each attack on the data packets delivery rate. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author



(a) One attacker (b) 10% of nodes are attackers

Figure 11 – Comparison of the impact of each attack on the control packets delivery rate. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author

attack, the delivery rate drop is less than 2%, while in the case of the FNI attack the drop is considerably higher, between 17% and 66%.

The delivery rate results for control packets are shown in Figures 11a and 11b for one attacker and for multiple attackers, respectively. Similar to the previous set of results for the data packet delivery rates, the FNI is the attack with the highest impact on the delivery rate. On the other hand, we observed two behaviors that were not present in the data packets results: (i) all the networks under the FFR attack with ten percent of nodes as attackers showed an improvement in the delivery rate for the control packets; and (ii) all the networks under the FDFF attack with multiple attackers showed a drop in their control packets delivery rate.

To understand both behaviors, an analysis of the total control packets overhead

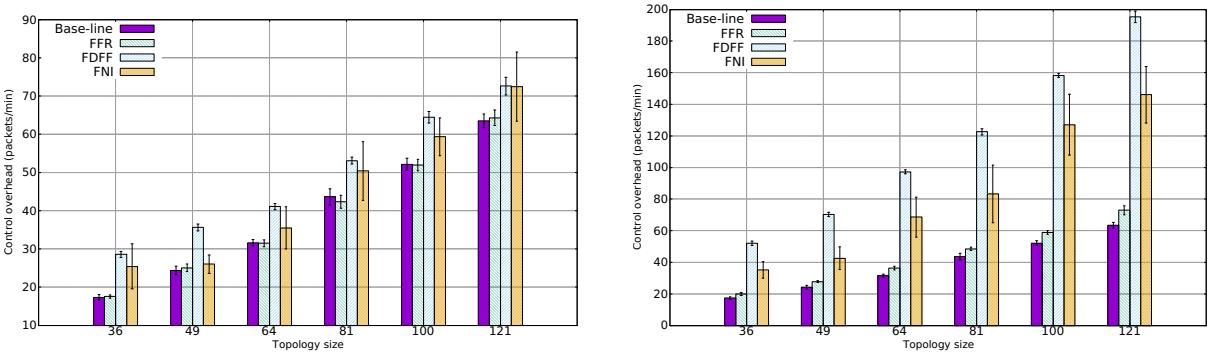
and the control packets traffic was conducted. The results in Figures 12a and 12b show that the networks under the FDFF attack have the largest control packets overhead for both single and multiple attackers. The FFR attack does not impact the control packets overhead when there is only one attacker, but induces an increase in this metric when there are multiple attackers. Then, a large control packet traffic due to the network configuration during the first three minutes was observed in Figure 13. Then, this number drops to zero. Subsequently, after the initial exchange of a large number of control packets the first three minutes of operation, in the baseline scenario the number of control packets remains very small, close to zero. On the other hand, in the FFR and in the FDFF attacks the control packets flow usage increases over 100 times once the attacks are launched.

Therefore, the control packets delivery rate obtained during the baseline scenario concerns mostly the first three minutes. On the other hand, when the attackers start to operate, there is constant but less dense control traffic than in the first three minutes. This is the main reason for the improvement observed in this metric. The situation is different for the FDFF attack mainly for two reasons: (i) the control packets overhead is larger than the control packets overhead in the FFR attack, and (ii) all the attacker's neighbors send a flow rule request at the same time, instead of only the single attacker sending one flow rule request per minute. Both situations increase the probabilities of collisions.

The control packets overhead results also provide information about the FNI attack. When there is only one attacker the average overhead in all the topologies sizes increases compared to the baseline results; interestingly, we also observed an increase in the standard deviation of this metric. In the case of multiple attackers, there is a high increase in the control packets overhead, with the corresponding gap – when compared to the baseline results – increasing with the number of nodes. For 36 nodes, the average control packets per minute increases by 102.78%, for 64 nodes it increases by 117.52%, and for 121 nodes it increases by 130.07%.

The data packets delay results depicted in Figure 14a and Figure 14b show a high dispersion for topologies from 64 to 121 nodes. Also, this dispersion is higher when there are multiple attackers in the network. Additionally, the mean values do not show any pattern along the different topologies sizes that could aid to determine the impact of each attack in the data packets delay. On the other hand, our baseline delay results coincide with the results obtained in (MARGI et al., 2018) and (ALVES; MARGI; KUIPERS, 2019) for topologies over 49 nodes. This indicates that it is common to have high dispersion in this metric even though the network is not under attack.

In the case of the control packets delay results shown in Figure 15a and Figure 15b, the networks with 36 and 49 nodes under the FDFF attack have the highest delay. This behavior is the same when there is only one attacker and when there are multiple attackers. For topologies over 49 nodes, the dispersion in the metrics increases and the difference



(a) One attacker

(b) 10% of nodes are attackers

Figure 12 – Comparison of the impact of each attack on the control packets overhead. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author

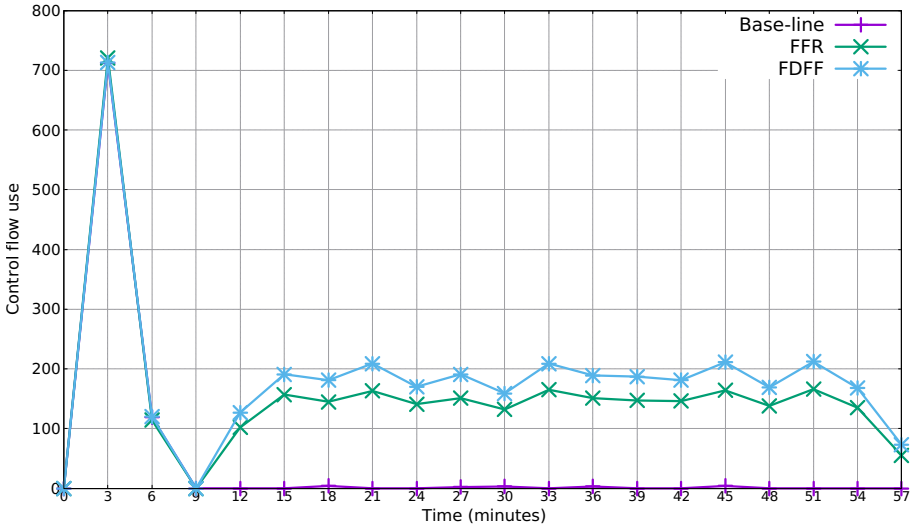


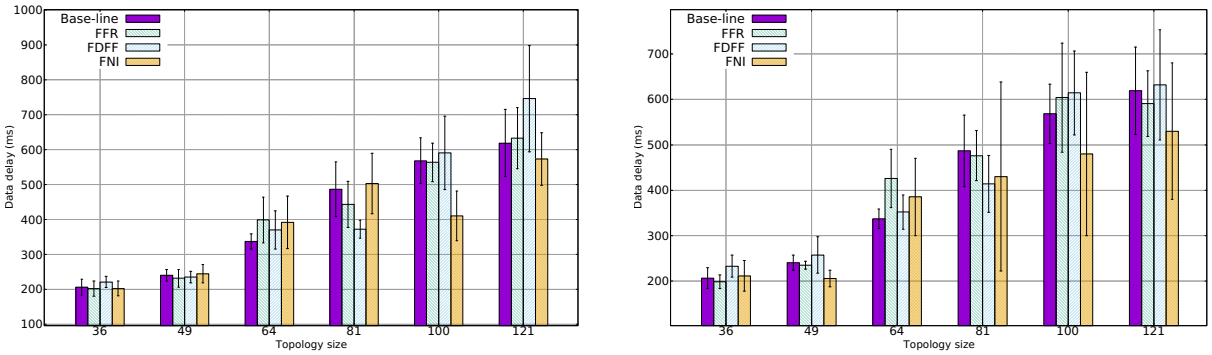
Figure 13 – Control packets flow usage

Source: Author

among the networks under attack delay results becomes less clear. For this reason, the dispersion in the delay metric render it difficult to determine the impact of each attack in the control packets delay when monitoring the average delay only.

Finally, the energy consumption results are shown in Figure 16a and Figure 16b for one attacker and multiple attackers, respectively. In both cases, the energy consumption remains unchanged, which means the attacks do not induce an energy consumption overhead for the network. On the other hand, all the nodes in the network were programmed to work without radio duty cycle, which means the radio module is turned on all the time. Since the radio module has the highest energy consumption in the node, the energy consumption overhead generated by the attacks becomes negligible.

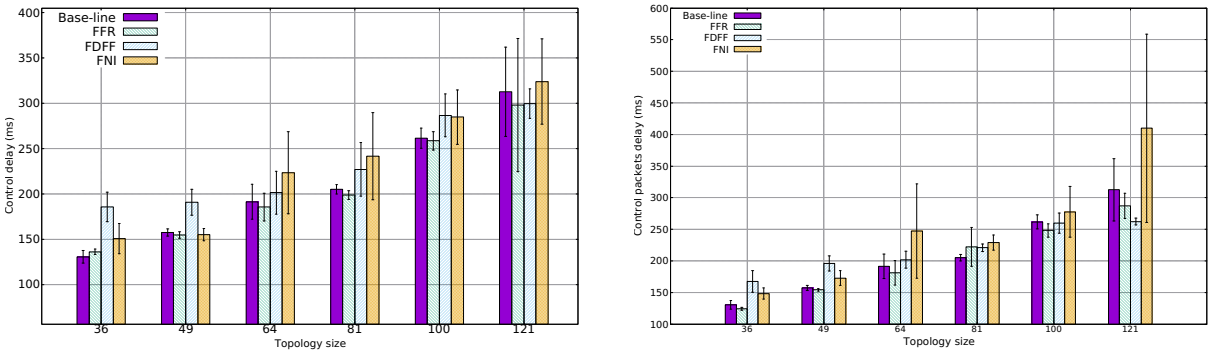
Summarizing, the FFR attack does not induce a significant change in the network



(a) One attacker (b) 10% of nodes are attackers

Figure 14 – Comparison of the impact of each attack on the average delay of data packets. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author



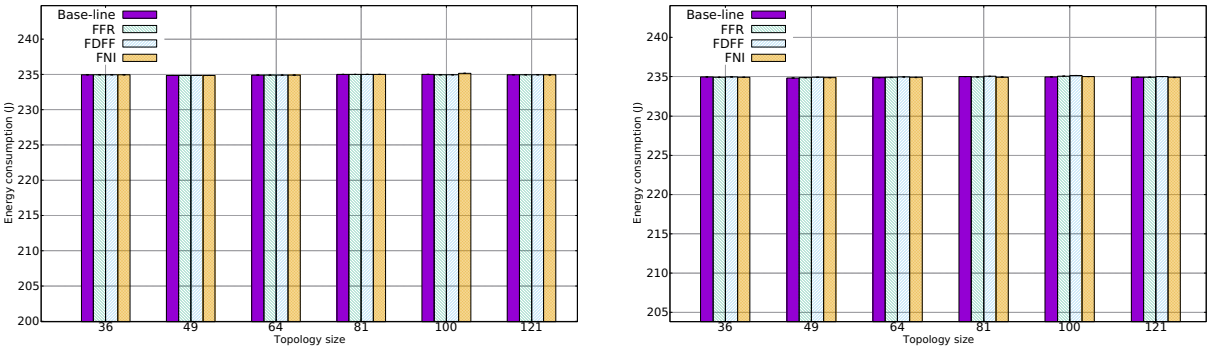
(a) One attacker (b) 10% of nodes are attackers

Figure 15 – Comparison of the impact of each attack on the average delay of control packets. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author

delivery rate, packets overhead and end-to-end delay. This means that those metrics are not proper indicators to detect when the network is under attack; at the same time, this attack is very mild and does not heavily impact the performance of the network. On the other hand, the control packets flow usage analysis offered useful information about the difference with the baseline scenario in terms of control packets traffic. Moreover, an increase in the average number of control packet delivery was identified for both single and multiple attackers, turning this metric a potential candidate for the identification of this type of attacks.

The scenarios under the FDFF attack indicated the highest control packets overhead and also a drop in the control packets delivery rate. Conversely, this attack did not modify the metrics related to the data plane. As a result, it is conceivable that a joint monitoring



(a) One attacker (b) 10% of nodes are attackers

Figure 16 – Comparison of the impact of each attack on the average energy consumption. (a) shows the results when there is only one attacker and (b) shows the results when 10% of the nodes are attackers.

Source: Author

of the control packet overhead and of the control packet delivery rate can offer the means to identify this type of attacks.

The FNI attack was the only one that affected both control and data packets metrics, both in terms of the average value of the metrics as well as of their dispersion. This attack reduced the control and data packets delivery rate and increased the control packets overhead. Also, the performance results show there is a significant difference when there is only one attacker and when there are multiple attackers.

Next, when the duty cycle is off, there are no significant modifications on the energy consumption. Lastly, the end-to-end delay metric used in this work did not give results that, at present, indicate the network is under attack. The main problem was the high dispersion in the results, which is consistent with previously published work.

4.2 Scalability and complexity

Such as discussed in Chapter 3, the state of the art is divided in two main approaches: the ones centralizing the detection, leveraged by the controller global view of the network; and hybrid approaches that delegate some control or intelligence back to network devices (SCOTT-HAYWARD; O'CALLAGHAN; SEZER, 2013)(AHMAD et al., 2015). Centralized approaches combine the global view of the network with powerful machine learning mechanisms to obtain accurate detectors, but at the cost of high packets traffic.

Experiments of network performance in SDN-based IoT networks (Alves et al., 2019) emulating TelosB motes, show that the delivery rate of data packets can drop below 90% in networks over 200 nodes. In these experiments all nodes were sending data packets to 1, 2, 3, or 4 sinks every 60 seconds, the neighbor reports sent to the controller were

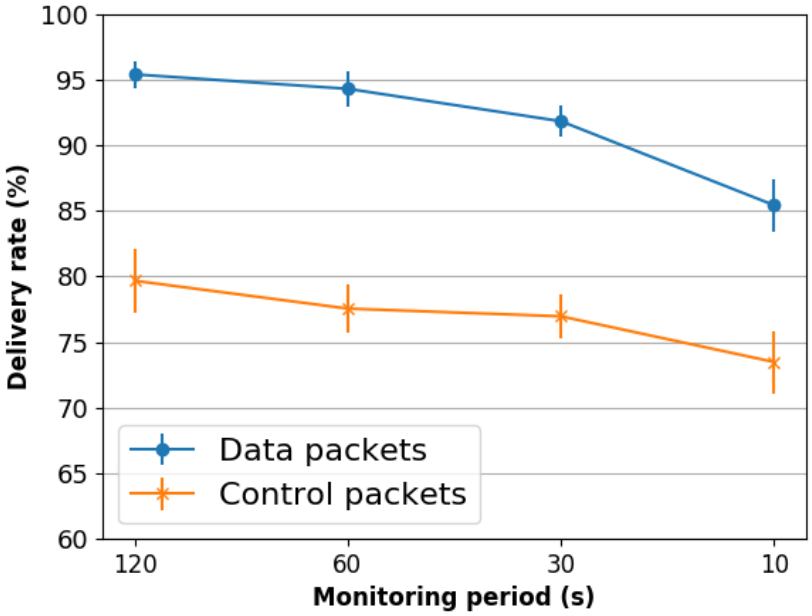


Figure 17 – Packets delivery rate increasing the monitoring period of the metrics used to detect attacks: (a) data packets and (b) control packets

Source: Author

limited to one every 60 seconds as well. Also, there was no security mechanism running in the network. Porting this scenario configuration to one with a completely centralized attack detection would be equivalent to every node reporting flow usage statistics to a controller or management server every 60 seconds. On one hand, aggressive attacks could tear down the network before two or three statistic updates reach the controller (SANGODOYIN et al., 2018). On the other hand, to increase the statistic update frequency n times, where $n \in \mathbb{R}^+$, is equivalent to multiplying the total packets traffic by $(1 + n/2)$. Increasing the frequency by two is equivalent to double the original packets traffic.

Figure 17 shows the average delivery rate for a 225-node topology, varying the period of time whereby every node reports metrics to a controller or management sink from 120 seconds to 10 seconds. These results show that, for an application where every node reports data every 60 seconds, reporting performance metrics every 10 seconds reduces data packets delivery rate in 10%. In the case of the control packets, the delivery rate did not reach an average value of 80%, even for the case reporting performance metrics every 120 seconds. Reducing the monitoring period to 10 seconds reduce control packets delivery rate in 6%.

Since the experiments were for multi-hop networks, the end-to-end delay metric considering all nodes has high dispersion. However, as shown in Figure 18, the average delay of data packets increase about 250% if changing the monitoring period from 120 seconds to 10 seconds. Furthermore, the results were not conclusive about the control

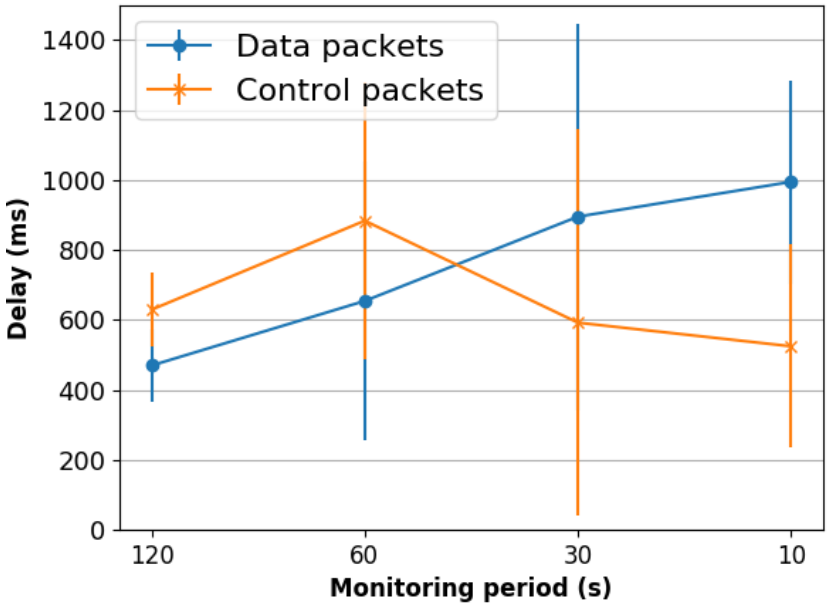


Figure 18 – Packets end-to-end delay increasing the monitoring periods of the metrics used to detect attacks: (a) data packets and (b) control packets

Source: Author

packets.

These results show that, even for applications with one sink, the performance metrics monitoring required in centralized approaches has a significant impact on the network performance. According to Alves et al. (2019), in scenarios with more than one sink node, these metrics can be reduced even more. The case with four sinks obtained 5% less data packets delivery rate than the case with only one sink. Therefore, in some cases with a certain quality of service requirements, centralized performance metrics monitoring is not the best option.

Hybrid approaches aids to keep malicious traffic as much as possible in the data plane, reducing the chances to attain the controller. Some hybrid approaches reduce packets traffic, however, their impact in terms of scalability has not been evaluated in the literature. Also, current hybrid proposals increase the nodes resources usage to the point that heterogeneous networks are required to implement the detection strategies. Thus, the complexity of the detection algorithms running in the network devices is a concern, not only processing requirement but memory space.

Proposals for SDN IoT networks, such as IT-SDN (Alves et al., 2019) or SDN-WISE (GALLUCCIO et al., 2015), require between 40 kB and 50 kB of flash memory to operate in standard conditions, this means, without considering the application layer. The TelosB mote is a WSN device with 48 KB of flash memory and 8 KB of RAM memory. Installing IT-SDN or SDN-WISE in these devices means taking most of its memory space, remaining

a few KBs for security applications. Some hybrid approaches based their attack detection on trust metrics. This reduces packets traffic but increases radio module usage, which is energy expensive. One strategy could be to detect the attacks based on their own behavior, which increases processing resources usage. One advantage is that processing is more energy efficient in contrast with communication energy consumption. Currently, there are devices with more memory resources, such as SensorTag and RE-Mote, however, targeting a solution that fits in TelosB motes turns it reachable to a wide option of devices.

Addressing the complexity problem is one step to address the scalability issue, since resource-constrained network devices would be able to run accurate anomalies detection algorithms on-site, alleviating the packets overhead required by centralized approaches. However, these individual nodes are not able to know if the anomaly is caused by an attack, or if it is caused by a benign behavior, such as network reconfiguration launched by the controller. In addition, if the anomaly is in fact caused by an attack, the SDN controller needs to be aware of this to take and/or execute configuration changes on the network devices to mitigate the attack. This means that the interaction among the different elements in the SDN architecture (Figure 2) is also vital to solve the security problem.

4.3 Summary

In this chapter, the security vulnerabilities of SDN-based resource-constrained networks and the scalability and complexity problem of current proposals were formulated. Three DoS attacks with different characteristics were implemented and their impact on the network performance was analyzed. The delivery rate and the control packets overhead were identified as the metrics where these attacks have more impact. Then the scalability of centralized security approaches was analyzed for networks with 225 nodes. These results showed how the monitoring frequency has significant impact on the network performance. Lastly, the complexity problem of current hybrid proposals was explained. Basically, they require resources that could limit their use in homogeneous networks. Thus, given the network security vulnerabilities, the impact of centralized intrusion detection in the network performance and the complexity problem of hybrid approaches, we posed the following question: “What are the benefits of solving the complexity problem in hybrid approaches to increase scalability of intrusion detection in SDN-based resource-constrained networks?”

5 COOPERATIVE INTRUSION DETECTION

This chapter is dedicated to explain the proposed method on intrusion detection proposal for software-defined resource-constrained networks. This proposal aims to address the scalability and complexity issues explained in Chapter 4 to detect DoS attacks.

The proposal is an intrusion detection strategy, inspired by multi-agent architectures (WOOLDRIDGE, 2009), where all the elements of the network cooperate to determine: i) when the network is under DoS attack, ii) the type of the attack and iii) to identify nodes launching the attack. The proposal considers four types of elements that participate in the intrusion detection: Security Manager, sensor devices, Performance Manager, SDN controller. The definition of these elements is based on the SDN architecture in the RFC 7426 (HALEPLIDIS et al., 2015).

The **Security Manager** is in charge of making the decisions in terms of security, and this means: i) determine when an anomaly, or anomalies, are caused by a DoS attack; ii) determine the type of the attack; and iii) identify the nodes launching the attack. The Security Manager uses all the information received from other elements to make decisions concerning the three points mentioned before.

The **sensor devices**, which compose the WSN itself, are constantly monitoring their behavior through metrics as: radio module usage, processing resources usage, and packets received and transmitted. Their objective is to detect anomalies on their behavior that could be related to a DoS attack. When a sensor device detect an anomaly, it raises an alarm and informs the Security Manager about it. Additionally, if possible, the sensor devices send information about nodes with suspicious behavior.

The **SDN Controller** is where the control decisions are taken. For this reason, this agent has topology information and control traffic information that is important for their own decisions and instrumental to detect DDoS attacks. The SDN Controller agent provides network graph information to the Security Manager and reconfigures the network according to Security Manager instructions.

The **Performance Manager** collects data from the sensor devices and the SDN Controller, calculates metrics to monitor the network performance, and provides these metrics values to the Security Manager, when required. The Security Manager analyzes these metrics to detect anomalies on the network performance.

Next, in Section 5.1 the role of the elements and their interactions before the attack, during the attack detection and attackers identification is explained.

5.1 Interaction among agents

As explained before, the Security Manager receives information from other agents to take security-wise decisions. In the following, we explain the interactions among the agents necessaries to detect DoS attacks. For this, we use three possible cases: i) no anomaly detected yet, ii) an anomaly is detected by a Sensor device, and ii) an anomaly is detected by the Security Manager agent. The interactions among the agents are summarized in Figure 19.

5.1.1 Normal operation

- Sensor device – Performance Manager: The sensor devices send performance metrics to the Performance Manager. The Sensor devices can be programmed to send the metrics periodically or when the Performance Manager requests it.
- Security Manager – Performance Manager: The Performance Manager sends performance metrics to the Security Manager. This interaction can be periodic or by request of the Security Manager.
- Performance Manager – SDN Controller agent: The Performance manager requests performance metrics to the SDN Controller.

5.1.2 Anomaly detected by a Sensor device

- Sensor device – Security Manager: The sensor agent informs to the Security Manager about the anomaly detected. If the Security Manager determines that the anomaly was not an attack, it informs to the Sensor devices to continue operating normally.
- Security Manager – Performance Manager: The Security manager asks to the Performance Manager about recent configurations executed on the node sending the alarm or in its neighborhood.
- Security Manager – SDN Controller: The Security Manager asks to the SDN Controller about recent configurations executed on the node sending the alarm or in its neighborhood.

5.1.3 Anomaly detected by the Security Manager agent

- Security Manager – Performance Manager: The Security manager asks to the Performance Manager about recent configurations executed on the network.
- Security Manager – SDN Controller: The Security Manager asks to the SDN Controller about recent configurations executed on the network.

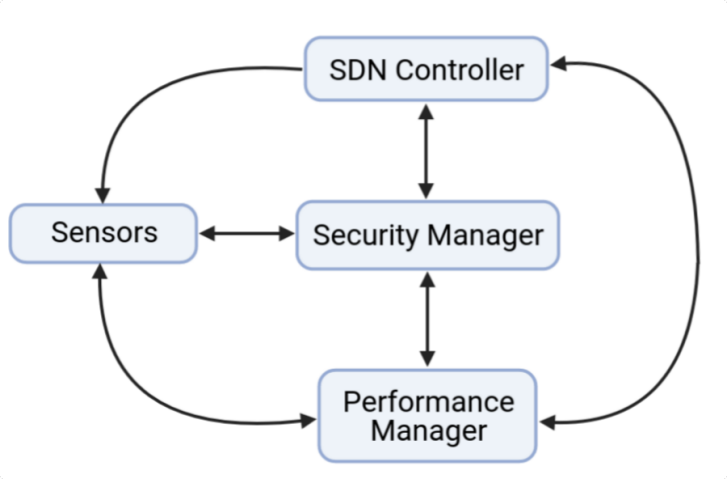


Figure 19 – Interaction among agents

Source: Author

- SDN Controller – Sensor devices: If the anomaly was classified as an attack, the SDN Controller reconfigures the Sensor devices to mitigate the attack.

5.2 Anomalies detection

The Security Manager takes all the security decisions. These decisions include determining when the network is under attack based on the anomalies detected and the information received from the other agents. The agents in charge of detecting anomalies are the sensor devices and the Security Manager. For this, they execute a change point analysis algorithm for real-time detection on time-series constructed from metrics as delivery rate and control packets overhead. The results in Chapter 4 showed DoS attacks have a significant impact on these metrics.

Skaperas, Mamatas and Chorti (2019) propose a real-time, autonomous and low complexity (i.e. $O(N \log N)$, where N is the length of the monitoring window) change point detection algorithm with a success rate exceeding 94%. Their proposal is composed of an off-line phase, an on-line phase and a trend indicator. The off-line phase is a training period employed to configure the on-line phase and the trend indicator determines the direction of the change. The low-complexity, success rate and access to the whole implementation were the main reasons to choose Skaperas, Mamatas and Chorti (2019) as the base of our anomaly detection algorithm. However, we decided to implement only the on-line phase to be able to fit it in resource-constrained nodes. The off-line phase was changed for a manual configuration based on a security policy: prioritize accuracy or detection speed. The trend indicator was not part of the current implementation as well.

5.2.1 On-line change point detection algorithm

Generally, CP problems are formulated as hypothesis tests. The null hypothesis represents the structural stability of the process, while the alternative hypothesis indicates one or multiple CPs and is used to detect an anomaly. The test statistics may be viewed as two-sample tests adjusted for the unknown break location, thus leading to max-type procedures. Often asymptotic relationships are derived to obtain critical values for the tests. After the null hypothesis is rejected, the location(s) of the break(s) need(s) to be estimated (AUE; HORVATH, 2013).

To outline the on-line CP algorithm, let $\{X_n : n \in \mathbb{N}\}$ be the time series of the metric monitored. Using Wold's theorem we can assume that, for $X_1, ..., X_N$, each sample is expressed as $X_n = \mu_n + Y_n$, where $\{\mu_n, n \in \mathbb{N}\}$ is the mean of the time series and $\{Y_n : n \in \mathbb{N}\}$ is a random zero mean term, so that we can rewrite X_n as:

$$X_n = \begin{cases} \mu + Y_n, & n = 1, \ldots, m + k^* - 1 \\ \mu + Y_n + I, & n = m + k^*, \ldots \end{cases} \tag{5.1}$$

where $k^* \in \mathbb{N}^*$ represents the unknown time of change and μ, $I \in \mathbb{R}^r$ represent the mean parameters before and after k^*, respectively. We here assume a period of no change in the mean of at least m samples, i.e., during the first m samples of our observation there is no change so that $\mu_1 = \ldots = \mu_m$.

During this period, our detector "learns" in real-time the statistics of the observed time series and the mean value in particular. The statistical hypothesis test is articulated as,

$$\begin{aligned} H_0 &: I = 0 \\ H_1 &: I \neq 0. \end{aligned} \tag{5.2}$$

The on-line sequential analysis belongs to the category of stopping time stochastic processes. In general, a chosen on-line test statistic $TS_{on}(m, l)$ and a given threshold $F(m, l)$ define the stopping time $\tau(m)$:

$$\tau(m) = \begin{cases} \min\{l \in \mathbb{N} : TS_{on}(m, l) \geqslant F(m, l)\}, \\ \infty, \text{ if } TS_{on}(m, l) < F(m, l) \ \forall l \in \mathbb{N}, \end{cases} \tag{5.3}$$

implying that $TS_{on}(m, l)$ is calculated on-line for every l in the monitoring period. The procedure stops if the test statistic exceeds the value of the threshold function $F(m, l)$. As soon as this happens, the null hypothesis is rejected and a CP is detected. $F(m, l)$ is defined as,

$$F(m, l) = cv_{on,\alpha} g(m, l), \tag{5.4}$$

where: (i) $cv_{on,a}$ is the critical value determined from the asymptotic behavior of the stopping time procedure under H_0 by letting $m \to \infty$, (ii) and $g(m, l)$ is the weight function defined as:

$$g(m,l) = \sqrt{m}\left(1+\frac{l}{m}\right)\left(\frac{l}{l+m}\right)^{\gamma} \tag{5.5}$$

where the sensitivity parameter $\gamma \in [0, 1/2)$.

The on-line algorithm uses the standard CUSUM detector (FREMDT, 2014), with test statistic denoted by TS_{on}^{ct}. Its corresponding critical value is denoted by $cv_{on,\alpha}^{ct}$ and the stopping rule by $\tau_{ct}(m)$. The sequential CUSUM detector is denoted by $E(m,l)$,

$$E(m,l) = \left(\overline{X}_{m+1,m+l} - \overline{X}_{1,m}\right). \tag{5.6}$$

The standard CUSUM test statistic is expressed as:

$$TS_{on}^{ct}(m,l) = l\widehat{\Omega}_m^{-\frac{1}{2}} E(m,l), \tag{5.7}$$

where $\widehat{\Omega}_m$ is the estimated long-run covariance, which captures the dependence between observations. Then, the stopping rule $\tau_{ct}(m)$, is defined as:

$$\tau_{ct}(m) = \min\{l \in \mathbb{N} : \|TS_{on}^{ct}(m,l)\|_1 \geq cv_{on,\alpha}^{ct} g(m,l)\}, \tag{5.8}$$

where the ℓ_1 norm is involved to modify TS_{on}^{ct} so that it can be compared to a one-dimensional threshold function. The critical value, $cv_{on,\alpha}^{ct}$, is derived from the asymptotic behavior of the stopping rule under H_0:

$$\lim_{m\to\infty} Pr\{\tau(m) < \infty\} \tag{5.9}$$

$$= \lim_{m\to\infty} Pr\left\{\sup_{1\leqslant l\leqslant\infty} \frac{\|TS_{on}^{ct}(m,l)\|_1}{g(m,l)} > cv_{on,\alpha}^{ct}\right\}$$

$$= Pr\left\{\sup_{t\in[0,1]} \frac{\|W(t)\|_1}{t^{\gamma}} > cv_{on,\alpha}^{ct}\right\} = \alpha \tag{5.10}$$

where $W(t)$ denotes the Brownian motion with mean 0 and variance t. The on-line critical values can be computed using Monte Carlo simulations, considering that,

$$cv_{on,\alpha}^{ct} = \sup_{t\in[0,1]} \frac{W(t)}{t^{\gamma}}. \tag{5.11}$$

Lastly, the estimated on-line CP, $\hat{k}_{on}^{*}$, is derived directly from the value of the stopping time $\tau(m)$, as,

$$\hat{k}_{on}^{*} = m + \{\tau(m) | \tau(m) < \infty\}. \tag{5.12}$$

In summary, the change point detection algorithm has 3 main steps. The first step is an observation period of the metric being monitored. From this observation, the algorithm "learns" in real-time the statistics of the time series and the mean value in particular.

The on-line change point detection starts when the observation period ends. During the on-line detection, the change on the time series for every new sample is evaluated. If after k samples no change point is detected, the k samples are aggregated to the time series observed in the first step and the new statistic characteristics are calculated. Then, a new on-line detection period starts. However, if a change point is detected, the algorithm stops.

5.3 Anomalies classification

In this section, the classification process of anomalies to determine when the network is under attack is described. The anomalies are classified in two groups: simple anomalies and complex anomalies. However, in this work we include the evaluation only for the simple anomalies. The evaluation of the classifier for complex anomalies is not part of the objectives of this work.

5.3.1 Simple anomalies classification

Simple anomalies include the cases of networks in steady-state without large control and management configurations apart from the initial one. We suppose a fixed number of nodes during the whole operation, which means, no nodes join or leave the network.

Under these conditions, the network performance should remain stable thus the Security Manager declares the network under attack when:

- an anomaly on a centralized metric is detected, or
- an attacker is identified.

The Security Manager was designed to be able to run multiple detectors in parallel. In this manner, we expect to increase the detection probability, but also, to relate the metric where the anomaly is detected and the type of the attack. Based on the results about the impact of DoS attacks in SDWSN presented in Chapter 4, we observed two important relations: control packets overhead is the first metric impacted by FDFF attacks (new-flow type) and the data packets delivery rate is the first metric impacted by the FNI attack (routing information integrity type). From these results, the next hypothesis was posed: *if the CP is first detected in the mean value of the control packets overhead, the attack is classified as FDFF; conversely, if the CP is detected first in the mean value of the data packet delivery rate, the attack is classified as FNI.*

The Security manager identifies the attackers based on the alarms received from sensor nodes. The first hypothesis is that nodes around the attacker are the first detecting an anomaly in their behavior. Thus, when the Security Manager receives an alarm, all its neighbors become suspects. While more nodes report alarms, the list of suspects increases but also the probability of one suspect being reported multiple times. Based on this, the

next identification criteria was established: *a suspect is classified as an attacker when all its neighbors have reported an anomaly detected.*

The FDFF attack is characterized for inducing the attackers' neighbors to send flow rule requests to the controller by creating data packets with unknown flows identifiers. Based on this, all nodes are able to determine a suspect when detecting an anomaly by checking the source of the last data packets with unknown flow received. Then, a suspect is identified as attacker when the Security Manager has received x reports from different nodes, where $x \in [1, neighbors]$. Low values of x prioritize detection speed while high values prioritize accuracy. The implementation of both identifiers is explained in Section 5.4.

5.3.2 Complex anomalies classification

This scenario includes the networks with constant global reconfigurations in both the control and management plane caused by: a change in the routing policies, a change in the management policies and nodes joining and leaving the network.

In these cases, a more complex verification is required. When multiple nodes join or leave the network, the graph and the routing metrics suffer modifications. This process involves the nodes joining the network, the nodes in their neighborhood, and the controller. Also, it indirectly involves all the nodes in the routes to reach the controller or other flow destinations. The new nodes and their neighbors will update their neighbor tables and send a report to the controller. Then, the controller calculates the new rules and configures the network. This increases the packets traffic and hardware usage metrics in all the nodes involved, and also increases the network control overhead metric.

To reduce the false positives in situations like these, the Security Manager asks the controller about recent configurations on the nodes that reported an anomaly, understanding "recent" as a parameter to investigate. If the controller confirms a recent configuration on these nodes, the next step is to ask if these configurations include flow rule requests for unknown flows, unknown for even the SDN Controller. If this in fact happened, the Security Manager classifies the anomaly as an attack. Conversely, if there are no unknown flows detected by the SDN Controller, the Security Manager considers these anomalies as normal, but remains aware of a centralized anomaly detection that could indicate a significant impact on the network, for example, a significant drop on the data packets delivery rate. If the SDN Controller denies recent configuration on these nodes, the Security Manager asks to the Performance Manager about recent configuration on these nodes. If the Performance Manager confirms the recent configuration, the Security Manager discard the attack. On the other hand, if the Performance Manager denies the recent configuration as well, the anomalies are classified as an attack, understanding that these anomalies are caused by unknown entities. The algorithm to classify the anomalies

detected on the Sensor devices is summarized in Figure 20.

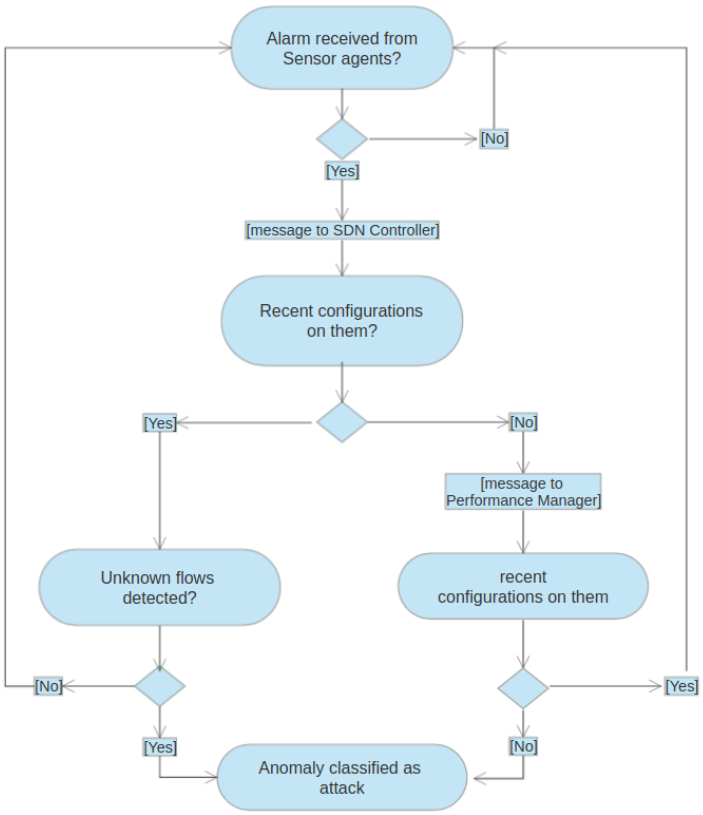


Figure 20 – Anomalies classification: Algorithm running on the Security Manager when receiving one or more anomaly alerts from the Sensor devices

Source: Author

5.4 Implementation

The proposed method was implemented using IT-SDN, an open SDWSN framework. Using IT-SDN southbound protocol we were able to communicate the WSN with the SDN controller and the Performance Manager. The SDN Controller and the Performance Manager run in a desktop computer and a WSN node works as gateway for both via serial connection. At the initial bootstrap, the controller configures every node in the network to reach it. Then, the Performance Manager informs to the controller that it is a destination for management packets using a flow id register packet. Now, the SDN Controller is able to configure routes to reach the Performance Manager.

Sensor nodes are pre-programmed to send performance metrics to the Performance Manager periodically. The period is a parameter that depends on the sampling frequency we want for the centralized monitoring in the Security Manager. Also, this is an important parameter since this period should change according to the topology size and performance metrics. However, for our case this is hardcoded based on previous tests to determine the right value. For example, in Chapter 4 it was shown that using a periodic period of 60

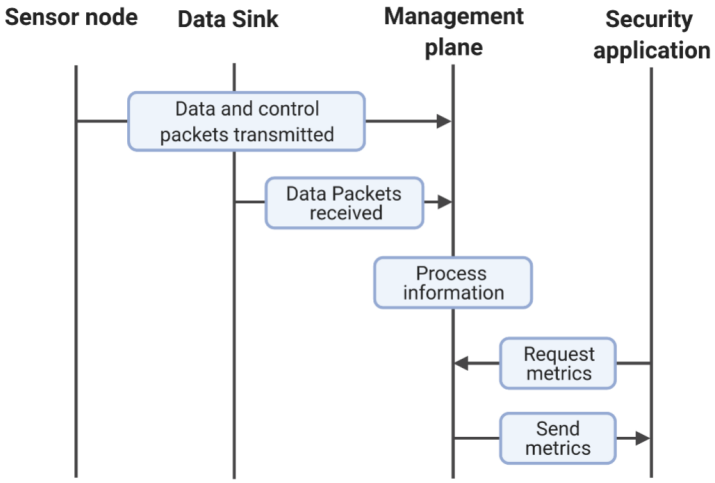


Figure 21 – Centralized monitoring messages exchange for time series constructions

Source: Author

seconds in a 225-node topology the impact on the data packets delivery rate was minimum. Then, the Security Manager is able to request performance metrics to the Performance Manager. The complete centralized monitoring messages exchange is depicted in Figure 21. However, it is worth mentioning that IT-SDN does not have a northbound protocol implementation, thus the Performance Manager stores the metrics information and the Security Manager accesses it.

Sensor nodes are also pre-programmed to monitor its own behavior, for example, radio module usage, processing overhead in miliseconds or number of packets transmitted. The algorithm running in every node is the same running in the controller but limited in terms of the number of metrics that can be analyzed at the same time. For the case of the TelosB mote, which is the platform we used, the limit was one metric using a training of 200 samples and a monitoring window of 50 samples, for samples of 16 bits. When a sensor node detects an anomaly, it sends a message to the Security Manager. A packet was included in the Southbound protocol, exclusively for this purpose. In this packet the sensor can include important information, such as the metric on which the anomaly was detected and a suspect address, among others. This packet is forwarded to the SDN controller, which in turn sends this information to the Security application. Then, the Security Manager can request topology information to the SDN Controller. For our case, we implemented two functions: one to ask the number of neighbors of one node and other to ask the addresses of its neighbors. The messages exchange is depicted in Figure 22.

Then, two algorithms were implemented to identify the attackers. The first algorithm (Algorithm 1) is based on the hypothesis that nodes around the attacker would be the first detecting anomalies on their behaviors. When the Security Manager receives an alarm, it asks the addresses of the suspect's neighbors to the controller. All the nodes in the neighborhood are now suspects. The Security Manager creates a counter for each

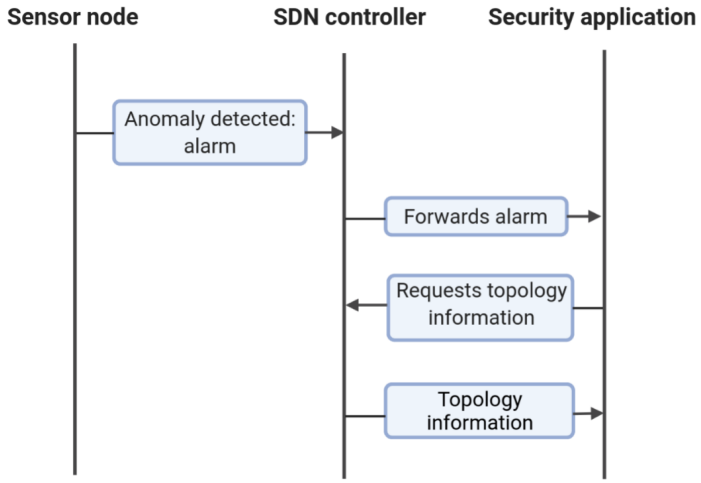


Figure 22 – Distributed anomaly detection messages exchange

Source Author

Algorithm 1 FDFF attackers identification

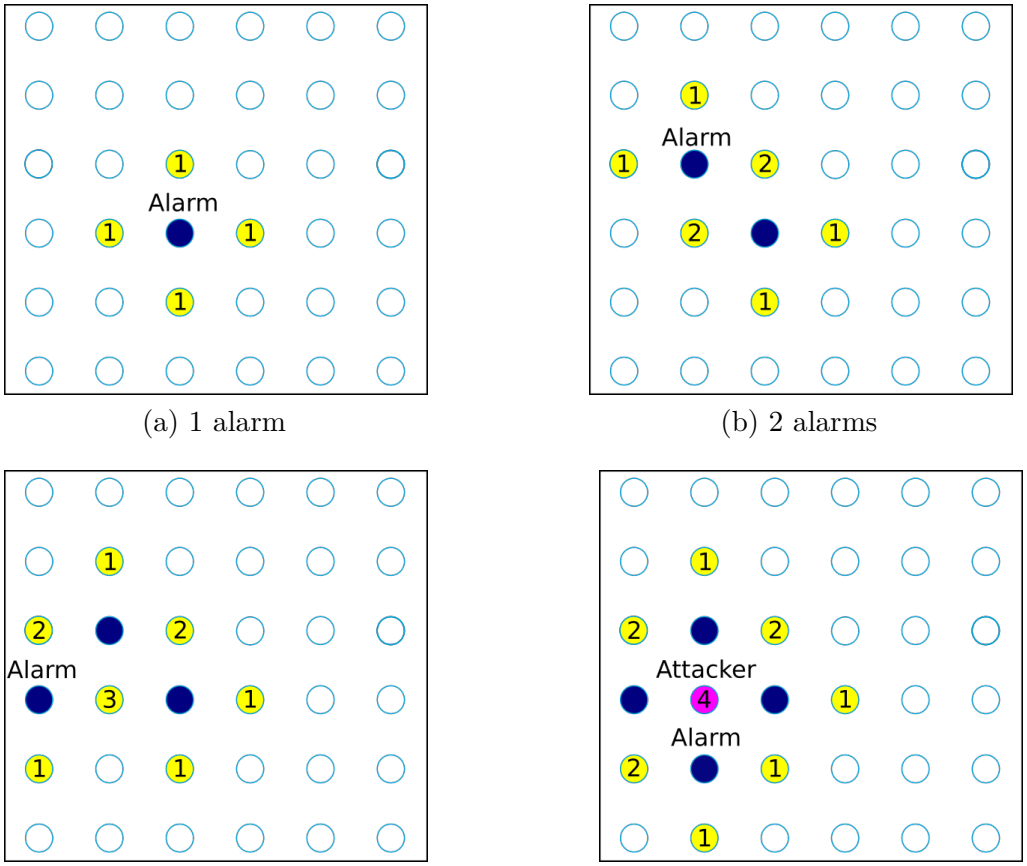
alarm_packet{source_node}: alarm received from "source node"
suspects_addr: addresses of the source node neighbors
counter(n): counter to register the times the node "n" has been reported as suspect
neighbors: number of nodes in the neighborhood of the suspect

---

Security Manager:
Wait alarm_packet{source_node}
suspects_addr = ask_controller_neigh_addr(source_node)
for n in suspects_addr **do** counter(n)++ neighbors = ask_controller_neigh_total(n)
    if counter(n) == neighbors **then**
        s = attacker
    end if
end for

suspect to register the times it is in the neighborhood of a node reporting an alarm. When the counter and the number of suspect's neighbors are equal, the suspect is classified as attacker since this means that all the nodes around the attacker detected an anomaly. The detection process is summarized in Figure 23.

Then, we proposed another identification algorithm but specific for the FDFF attack. In this case, a register on the sensor nodes stores the addresses of the nodes sending data packets with unknown flow IDs. The size of the register will be defined according to the detection speed results. When the node detects an anomaly, it checks the addresses in the register and selects as suspect the node with the higher frequency, i.e. the mode. The size of the register should be defined according to the remaining memory space and the detection speed of the detector. Since the detection speed changes according to the attack characteristics and the algorithm configuration, this parameter would be defined when having first results about distributed detection. Also, this register should be refreshed



(a) 1 alarm

(b) 2 alarms

(c) 3 alarms

(d) 4 alarms

Figure 23 – Attacker identification using Algorithm 1. In blue, the node reporting the alarm and in yellow the nodes that become suspects. Inside the suspects in the number of neighbors that have reported an alarm

Source: Author

periodically to avoid declare as suspect benign nodes.

The Security Manager receives the alarm and the suspect's address. The Security manager checks if this address has been reported before and if not, it creates a counter for future reports. Then, the Security Manager asks the number of neighbors of the suspect to the controller and compares this value with the suspect's counter. If the counter and the number of neighbors are equal, the Security Manager classifies the suspect as an attacker. However, since in this case the sensor node is already detecting a suspect, the attack declaration and attacker identification could be done before all the neighbors report the alarm. Assuming this, we could reduce the detection speed but also increase the false positive rate. Algorithm 2 presents the attacker identification algorithm.

These algorithms performance relies on the hypothesis that the attacker will target all its neighbors. In the case the attacker targets only a a section of its neighborhood, algorithms 1 and 2 will not identify the attacker. However, there is a list of suspects that can provide a clue to the Security Manager about the nodes launching the attack when

Algorithm 2 FDFF attackers identification 2

alarm: signal when the sensor node detects an anomaly
new_packet_flow: flow id of a new packet received
suspect_register: register to store addresses of the suspects
suspect: address of the suspect
suspect_counter: stores the times a suspect has been reported
alarm_packet: packet sent by the sensor nodes to inform of an anomaly
neighbors: number of nodes in the neighborhood of the suspect

---

In sensor node:
while !alarm **do**
  if new_packet_flow == unknown flow **then**
    include source in suspect_register
  end if
end while
suspect = mode(suspect_register)
send alarm to Security Manager
Security Manager:
Wait alarm_packet
if new suspect in alarm **then**
  create suspect_counter
else
  suspect_counter++
end if
neighbors = ask_to_controller(suspect)
if suspect_counter == neighbors **then**
  suspect is attacker
end if

detecting an anomaly using one centralized detector. This strategy will be evaluated in future works.

5.5 Summary

This chapter describes our proposed method for intrusion detection based on the cooperation of all the elements of the network and inspired on the multi-agents paradigm. First, four agent types and their interactions were defined. Next, the on-line anomaly detection algorithm proposal was explained. Then, the anomalies classification criteria was described. For this, we defined two scenarios: simple anomalies and complex anomalies. Lastly, the implementation of the whole proposal was explained.

6 PERFORMANCE EVALUATION

This chapter contains the performance evaluation results of the cooperative intrusion detection proposal. Section 6.1 describes the experiments and methods while Section 6.2 presents and discusses the results. Lastly, Section 6.3 summarizes the insights obtained from these experiments.

6.1 Experiments design

From the analysis of the literature (Chapter 3), we noticed that: i) scalability in intrusion detection proposals for resource-constrained SDN-based networks has not been addressed and ii) hybrid approaches, which address high packets overhead, are limited by lack of resources in network devices.

The main objective of this work is to design and evaluate a hybrid intrusion detection for resource-constrained SDN-based networks that address distributed intrusion detection limitations and increase scalability, according to the results presented in Chapter 4. Three groups of experiments were designed to evaluate our proposal according to the objectives mentioned before.

In the first group, the centralized DDoS attack detection performance of the Security Manager was evaluated. The second group was designed to test the distributed anomaly detection, running the change point detection algorithm on every node. The third group comprises the experiments to evaluate the performance of the whole proposal. The experiments are mainly simulations of WSNs using Cooja simulator (Osterlind et al., 2006), which allowed us to emulate real platforms.

For all the groups, fully-bidirectional grid topologies emulating Tmote sky (i.e. an equivalent of the widely known TelosB mote (CROSSBOW, 2004)) were used. This platform has an MSP430 microprocessor, 48 kB of flash memory, 10 kB of RAM and a CC2420 transceiver. For the first and second groups, the detection performance was evaluated on 36-node and 100-node topologies. For the third group, 225-node topologies were used.

The detection performance was evaluated considering i) detection rate (DR); ii) false positive rate (FPR); iii) false negative rate (FPR) and iv) detection time median (DTM), indicating the median of the time instances elapsed from the launch of the attack to the instance it was identified. The detection rate is the probability of correctly detect an attack. The false positive rate is the probability of classifying as attack a non-attack event. The false negative rate is the probability of classifying as non-attack an attack event. The detection time median is the median of the number of samples required to

detect the attack.

To calculate those metrics, we classify the events detected as: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). True positive is the result of an attack event that is successfully detected. False positive is the result of a non-attack event that is classified as attack. True negative is the result of a non-attack event that is successfully classified as non-attack. False negative is the result of an attack event that is classified as non-attack event. In this manner, the security metrics are defined as:

- Detection rate:
$$DR = \frac{TP}{TP + FN} \tag{6.1}$$
- False positive rate:
$$FPR = \frac{FP}{FP + TN} \tag{6.2}$$
- False negative rate:
$$FPR = \frac{FN}{FN + TP} \tag{6.3}$$

Such as explained in Section 5.2, we implemented an on-line change point detection algorithm based on Skaperas, Mamatas and Chorti (2019). Since we did not implement the off-line phase to reduce the processing and memory resources required, it was necessary to determine the parameters values that increase or reduce detection performance. Two parameters were investigated: the number of samples for the training period $m = \{100, 150, 200\}$ and the sensitivity $\gamma \in [0, 0.5)$. From that, we introduced a "detection score" to capture the relative importance given to the DR versus the DTM (which focuses on detecting changes in a signal or a time series as quickly as possible after they occur (POOR; HADJILIADIS, 2008)). The proposed detection score, denoted by P_{DS}, was defined as:

$$P_{DS}(A, B) = A(1 - S) + B(DR), \text{ with } A + B = 1, \tag{6.4}$$

where A and B are coefficients that determine the relative weight of each term, and $S = \frac{DTM}{l}$ with l the number of samples monitored after the attack starts. We used five combinations of A and B, i.e., $(A, B) \in \{(1, 0), (0.8, 0.2), (0.5, 0.5), (0.2, 0.8), (0, 1)\}$, to compare the results when prioritizing the speed of detection ($A > B$) versus when prioritizing the detection rate ($A < B$).

The scalability of the proposal was measured based on control packets overhead, data and control packets delivery rate and end-to-end delay. The control packets overhead is the total of control packets transmitted during the simulation time. The control packets delivery rate is the result of dividing the number of control packets successfully received by the total number of control packets sent and similar for the same of the data packets. The end-to-end delay is the average time the packets spent to reach their destination.

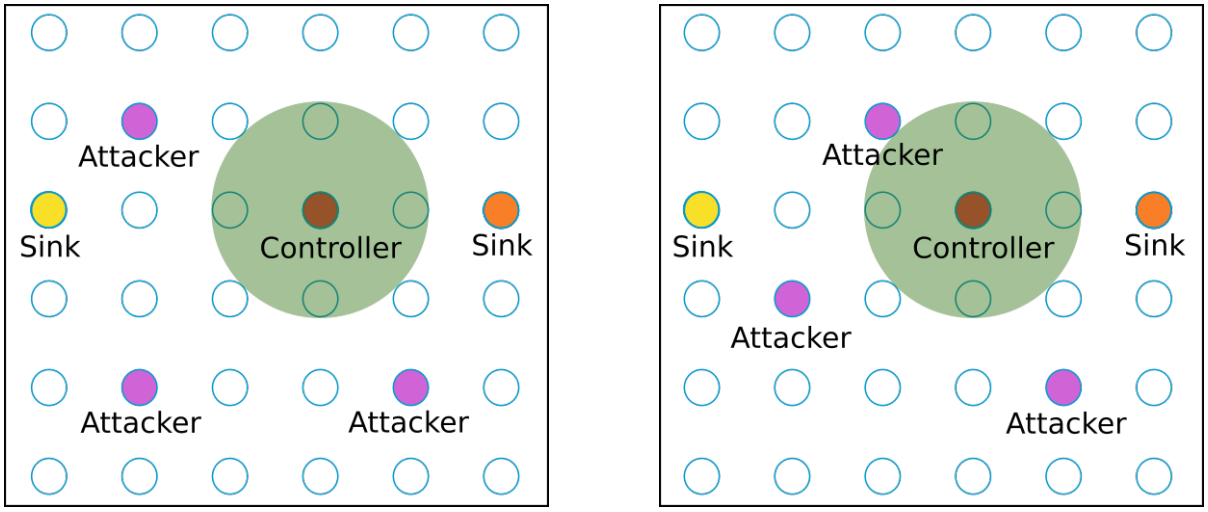
Every node was programmed to periodically generate data and management traffic. For the groups one and two, the sensor nodes send one data packet to a sink node every 30 seconds and one management packet to the other sink every 120 seconds. The data packets delivery rate and the control packets overhead were observed every two minutes, considering the exchange of messages in the whole network during this window of time. The delivery rate was calculated by dividing the number of data packets successfully received by the number of data packets sent. The control packets overhead was quantified as the number of control packets sent. Since samples were collected every two minutes, each simulation was run for 10 hours. During the first eight hours the network operated normally (i.e., for 240 samples there was no change), then the attack was triggered. This imposed a bound $m < 240$.

For the group three, the period set up for both data and management packets was 60 seconds. Using 60 seconds for data traffic we have similar configuration to Alves et al. (2019). This is important since their results are a valid reference for evaluation in terms of scalability. For management traffic, we reduced the period from 120 seconds to 60 seconds. According to the analysis in Section 4.2, the decrease in the data packets delivery rate for using 60 seconds is less than 1% and remains around 95%. Thus, we are able to increase the monitoring metrics sampling by two without a significant impact on the data packets delivery rate. The simulation time was also reduced to 18000 seconds since reducing the management traffic we are able to reduce the monitoring sampling period, thus less time is required for the training phase.

All the scenarios have only one controller and two sinks, one sink exclusive for data traffic and the other one exclusive for management traffic. The controller was placed in the center of the grid and the sinks were placed in the middle of the grid edge, since this location gave a better performance in terms of delay, control overhead, energy consumption and delivery rate according to Alves et al. (2019). The attackers were semi-randomly distributed in the network under the condition that two or more attackers cannot be neighbors and this distribution remains equal in every scenario replication. Figures 24, 25 and 26 show the attackers distribution for 36, 100 and 225 nodes, respectively, when multiple nodes are attackers. The green circle around the controller represents the devices radio range. Notice that no attackers are in the radio range of any sink or controller nodes.

The SDWSN was implemented using IT-SDN [1], without changing the default configuration (Alves et al., 2019). Simulations and IT-SDN main configuration is summarized in Table 3.

[1] Available at http://www.larc.usp.br/users/cbmargi/www/it-sdn/. Accessed: November 15th 2021



(a) FDFF attack (b) FNI attack

Figure 24 – Topology example for 36 nodes with three nodes behaving as attackers: there is one SDN controller, two sinks and three attackers. The green circle represents the radio range of all nodes. No attackers are in the radio range of any sink or controller nodes

Source: Author

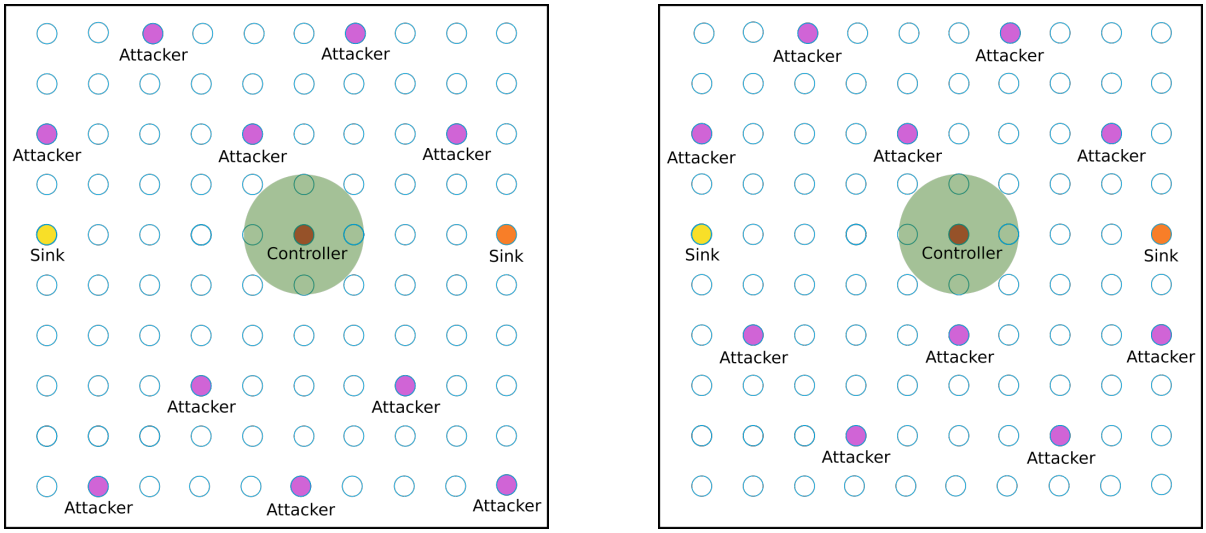
6.2 Results

6.2.1 First group

A dataset comprising 480 simulations was generated, divided in 240 simulations of FNI attacks and 240 simulations of FDFF attacks. Then, each subgroup was split in two sets: one set for parameterization to capture different trade-offs between the detection rate and the speed of detection and the other for validation. In particular, we used the first set to determine the optimal values of $\{m, \gamma\}$. Then, using the values determined for $\{m, \gamma\}$, the CP detector algorithm was executed over the validation sets to evaluate the performance achieved. We varied the number of intruders (attackers) in three proportions: 5%, 10% and 20% of the total of nodes in the network.

The algorithm was executed on the first set for $m \in \{100, 150, 200\}$ and $\gamma \in \{0, 0.15, 0.25, 0.35, 0.45, 0.49\}$ to determine the values that provide the best performance for different trade-offs between the detection rate DR and the detection time median DTM using P_{DS} (Equation 6.4).

During evaluation, two CP detectors were running in parallel. One detector for monitoring the control packets overhead and the other one for monitoring the data packets delivery rate. The validation set comprised both FDFF and FNI attack simulations, 50% of each one, including all chosen topologies and attack intensity levels. In the validation stage we used the optimal pairs (m, γ) identified for each pair (A, B) to maximize the



(a) FDFF attack

(b) FNI attack

Figure 25 – Topology example for 100 nodes with 10 nodes behaving as attackers: there is one SDN controller, two sinks and ten attackers. The green circle represents the radio range of all nodes. No attackers are in the radio range of any sink or controller nodes

Source: Author

metric $P_{DS}(A, B)$. Whenever a CP was detected, we stopped the detectors, declared the network under attack, and determined which metric triggered the detector. If the detector monitoring the control overhead was triggered first, we declared an FDFF attack, alternatively, if the detector monitoring the data packet delivery rate was triggered first, we declared an FNI type of attack.

6.2.1.1 Optimizing m and γ

The main objective of these experiments was to determine the parameters $\{m, \gamma\}$ that could provide the best detection performance. P_{DS}. was calculated for all topologies, attack scenarios and combinations of m and γ. Then, the results for $\alpha \in \{0.90, 0.95, 0.99\}$ were analyzed. The first results showed that in 90% of all cases P_{DS} was maximized when $m = 200$, turning this value an universally optimal choice and the m value used for the remaining of the analysis. This means that, when running the online detector, no training is required other than the observation of 200 samples of normal network operation.

For the next part, the results were separated grouping each attack based on the monitoring metric: for the FDFF attack, the control overhead CP detection results were analyzed, while for a FNI attack the data packets delivery rate, CP detection results were analyzed based on Segura et al. (2020). The average value of P_{DS} as a function of γ and α for the case of a FDFF attack is depicted in Figure 27. Figure 27a shows that when prioritizing faster detection (i.e. $A = 1$) the higher values of P_{DS} are obtained for

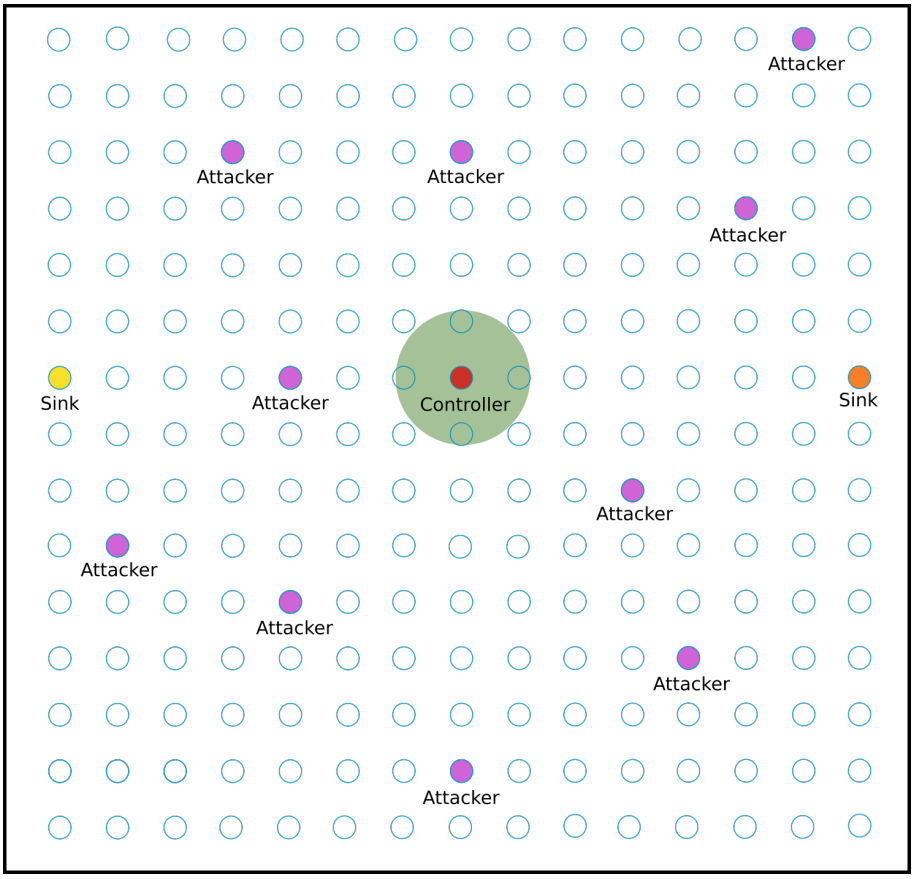


Figure 26 – Topology example for 225 nodes with 10 nodes behaving as attackers: there is one SDN controller, two sinks and ten attackers. The green circle represents the radio range of all nodes. No attackers are in the radio range of any sink or controller nodes

Source: Author

$\gamma = \{0.35, 0.45\}$ and the lowest for $\gamma = \{0, 0.15\}$. In turn, Figure 27c shows that prioritizing the detection rate, the higher values of P_{DS} were obtained for $\gamma = \{0, 0.15, 0.25\}$, reaching $P_{DS} = 1$. Lastly, if $A = B = 0.5$, which means we do not prioritize nor faster detection neither detection rate, higher values of P_{DS} were for $\gamma = \{0.25, 0.35\}$ and its curve is flatter than the other two cases.

The average value of P_{DS} for the case of a FNI attack is presented in Figure 28. Contrary to the results in Figure 27, in this case the trend was not as clear because lower values of γ maximized P_{DS} when $A = B = 0.5$ and $B = 1$, i.e., the detection rate influences P_{DS} more than the detection speed. This is also supported by the results in 28a where we observe that considering only the detection speed component (e.g. $A = 1$) the curve is flatter, varying from 0.88 to 0.91 when $\alpha = 0.99$ and varying from 0.90 to 0.93 when $\alpha = \{0.95, 0.90\}$. Conversely, the results in 28a, considering only the detection rate component, P_{DS} varies from 0.75 to 0.95 approximately.

By varying γ, we were able to configure the detector to prioritize faster detection

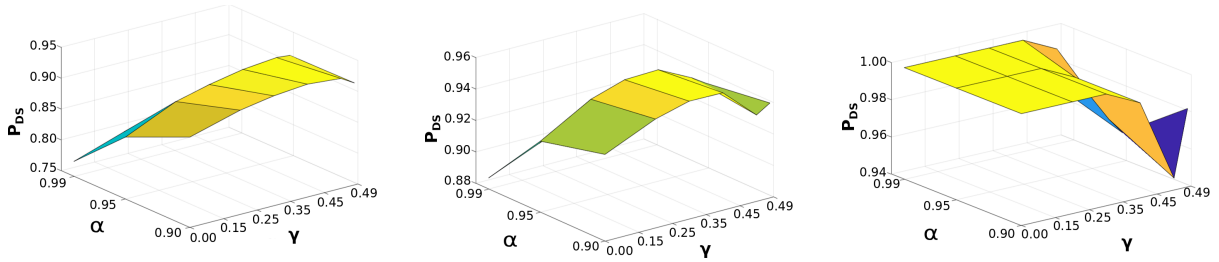
Table 3 – Simulation parameters

Simulation parameters	
Topology	Square grid
Number of nodes	$\{36, 100, 225\}$
Simulation time (s)	$\{18000, 36000\}$
Node boot interval (s)	$[0, 1]$
Number of sinks	2
Sinks position	Middle of the grid edge
controller position	center
Data traffic rate	1 packet every 30 seconds
Management traffic rate	1 packet every two minutes
Data payload size	10 bytes
Management payload size	10 bytes
Data traffic start time	$[2, 3]$ min
Radio module power	0 dB
Distance between neighbors	50 m
Attacks start after (s)	$\{13200, 28800\}$

IT-SDN parameters	
Controller retransmission timeout	60 s
ND protocol	Collect-based
Link metric	ETX
Neighbor report max frequency	1 packer per minute
CD protocol	none
Flow setup	source routed
Route calculation algorithm	Dijkstra
Route recalculation threshold	10%
Flow setup types	regular or source routed
Flow table size	10 entries

Source: Author

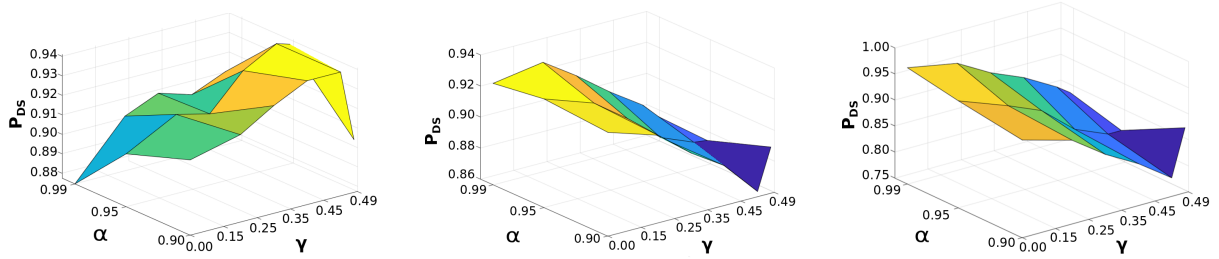
or accuracy. However, the response was different for both attacks. In Table 4, the values of γ that maximize P_{DS} are shown. In cases where more than one value resulted in the same or very similar values, one was chosen arbitrarily.



(a) $A = 1$ and $B = 0$ (b) $A = B = 0.5$ (c) $A = 0$ and $B = 1$

Figure 27 – Metric P_{DS} in function of γ and α for FDFF attack: (a) shows P_{DS} when prioritizing quickest detection, (b) shows P_{DS} when giving the same weight to detection speed and detection rate, and (c) shows P_{DS} when prioritizing detection rate

Source: Author



(a) $A = 1$ and $B = 0$ (b) $A = B = 0.5$ (c) $A = 0$ and $B = 1$

Figure 28 – Metric P_{DS} in function of γ and α for FNI attack: (a) shows P_{DS} when prioritizing quickest detection, (b) shows P_{DS} when giving the same weight to detection speed and detection rate, and (c) shows P_{DS} when prioritizing detection rate

Source: Author

6.2.1.2 Centralized detector performance

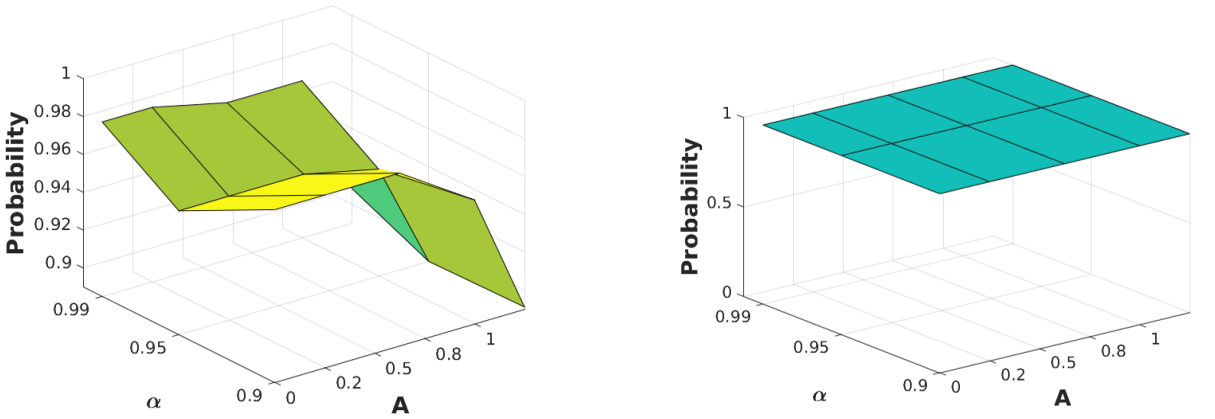
For this part, two detectors were set running simultaneously, one monitoring the control overhead and the other one monitoring the data packets delivery rate. Both detectors were configured according to the values in Table 4 and using $m = 200$. The first experiment was devised to identify the type of the attack based on the first detector triggered. The probability of the control overhead CP detector being triggered first in case of a FDFF attack is depicted in Figure 29a. These results showed that in the worst case the detector monitoring the control overhead has a probability between 0.89 and 0.98 of being triggered first in case of FDFF attack. In case of FNI attack, the detector monitoring the data packets delivery rate was triggered first in 100% of the events, as shown in Figure 29b. These results showed that there is evidence to support the hypothesis drawn up in our previous works about the relation metric / attack (SEGURA et al., 2020).

Next, the detection performance using the parameters that maximize P_{DS} was

Table 4 – γ that maximizes P_{DS}

P_{DS}	γ		
	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.99$
Best γ for control overhead CP detector			
$A = 1$ and $B = 0$	0.45	0.45	0.45
$A = 0.8$ and $B = 0.2$	0.35	0.35	0.45
$A = 0.5$ and $B = 0.5$	0.25	0.35	0.45
$A = 0.2$ and $B = 0.8$	0.25	0.25	0.35
$A = 0$ and $B = 1$	0	0	0
Best γ for delivery rate CP detector			
$A = 1$ and $B = 0$	0.45	0.45	0.45
$A = 0.8$ and $B = 0.2$	0	0.15	0.15
$A = 0.5$ and $B = 0.5$	0	0	0.15
$A = 0.2$ and $B = 0.8$	0	0	0
$A = 0$ and $B = 1$	0	0	0

Source: Author



(a) Probability of control overhead CP detector being triggered first in case of FDFF attack

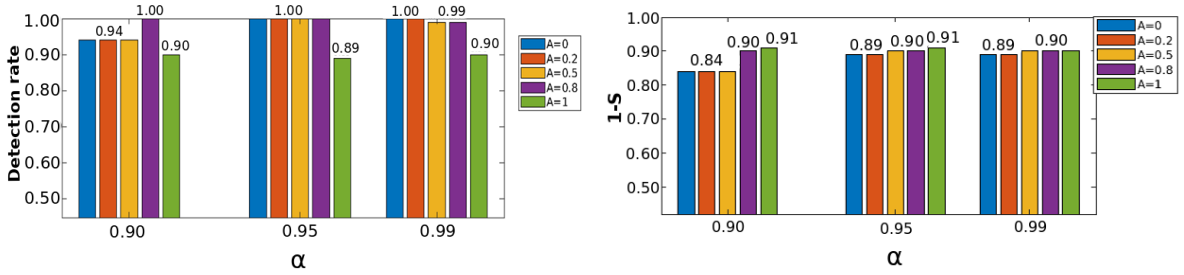
(b) Probability of data packets delivery rate CP detector being triggered first in case of FNI attack

Figure 29 – Attack type identification probability

Source: Author

analyzed. Figure 30 depicts the detection rate DR and the metric $1 - S$ when the network is under FDFF attack. In terms of detection rate, four out of five configurations achieved $DR \geq \alpha$. In terms of detection speed, $1 - S \geq 0.90$ for $A = \{0.8, 1\}$. Considering both DR and $1 - S$ the results for $A = 0.8$ provided the best trade off.

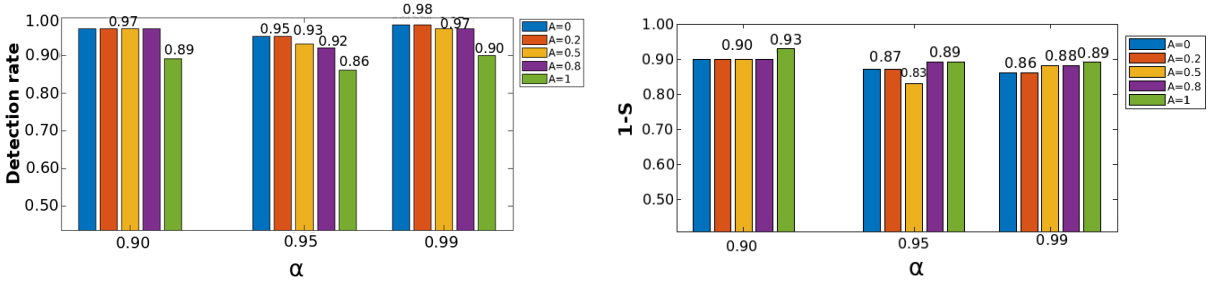
Figure 31 shows the detection rate and the detection speed metrics for the FNI attack using the identified values of γ. In this case, four out of five configurations achieved $DR \geq \alpha$ for $\alpha = 0.90$; two out of five configurations achieved $DR \geq \alpha$ for $\alpha = 0.95$; and



(a) Detection rate (b) Detection speed

Figure 30 – Detection performance of FDFF attack using γ and m values that optimize P_{DS} for five different cases of $\{A, B\}$

Source: Author



(a) Detection rate (b) Detection speed

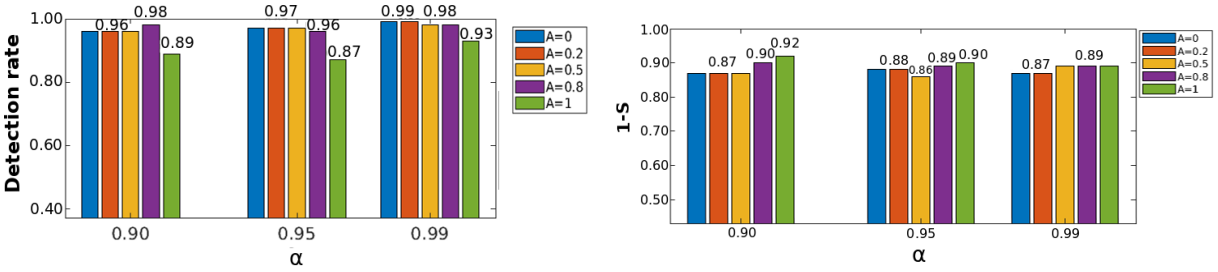
Figure 31 – Detection performance of FNI attack using γ and m values that optimize P_{DS} for five different cases of $\{A, B\}$

Source: Author

zero configurations achieved $DR \geq \alpha$ for $\alpha = 0.99$. In terms of detection speed, $A = 0$ obtained the fastest detection, as intuitively expected based on the results from Figure 28. Comparing the results for $A = 1$ and $A = 0$, we can maximize DR at the cost of 0.03 in $1 - S$, which is equivalent to 1.8 samples. On the other hand, if we are looking for fastest detection, DR drops to 0.90 or below.

Next, the detection performance irrespective of the type of attack was analyzed, when both detectors were running simultaneously. The results in Figure 32 showed a detection rate over α when $A = \{0, 0.2, 0.5, 0.8\}$ for $\alpha = \{0.90, 0.95\}$; when $\alpha = 0.99$, $DR = \alpha$ for $A = \{0, 0.2\}$ only. As a conclusion, to maximize the detection rate the configuration for $A = \{0, 0.2\}$ should be used. In terms of detection speed, as shown in Figure 32b, to maximize the detection rate means a lag of 3 samples on average with respect to the fastest detection result obtained.

FDFF and FNI attacks detection using the complete version of Skaperas, Mamatas and Chorti (2019) algorithm was analyzed (SEGURA et al., 2020) in a previous work. Tables 5 and 6 compare the results using the complete version and the simplified version



(a) Detection rate (b) Detection speed

Figure 32 – Detection performance of FDFF and FNI attacks using γ and m values that optimize P_{DS} for five different cases of $\{A, B\}$

Source: Author

proposed here for $\alpha = 0.95$, $A = B = 0.5$ (simplified) when 5% of nodes behaves as attackers for 36 and 100 nodes, respectively. The simplified version obtained better detection accuracy while the complete version was quicker, excluding the FNI attack for 100 nodes. From these results, we noticed that even using a simplified version of the original algorithm, the right selection of the parameters compensates the lack of the off-line phase.

Table 5 – Complete versus simplified change point detection algorithm: 36 nodes when 5% behave as attacker

Algorithm	**DR**	**FPR**	**FNR**	**DTM**
FDFF: control packets overhead				
Complete version	0.97	0.03	0.00	9
Simplified	1.00	0.00	0.00	14
FNI: data packets delivery rate				
Complete version	0.80	0.13	0.07	7
Simplified	0.92	0.07	0	9

Source: Author

Table 6 – Complete versus simplified change point detection algorithm: 100 nodes when 5% behave as attacker

Algorithm	**DR**	**FPR**	**FNR**	**DTM**
FDFF: control packets overhead				
Complete version	1.00	0.00	0.00	10
Simplified	1.00	0.00	0.00	11
FNI: data packets delivery rate				
Complete version	0.83	0.17	0.00	6
Simplified	0.92	0.07	0.00	5

Source: Author

Summarizing Section 6.2.1, we chose the pairs (m, γ) that maximized the detection performance metric P_{DS}. Our results showed that in 90% of cases $m = 200$ maximized

P_{DS}. With respect to γ, it was observed that using $\gamma = \{0.45, 0.49\}$ reduced the time to detect the attack but this had an adverse effect on the detection rate. Conversely, when $\gamma = \{0, 0.15\}$ the detection rate was maximized at the cost of delaying the detection. Then, the CP detectors using the parameters values chosen before were tested. The results showed that it was possible to detect the attack with $DR \geq \alpha$ when $B > A$. Yet, prioritizing the fastest detection, the DR drops to 0.93 or below. In terms of detection rate, this proposal and our previous proposal (SEGURA et al., 2020) have a similar performance. Moreover, here we provide concrete evidence to support the relation between monitored metric and the type of attack.

6.2.2 Second group

A dataset of 120 simulations divided into two groups was generated, divided in two parts so that half concerned for a FDFF attack and the other half for a FNI attack. For both attacks, grid topologies of 36 and 100 nodes were simulated, whereby 10% of nodes were attackers. For these experiments, we prioritized detection accuracy over detection speed and thus the detector was configured using $\gamma = 0$ while we set the target $\alpha = 0.99$, and $m = 200$ according to the results from Section 6.2.1.1.

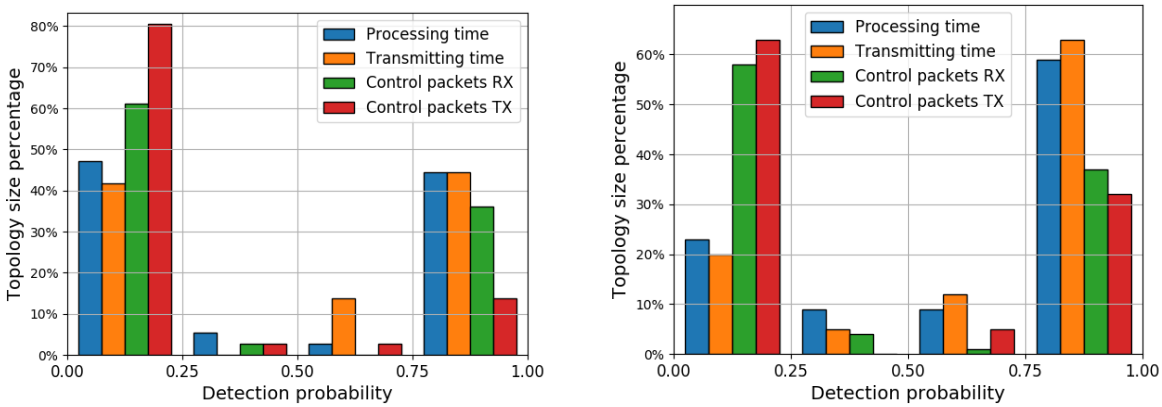
The detection performance was evaluated on every node monitoring each metric (processing time, transmitting time, control packets received and control packets transmitted) separately, i.e., running only one detector at time due to memory constraints on the nodes. For this evaluation the detection probability of every node on each scenario was calculated. The same simulation parameters and attackers positions used for the centralized detection experiments, were maintained.

The detection performance analysis is based on three perspectives under the condition the network is under attack: (i) probability of CP detection on each node; (ii) percentage of nodes reporting with high detection rates; (iii), and location of nodes reporting high detection rates.

6.2.2.1 Results for FDFF attack

The detection probability distribution when the network is under FDFF attack for 36 and 100 nodes is shown in Figure 33. In the case of 36 nodes, as shown in Figure 33a, more than 40% of the nodes detected an anomaly on their processing and transmitting time in more than 75% of times the network was under a FDFF attack. The percentage of nodes that detected an anomaly on these two metrics in more than 75% of the events grew up to values around 60 for the case of 100 nodes (Figure 33b).

Next, we further zoomed in detection probabilities greater than 0.90, shown in Figure 34. This value is important since works with better detection results reported detection probabilities over 0.90 (Chapter 3). For the case of 36 nodes, more than 30% of



(a) 36 nodes (b) 100 nodes

Figure 33 – Detection probability distribution of FDFF attack: Comparison of detection probability when monitoring the processing time, transmitting time, control packets received, and control packets transmitted.

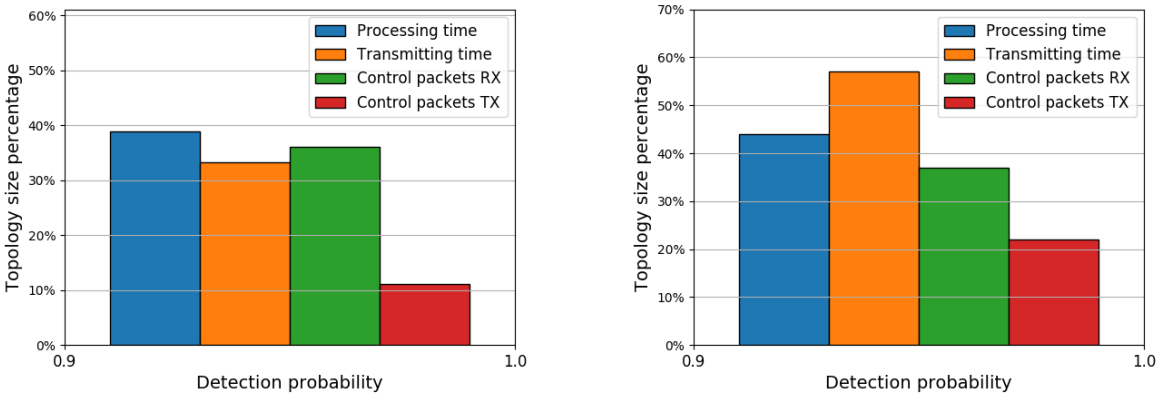
Source: Author

nodes reported an alarm, in at least 90% of events, when monitoring either processing time, transmitting time or control packets received. The case monitoring the control packets transmitted was the worst result in terms of percentage of nodes reporting an anomaly. In the hypothetical case where we are running an intrusion detection policy based on the number of alarms received by the controller, we reduced the detection chances if monitoring the control packets transmitted instead of any of the other three metrics.

The percentage of nodes reporting an alarm increases in general with the network size, obtaining the highest result when monitoring the transmitting time and the lowest result when monitoring the control packets transmitted. Opposite to the results for 36 nodes, where the percentage of nodes reporting an alarm was similar when monitoring either the processing time, the transmitting time, or the control packets received, for 100 nodes there was a clear difference among the results for all the metrics. The percentage of nodes when monitoring the transmitting time was 12% over the results when monitoring the processing time, and 20% over the results when monitoring the control packets received.

In brief, for 36 nodes the percentage of nodes reporting alarms with $P_{DR} \geq 0.90$ was similar when monitoring either the processing time, transmitting time, or control packets received. However, for 100 nodes the results when monitoring the transmitting time were clearly over the results when monitoring any of the other metrics. In the hypothetical case where the nodes have resources to monitor only one metric, the transmitting time is the one that provides the best trade off in terms of percentage of nodes reporting an alarm.

Subsequently, the position of nodes in the network and their detection probabilities were analyzed. The heat maps for 36 nodes when monitoring the transmitting time and



(a) 36 nodes (b) 100 nodes

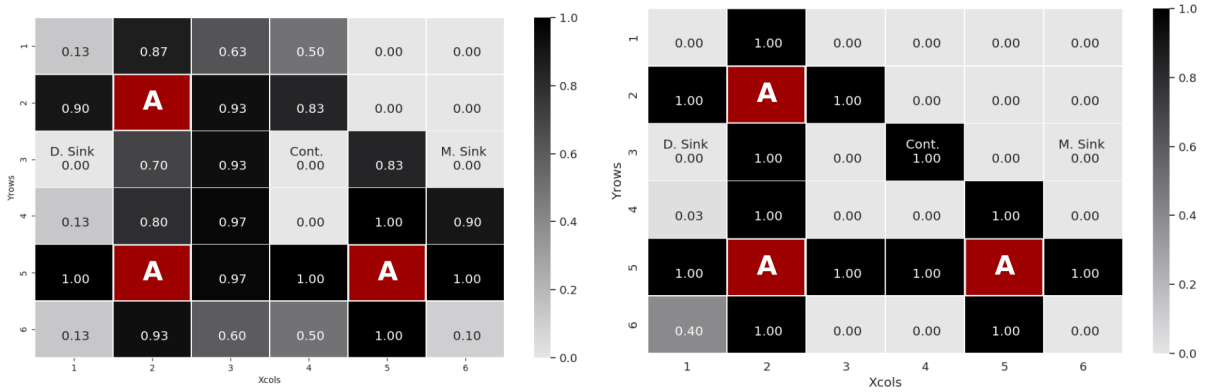
Figure 34 – Percentage of nodes with detection probability larger than 90% for the FDFF attack: Comparison of detection probability when monitoring the processing time, transmitting time, control packets received, and control packets transmitted.

Source: Author

the control packets received are shown in Figure 35. From these results we make two observations: i) in the case monitoring the transmitting time, the neighbors of the attackers had higher detection probability than nodes farther; and ii) in the case monitoring the control packets received, excluding the controller and the node on the lower left corner, all nodes reporting an alarm were in the attacker's neighborhood and had a $P_{DR} = 1$. For 100 nodes, a similar behavior was observed when monitoring the control packets received (Figure 36b), but when monitoring the transmitting time (Figure 36a) the high detection probability is not exclusive for attackers' neighbors and it is spread all over the topology. This happened because when the network was under attack, the number of control packets increased and this impacted the radio usage of all nodes forwarding those packets. On the other hand, the control packets received is a metric that impacts only the node that receives the packet. In Subsection 6.2.2.3 we explore how to use node's location and address to identify the attackers.

6.2.2.2 Results for FNI attack

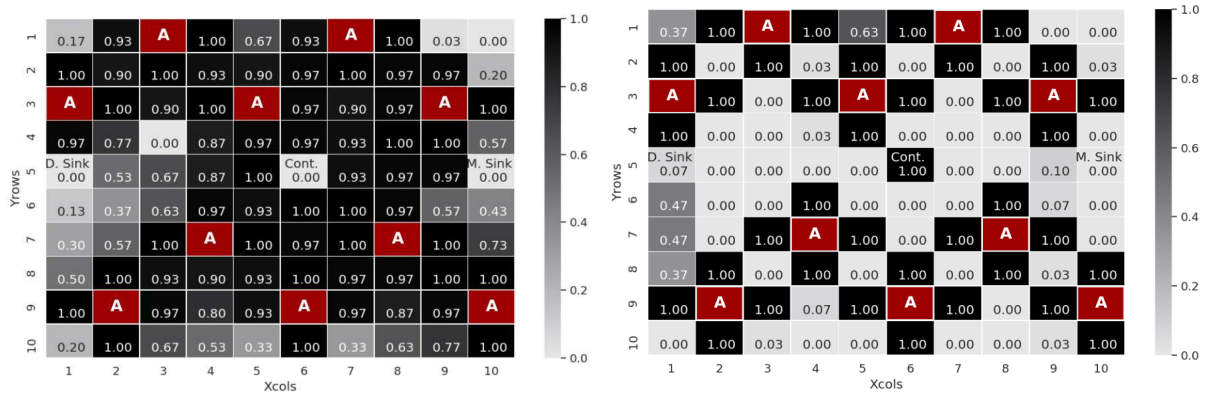
Figure 37 shows the detection probability density distribution results when the network was under an FNI attack. For 36 nodes (Figure 37a) we observed a similar behavior for all four metrics: high density in probabilities around 0 and 0.25 that decreased as the detection probability grows up, being the result for control packets received the one with highest density in probabilities over 0.50. In the case of 100 nodes, the results for control packets transmitted maintained the behavior observed for 36 nodes, with high density in probabilities between 0 and 0.20 that decreased for higher probabilities. The results for



(a) Transmitting time (b) Control packets received

Figure 35 – Detection probability heat maps for 36 nodes when the network is under FDFF attack. Each square represents a node in the network and the number inside them is the detection probability result. The red squares with an "A" inside are the attackers. (a) shows the results when monitoring the transmitting time and (b) shows the results when monitoring the control packets received.

Source: Author



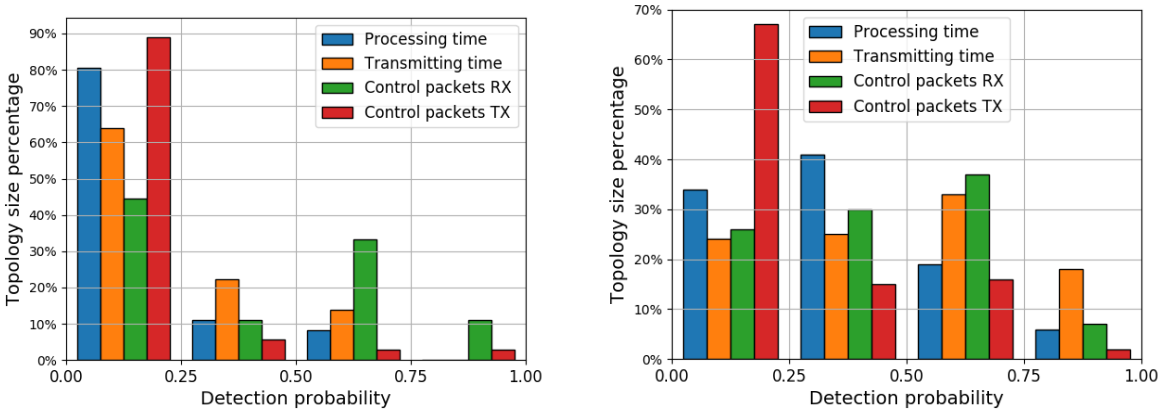
(a) Transmitting time (b) Control packets received

Figure 36 – Detection probability heat maps for 100 nodes when the network is under FDFF attack. Each square represents a node in the network and the number inside them is the detection probability result. The red squares with an "A" inside are the attackers. (a) shows the results when monitoring the transmitting time and (b) shows the results when monitoring the control packets received.

Source: Author

processing time, transmitting time, and control packets received showed high detection probability density around 0.25 and 0.50. Then, for detection probabilities over 0.90, the highest density was for the transmitting time. The reason why we observed more impact on the transmitting time and the control packets received is because this attack leads to a network reconfiguration using wrong neighborhood information. First, the network reconfiguration means several control packets from the controller to the nodes, which increases this metric on these nodes. Then, since the reconfiguration is based on wrong

information, the number of packets retransmission increases, increasing the transmitting time metric as well.



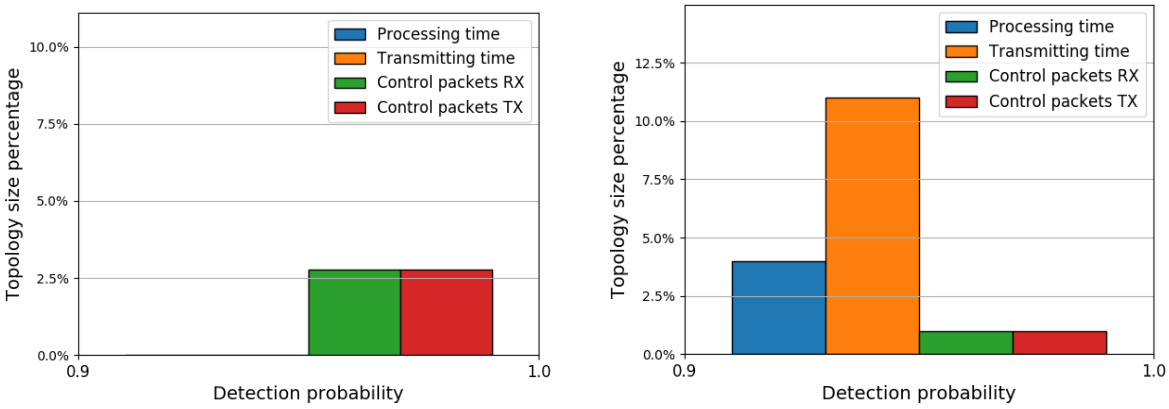
(a) 36 nodes (b) 100 nodes

Figure 37 – Detection probability distribution of FNI attack: Comparison of detection probability when monitoring the processing time, transmitting time, control packets received, and control packets transmitted. The "x" axis represents the detection probability divided in four groups: $[0, 0.25)$, $[0.25, 0.50)$, $[0.50, 0.75])$, and $[0.75, 1]$. The "y" axis represents the percentage of the total nodes that obtained this detection probability. (a) shows the results for 36 nodes and (b) shows the results for 100 nodes.

Source: Author

To confirm previous results, the percentage of nodes reporting an alarm with probabilities $P_{DR} \geq 0.90$ was calculated. Figure 38 shows these results for 36 and 100 nodes. For 36 nodes, 2.7% of nodes obtained a $P_{DR} \geq 0.90$ when monitoring either the control packets received or the control packets transmitted. Since 2.7% represent less than one node, we consider that our distributed proposal is not able to detect an FNI attack in a small topology with a probability above 0.90. For 100 nodes, the highest result was for the case monitoring the transmitting time, where 11% of nodes obtained a $P_{DR} \geq 0.90$. The percentage of nodes reporting an alarm when monitoring the transmitting time with $P_{DR} \geq 0.90$ is higher for 100 nodes than for 36 nodes because of two reasons: there were more nodes using an attacker to reach the controller, which increased the percentage of nodes affected; and the distance between the attackers and the controller was larger for 100 nodes, which means more nodes participated forwarding packets for network reconfiguration. In summary, transmitting time is the only metric where a FNI attack could be detected by multiple nodes, but up to this point, it seems it does not work in small-medium networks, i.e. around 36 nodes.

Notwithstanding the detection performance of FNI attack, in Figure 33 a high density of nodes reporting alarms in probabilities over 0.50 when monitoring the control



(a) 36 nodes (b) 100 nodes

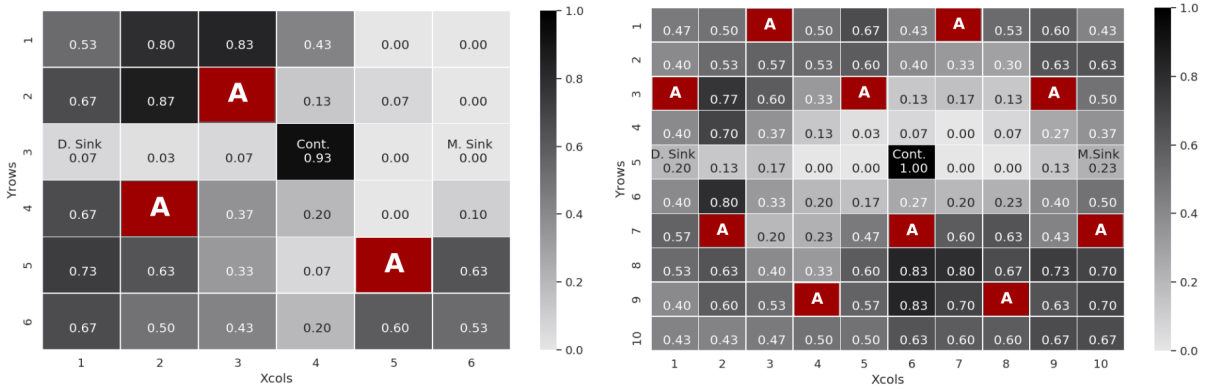
Figure 38 – Percentage of nodes with detection probability of FNI attack larger than 90%: Comparison of detection probability when monitoring the processing time, transmitting time, control packets received, and control packets transmitted. The "y" axis represents the percentage of the total nodes with high detection probability. (a) shows the results for 36 nodes and (b) shows the results for 100 nodes.

Source: Author

packets received and the transmitting time was observed. Consequently, we decided to investigate the location of those nodes in the topology. We observed that in the cases monitoring the control packets received, as shown in Figure 39, some nodes around the attackers concentrated the higher detection probability values, but others also in the attacker´s neighborhood had detection probabilities around zero. The question arises as to why this is observed; the reason being that neighboring nodes with higher detection probabilities used the attacker to route their packets toward the controller, thus the network misconfiguration reached them first.

From these results, a second strategy based on data aggregation was motivated, analyzing CP detection per regions (areas). To this end, we divided the 36 nodes in four groups and the 100 nodes in nine groups and created one time series per group. Each sample of this time series represented the sum of time series of all nodes in the group, thus we executed one CP detector per group. Figure 40 shows P_{DR} results for 36 and 100 nodes when monitoring the control packets received. Excluding the groups that contained the controller, in all cases the detection probability achieved is better than the one obtained by any of the nodes individually. This indicates that with data aggregation we lose granularity but we gain in detection rates.

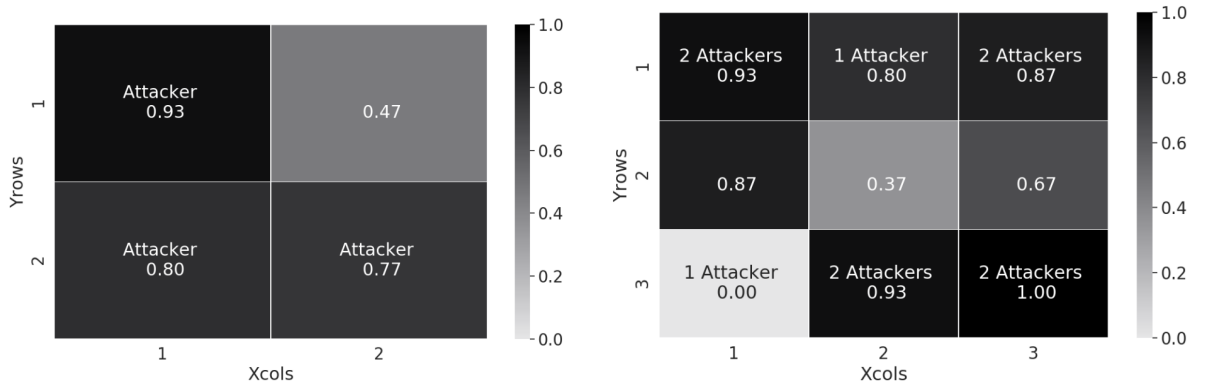
Summarizing Subsection 6.2.2, the distributed anomaly detection proposal was evaluated on networks under FDFF and FNI attack, monitoring four metrics obtained from each node: processing time, transmitting time, control packets received, and control



(a) 36 nodes (b) 100 nodes

Figure 39 – Detection probability heat maps when the network is under FNI attack. Each square represents a node in the network and the number inside them is the detection probability result. The red squares with an "A" inside are the attackers. (a) shows the results for 36 nodes and (b) shows the results for 100 nodes.

Source: Author



(a) 36 nodes (b) 100 nodes

Figure 40 – Detection probability heat maps when the network is under FNI attack. Each square represents a group of nodes and the number inside them is the detection probability result aggregating the control packets received information of all nodes in the group. (a) shows the results for 36 nodes and (b) shows the results for 100 nodes.

Source: Author

packets transmitted. Our results showed that in case of FDFF attack, at least 33% of the total of nodes obtained a detection probability equal or over 90% when monitoring the processing time, the transmitting time, or the control packets received. Also, the location of the nodes with higher detection probability coud be used to determine the node launching the attack. In the cases when the network was under a FNI attack we had two scenarios: i) for 36 nodes, 2.5% of nodes obtained a detection probability equal or over 90% when monitoring the control packets received and the control packets transmitted; and ii) for

100 nodes we obtained the best result when monitoring the transmitting time, where 11% of nodes reported a detection probability equal or over 90%. Since the results for the FNI attack were not satisfactory, we introduced a second strategy based on data aggregation. Our results showed that using this strategy we increased the detection probability but lost in granularity.

6.2.2.3 Attacker detection

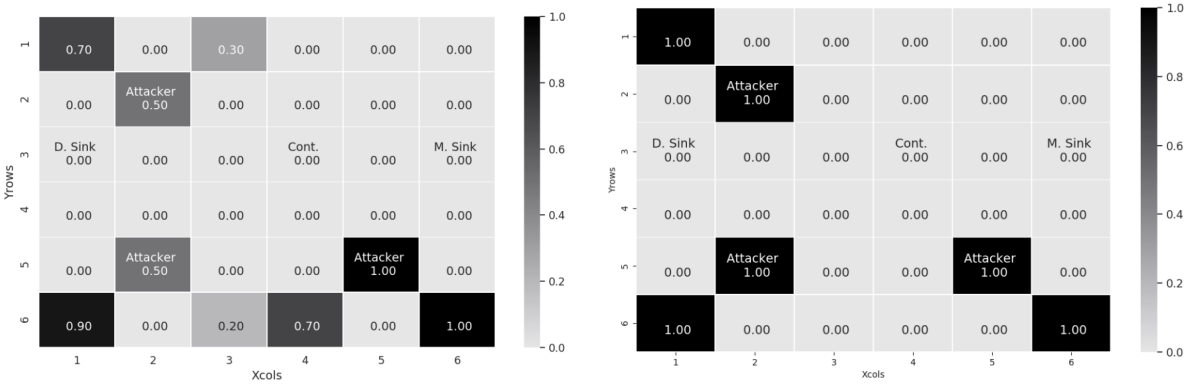
The results discussed in Subsections 6.2.1 and 6.2.2 showed that the CP detectors for DoS attacks worked for both centralized and distributed detection, but also we observed that the distributed detection provides information that infers the attackers' location. In this direction, our proposal explores the SDN characteristics by using the controller global view of the network to identify the attacker's address or location based on the alarms reported by the nodes. In this section we present the identification results when the network is under an FDFF or FNI attack. The analysis is separated by the type of attack.

6.2.2.3.1 Attacker detection in FDFF attack

First, we started testing Algorithm 1. Figure 41 depicts the attacker identification results for 36 nodes when monitoring the transmitting time and the control packets received. The heat map shows the probability that each node has of being identified as attacker. We observed that for the case monitoring transmitting time (Figure 41a), in addition to the three attackers, seven benign nodes were identified as attackers as well. The probabilities of those nodes being misidentified as attackers ranged from 0.10 to 1.00, which means that some nodes were misidentified in all cases. In the case monitoring the control packets received (Figure 41b), all the attackers were correctly identified in all cases. On the other hand, 3 more nodes were misidentified in all the cases as well.

We observed that the main problem of our identification algorithm was on the corners of the grid. This is because the corners have only two neighbors, and those neighbors are also in the attackers' neighborhood. This means, all the times our algorithm identified the attacker, automatically the corners were misidentified as attackers as well. The next step was to evaluate the performance of Algorithm 2.

As shown in Figure 42, monitoring either the transmitting time or the control packets received, no misidentifications were registered. When monitoring the control packets received the identification probability was 1.00 for all the attackers, while when monitoring the transmitting time the identification probability was between 0.85 and 1.00. When evaluating the identification algorithm for 100 nodes (Figure 43) we obtained excellent results as well; no misidentifications and identification probabilities over 0.93. In fact, when monitoring the control packets received the identification probability was 1.00 for all the attackers. The main benefit of Algorithm 2 is that by reporting suspects, the

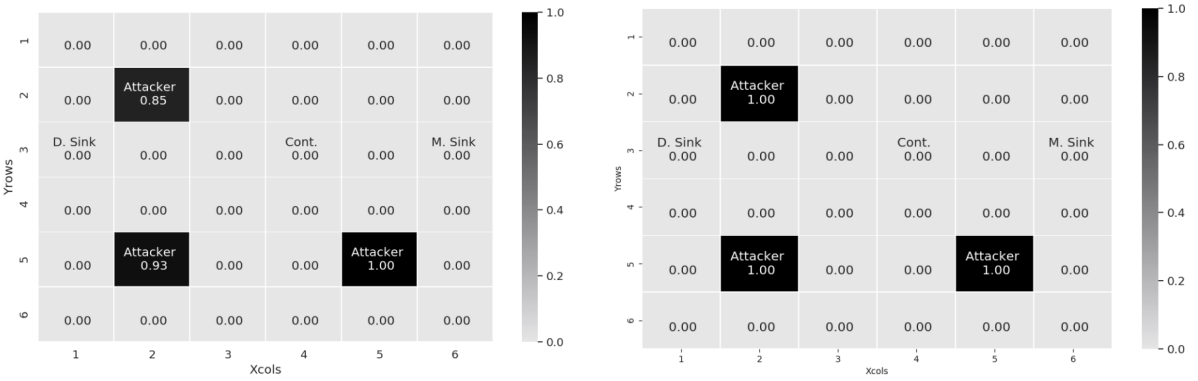


(a) Transmitting time (b) Control packets received

Figure 41 – Attackers identification probability using Algorithm 1 when the network is under an FDFF attack: case of 36 nodes. Each square in the map represents a node in the network. The number in the squares represent the probability of this node being classified as attacker.(a) shows the results when monitoring the transmitting time and (b) shows the results when monitoring the control packets received

Source: Author

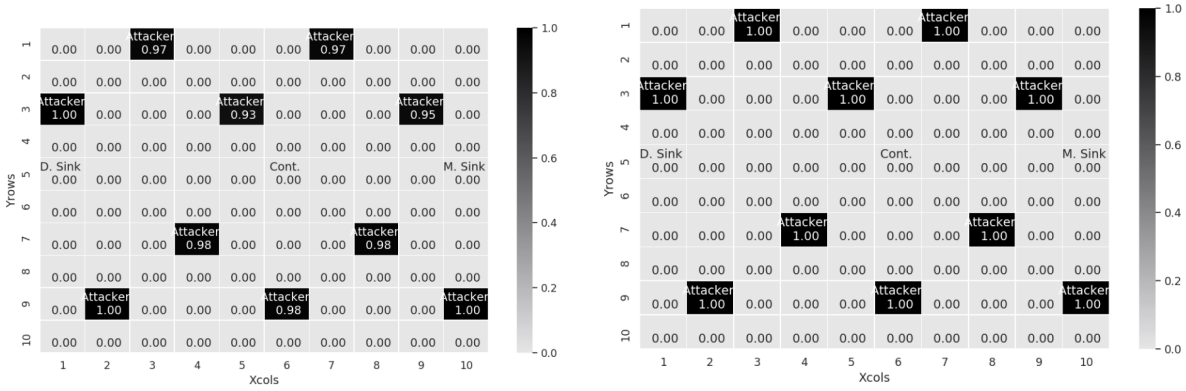
Security Manager can discard possible suspects and focused only on the ones with higher probability. However this algorithm is very tight to FDFF characteristics.



(a) Transmitting time (b) Control packets received

Figure 42 – Attackers identification probability using Algorithm 2 when the network is under an FDFF attack: case of 36 nodes. Each square in the map represents a node in the network. The number in the squares represent the probability of this node being classified as attacker.(a) shows the results when monitoring the transmitting time and (b) shows the results when monitoring the control packets received

Source: Author



(a) Transmitting time (b) Control packets received

Figure 43 – Attackers identification probability using Algorithm 2 when the network is under an FDFF attack: case of 100 nodes. Each square in the map represents a node in the network. The number in the squares represent the probability of this node being classified as attacker.(a) shows the results when monitoring the transmitting time and (b) shows the results when monitoring the control packets received

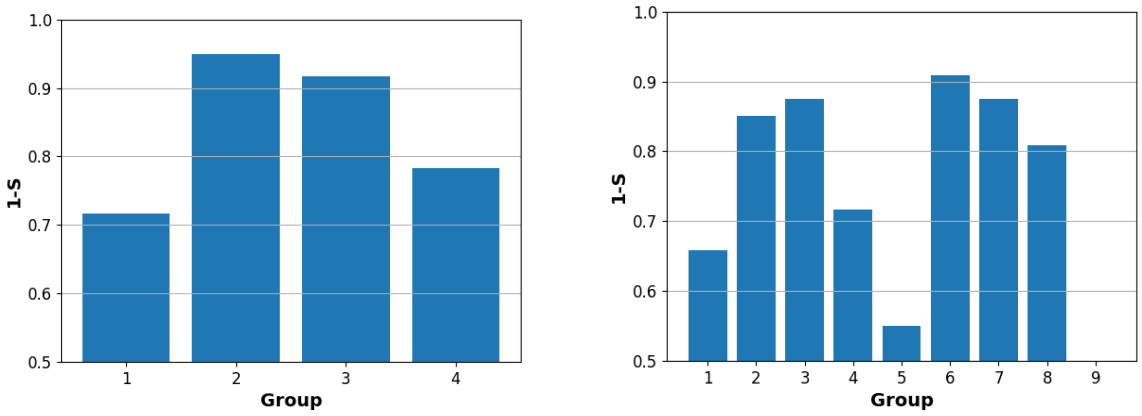
Source: Author

6.2.2.3.2 Attacker detection in FNI attack

The results in subsection 6.2.2.2 showed that, for the case of FNI attacks, the percentage of nodes with high detection probability was low and also not all the nodes around the attackers detected the attack, opposite to the observed for the FDFF attack. Because of this, the attacker detection based on data aggregation was evaluated. Our objective was to, at least, identify the area where the attacker was located.

From Figure 40 we noticed that our FNI detection strategy based on data aggregation increased the detection probability if compared with our initial approach, running the detector on every node. However, the data aggregation strategy results show detection probabilities over 37% in areas without attackers. The first impression is that, in the case we track the attackers based on the alarm received from one group, this could lead to false positives because of the detection probability in areas without attackers. Thus, we analyzed the detection speed on every group.

Figure 44 shows the $1 - S$ metric (normalized DMT) for 36 and 100 nodes when monitoring the control packets received. For 36 nodes (Figure 44a), our results showed that group 1 (the group without an attacker) obtained the lowest $1 - S$, which means this is the last group reporting an alarm. Now, suppose we trust on the hypothesis that the groups reporting an alarm has an attacker within it. We may execute a mitigation strategy targeting the firsts groups reporting an alarm because, as shown in Figure 40a, these groups have the highest detection probability. In the best scenario, a successfully mitigation will stop the attack before the detectors on the groups without attackers detect



(a) 36 nodes (b) 100 nodes

Figure 44 – Detection speed (1-S metric) for FNI detection by data aggregation when monitoring the control packets received. (a) shows the results for 36 nodes and (b) shows the result for the case of 100 nodes

Source: Author

a change. However, in the case of 100 nodes (Figure 44b) the results did not show a similar trend. Groups 4, 5, and 6 are the ones without an attacker within it; interestingly, group 6 had the highest $1-S$. Conversely, group 9 has one attacker and did not report alarms. This occurred because of two reasons: i) the attacker in group 9, which is in the border with group 6, had impact on group 6, and ii) the data sink in group 6 increased the packets traffic which accelerated the change in the time series.

Summarizing, Algorithm 1 was proposed to identify attackers when the network is under FDFF attack, but results showed this algorithm leads to false positive cases. Then, Algorithm 2 was proposed and results showed it solved the false positive cases and improved attackers identification. Attacker identification probability is over 0.93, when monitoring the transmitting time, equal to 1 when monitoring the control packets received. Conversely, for the FNI attack, we analyzed the attackers identification based on the data aggregation detection results when monitoring the control packets received. We noticed that for 36 nodes, the groups with attackers within it reported the alarm faster than the group without attackers. On the other hand, for 100 nodes we did not observe a reliable relation between any metric and the presence of attackers in the groups.

6.2.3 Third group

The results of the cooperative intrusion detection in networks with 225 nodes are shown in this Subsection. The centralized monitoring in the Security Manager was set to 60 seconds while the distributed monitoring (i.e. on every individual node) was varied between 60 and 10 seconds. Transmitting time was the main metric used for the distributed approach, since monitoring this metric we maximize the number of nodes

detecting anomalies on large networks (e.g. Figures 33b and 37b). However, monitoring this metric increases the processing resources usage in 5%, sampling every 60 seconds, and 11% sampling every 10 seconds. The whole anomaly detection for one metric requires around 7.2 KB, which is equivalent to 15% of the total flash memory size.

The proposal was tested on FDFF and FNI attacks, varying the number of attackers among one, four and ten. According to Nooribakhsh and Mollamotalebi (2020), change point detection accuracy depends on a sudden change on the metric behavior. Thus, the attackers were programmed to trigger the attacks every 60 or 10 seconds. This frequency is slow compared to other DoS attacks experiments, but is enough to impact on the network performance (Chapter 4). Also, analyzing the performance under these conditions, we are able to evaluate the proposal under conditions that are challenging for the anomaly detection algorithm.

The network was declared under attack when the Security Manager detected an anomaly on the centralized monitoring or identified a node as attacker. The policy to identify the attacker using Algorithm 2 was set to 50%, which means that a node is classified as attacker when at least 50% of its neighbors reported it as suspect.

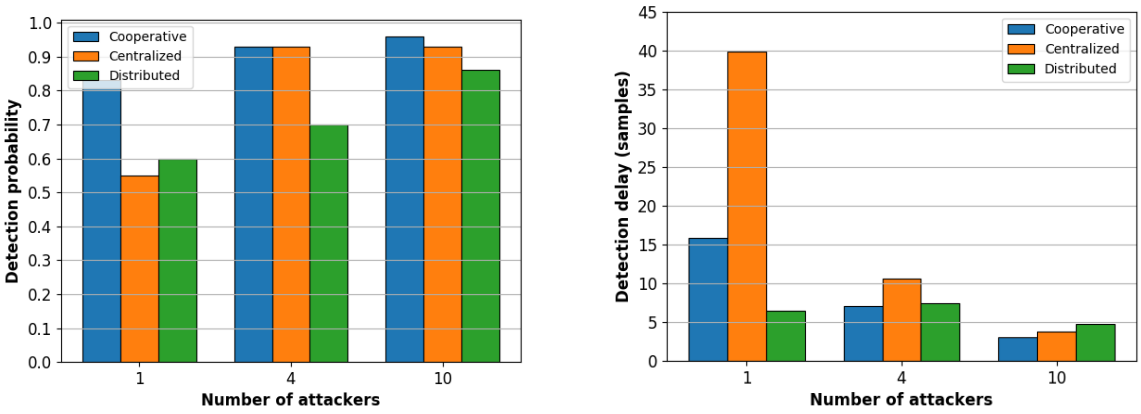
6.2.3.1 Results for FDFF attack

The DR and DTM results for the cooperative, centralized and distributed approaches for different number of attackers are shown in Figure 45. From these results, we observed that the main benefit of using the cooperative approach is when there are few attackers, obtaining an improvement of more than 0.20 in the detection probability. For four and ten attackers, the cooperative approach obtained equal or better detection probability than the centralized or distributed approach.

Different from the cases with 36 and 100 nodes, with detection probabilities exceeding 0.90, the centralized detection probability was around 0.55% for topologies with 225 nodes with only one attacker. The main reason for these results is that the attacker's neighborhood has the same size than the cases for 36 and 100 nodes but in a larger topology, thus the impact in global metrics is lower and slower. The distributed approach was a bit better (0.60) for this case and the reason is because it does not observe global metrics.

The FDFF attack impacts on the transmitting time of nodes beyond its neighborhood, which means that multiple attackers increase the anomaly detection probability in all nodes. One option to improve the detection rate when there is only one attacker could be by monitoring the control packets received, since this metric is less sensible to the number of attackers.

The DTM also presented interesting results. Increasing the number of attackers



(a) DR (b) DTM

Figure 45 – Comparison of the cooperative, centralized and distributed approaches for different number of attackers: attack every 60 seconds. (a) shows detection rate and (b) shows the detection time median

Source: Author

Table 7 – Centralized versus distributed approaches for one attack

Approach		**60 s**	**30 s**	**10 s**
Distributed	DR	1	0.8	0
	DTM	4.55	15	–
Centralized	DR	1	0.8	0.7
	DTM	4.55	15	28.14

Source: Author

reduces the detection time for the centralized and cooperative approaches, which was the expected since the impact of the attack is related to the number of attackers, however, this was not the case for the distributed approach. The results in Figure 45 concern the cases monitoring the transmitting time every 60, 30 and 10 seconds. The DTM is calculated using only the delay of true positive events, but for the case with one attacker monitoring the transmitting time every 10 seconds the number of true positives using the distributed approach was zero. The results for the specific case of one attacker are shown in Table 7. Here we observed that the cooperative approach is as fast as the distributed approach and is able to detect the attack when the distributed was not. However, increasing the monitoring frequency reduces the performance of the distributed approach. This could be caused by the periodicity of the data packets (application) and the attack launching period, both programmed in 60 seconds. The periodicity of the packet traffic caused that multiple samples had value zero.

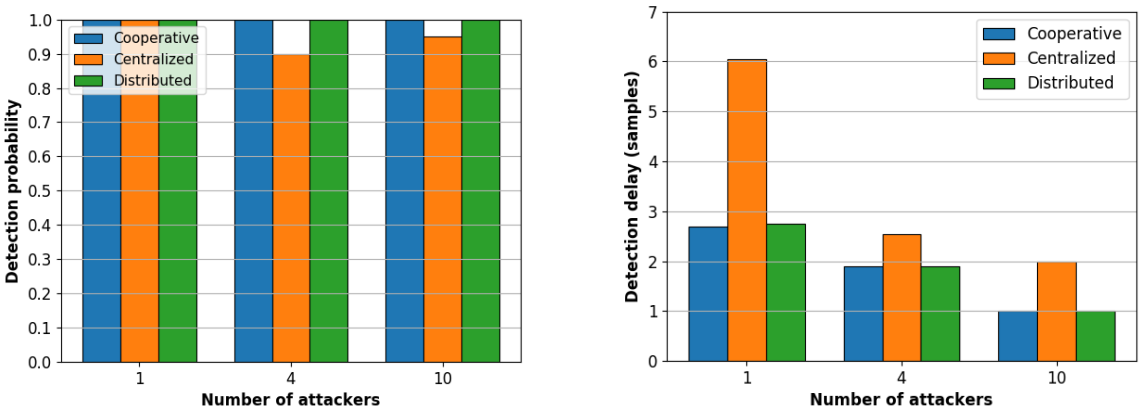
Motivated by these results, we simulated the case monitoring the control packets received every ten seconds. This metric is not related with normal data packets traffic

and depends on other processes without periodicity, such as neighbor information reports and routing rules configuration. The results for the case with one attacker showed a DR of 1.00 and all the attacker identifications occurred within the first minute, which is the fastest detection of all cases evaluated. Moreover, only the attacker's neighbors detected the anomaly, which aids to reduce false positive identification. However, the detection using transmitting time with a 60-second sampling period provides information about the impact of the attack in the whole network. One strategy could be to use both metrics with different sampling periods, but this require WSN devices with more memory resources. With our IT-SDN configuration, the TelosB mote does not have enough memory space to run both detectors.

From these first experiments, we noticed that the cooperative approach has the benefits of both centralized and distributed approach; it can identify the type of the attack and the attackers. Considering all the simulations, 64% of the attackers and 81% of the attacks were correctly identified. However, we observed before that the distributed approach had poor performance when using a monitoring period of 10 seconds. Considering only the simulations for 30 and 60 seconds, the percentage of attackers correctly detected increased to 94.33%. Likewise, the percentage of attacks correctly identified considering only the simulations for four and ten attackers increased to 98.33%.

Next, the case where the attack is launched every 10 seconds was analyzed. The DR and DTM metrics for one, four and ten attackers are depicted in Figure 46. From these results we noticed that all the approaches obtained a faster response for faster attacks and that the cooperative and distributed approach obtained equal results in terms of detection rate and detection speed. Thus, the benefit of the cooperative approach is the capacity of identifying the type of the attack thanks to the centralized monitoring. For these simulations, 98% of the attacks were correctly detected as FDFF and 98% of attackers were correctly detected.

It is worth mentioning that in this case the distributed approach monitoring every 10 seconds obtained a detection probability of 1, when there was only one attacker in the network, while for the case launching the attack every 60 seconds the detection rate was 0 (Table 7). This opens the discussion again about the sensibility of the transmitting time metric to the monitoring period and the importance of complementing monitoring other metrics, such as control packets received. Moreover, the detection performance for lower frequencies, also known as advanced persistant threats, could also be interesting to analyze since CP detection was designed to detect abrupt changes. Policies based on flow table changes analysis could be tested to detect this type of attack.



(a) DR (b) DTM

Figure 46 – Comparison of the cooperative, centralized and distributed approaches for different number of attackers: attack every 10 seconds. (a) shows detection rate and (b) shows the detection time median

Source: Author

6.2.3.2 Results for the FNI attack

The results in Subsection 6.2.2 showed the algorithms we proposed for attacker identification were not effective when the network was under FNI attack. Thus, based on the initial attack declaration hypothesis, in this case the detection relied only on the centralized monitoring. The results showed that 96.88% of attacks were correctly detected. The DR results classified by number of attackers are depicted in Figure 47. The only unexpected result is the $DR = 0.90$ for the case with 10 attackers launching the attack every 10 seconds. Analyzing the traces of the simulations, in multiple events the detector was triggered before the attack started, which reduced the DR metric. However, this does not mean the detector has lower detection rate for 10 attackers. This is related to the false positive rate.

About detection speed, the DTM when the attack was launched every 60 seconds was 5.5 samples, while when the attack was launched every 10 seconds it was 4.5 samples. The average delivery rate when the anomaly was detected is around 85%, for 1 and 4 nodes launching the attack every 60 seconds, and 67.67% for 10 attackers. In the case of the nodes launching the attack every 10 seconds, the delivery rate was between 77% and 73%. Aiming to improve the DTM and hence reduce the delivery rate drop caused by the attack, we explored the case of attack declaration based on the number of alarms received. The percentage of total nodes that reported an alarm when the network was under a FNI attack is depicted in Figure 48. From these results we noticed that: at least 10% of the total nodes detected an anomaly, the number of alarms is related to the number of attackers and, such as the case of FDFF attack, the distributed anomaly detection decrease

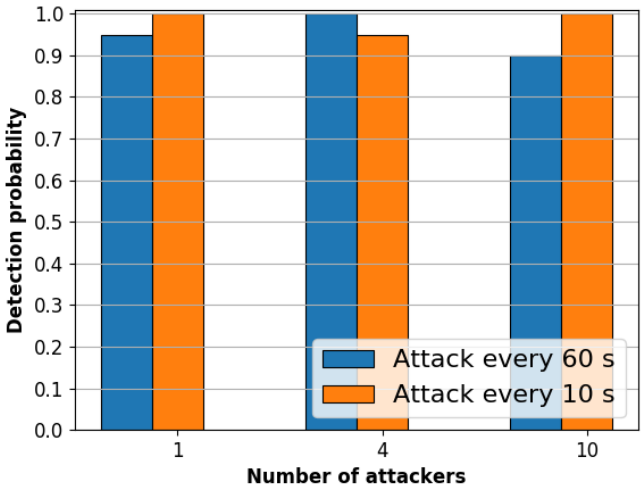
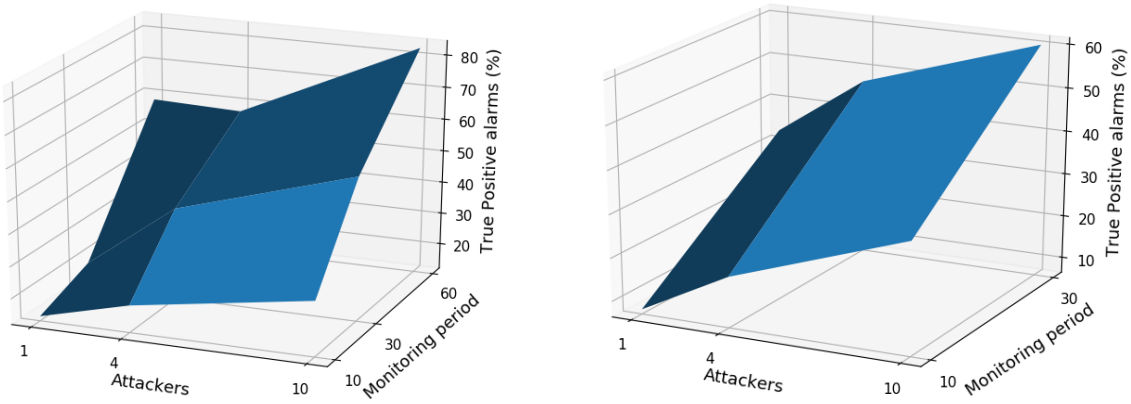


Figure 47 – DR results when the network was under FNI attack: classification by number of attackers

Source: Author

when reducing the monitoring period. The last point was caused by the periodicity of the packets traffic and its impact on the transmitting time, such as explained for the FDFF attack.



(a) Attack every 60 seconds

(b) Attack every 10 seconds

Figure 48 – Number of true positive alarms in function of the monitoring period and the number of attacks: (a) shows the results when FNI attack was launched every 60 seconds and (b) when the FNI attack was launched every 10 seconds

Source: Author

The next step was to determine the number of alarms received by the Security Manager before declaring the attack based on the centralized monitoring. Using this information, we were able to calculate the probability of detecting the attack before the centralized monitoring in function of the topology size percentage. The results are depicted

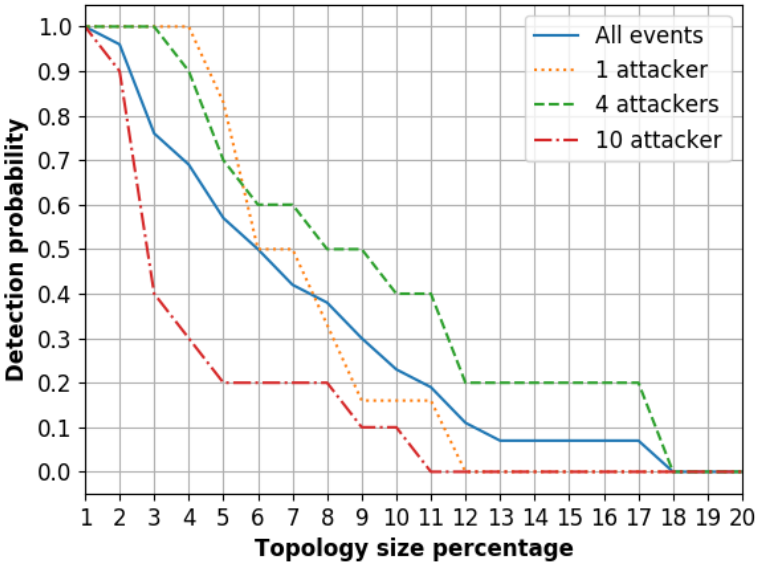


Figure 49 – Probability of FNI attack detection before the centralized monitoring based on the number of alarms received: 225-node topology case

Source: Author

in Figure 49.

The distributed FPR considering all the experiments was 0.007, which is equivalent to less than 2 nodes detecting an anomaly when the network is not under attack. However, when the nodes are monitoring the transmitting time every 30 seconds, the FPR increases to 0.021, which is equivalent to 2.1% of the topology size. Therefore, let us suppose the Security Manager is programmed to declare the network under attack when at least 3% of the total nodes report an alarm. In this case, the probability of detecting the attack before the centralized monitoring is 0.76. The probability increases to 1 for the cases with one and four attackers. Applying this policy, the average data packets delivery rate at the moment the Security Manager declares the network is under attack increases from 85% to 91.92%, for one and four attackers; and increases from 67.67% to 80.37% for 10 attackers.

6.2.3.3 Network Performance

Previous results showed the detection performance results of our proposal in comparison with a centralized and a distributed approaches. In this Subsection, the network performance results are analyzed. We compared the control and data packets delivery rate for the centralized and the cooperative approach varying the monitoring period. In the centralized approach, this defines the periodicity of the communication of every node with the Performance Manager, but for the cooperative approach this period defines the distributed sampling maintaining a centralized monitoring period in 60 seconds.

The delivery results for control and data packets are depicted in Figure 50. We

observed that the cooperative approach is able to maintain the data packets delivery rate stable by reducing the monitoring period on every node. In terms of scalability, this also means that using the cooperative approach, we would be able to reach or maintain delivery rate requirements on larger topologies since.

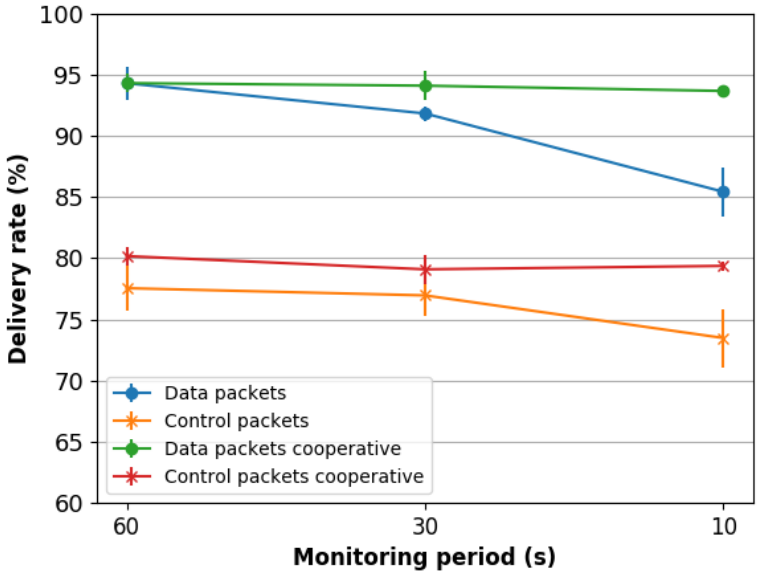


Figure 50 – Packets delivery rate increasing the monitoring periods of the metrics used to detect attacks: centralized vs cooperative approach

Source: Author

The end-to-end delay for control and data packets is depicted in Figure 51. In this case, the main benefit is for the data packets. Using the cooperative approach the delay remains around 400 ms regardless the monitoring period, while using the centralized the delay increases up to 100 ms. For the case of the control packets the results are not as clear as the data packets results. There is a high dispersion in the results obtained using the centralized approach that difficults the comparison. Based only on the average value, the cooperative approach is superior only when using a monitoring period of 60 seconds.

6.3 Summary

In this chapter, the detection performance of the cooperative intrusion detection proposal was evaluated. More than 800 simulations were conducted, varying the network size, the number of attackers and emulating real WSN devices. The experiments were divided in three groups. The first group (Subsection 6.2.1) was meant to evaluate the performance of the change point detection algorithm for centralized DoS attack detection. The second group (Subsection 6.2.2) comprises the experiments to evaluate the performance of the distributed DoS attack detection. Lastly, the third group evaluates the performance

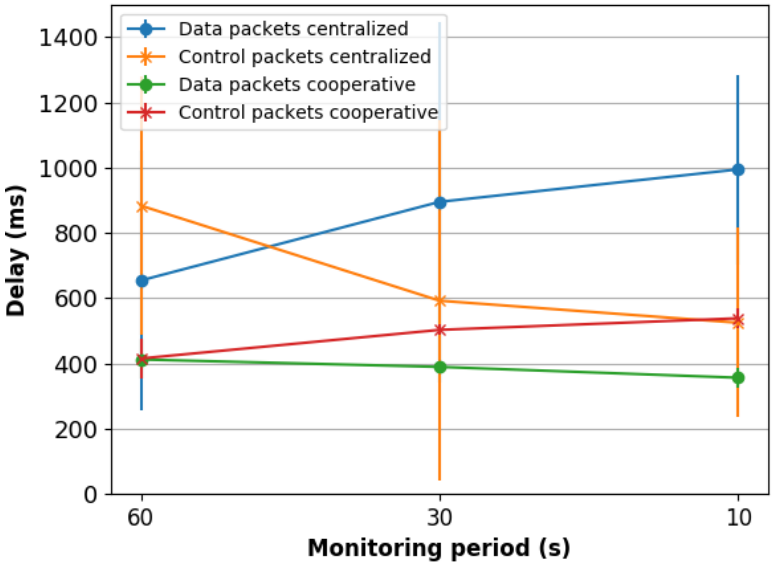


Figure 51 – Packets end-to-end delay for different monitoring periods of the metrics used to detect attacks: centralized vs cooperative approach

of the cooperative intrusion detection and compares the results with the cases using either the centralized or distributed detection.

Results showed that the cooperative approach was able to solve the scalability problem of centralized approaches, leveraged by the distributed anomaly detection. By reducing the centralized monitoring sampling rate, we were able to implement the DoS detection in topologies of 225 nodes without reducing the data packets delivery rate. To compensate the delay that could be caused by the centralized sampling rate reduction, all nodes are monitoring their own behavior aiming to detect anomalies related with a DoS attack. In this way, we concluded that the cooperative approach improved detection rate, specially for the case with few attackers, reduced the detection time, was accurate to classify attacks between FDFF and FNI attacks, and was able to identify FDFF attackers, fulfilling all the metrics analyzed in Chapter 3 for the state of the art, such as shown in Table 8.

Table 8 – Related work

Reference	High DR	Multiple types	Attack type ident.	Resour. cons.	Attacker ident.	Scal.	Complex.
Centralized Approaches							
(CHOI; KWAK, 2016)	✓						
(Özçelik; Chalabianloo; Gür, 2017)					✓		
(Bhunia; Gurusamy, 2017)	✓						
(ZARCA et al., 2018)				✓			
(BAGAA et al., 2020)	✓	✓		✓			
(Ravi; Shalinie, 2020)	✓				✓		
(FARHIN et al., 2020)		✓					
(WANI; S; KHALIQ, 2021)	✓	✓	✓				
Hybrid Approaches							
(WANG et al., 2018)		✓	✓	✓	✓		
(BIN-YAHYA; ALHUSSEIN; SHEN, 2021)	✓	✓	✓	✓	✓		
(Miranda et al., 2020)		✓		✓		✓	
(GRIGORYAN et al., 2018)					✓		
(LI et al., 2019)	✓	✓					
(Yin; Zhang; Yang, 2018)				✓	✓		
This thesis	✓	✓	✓	✓	✓	✓	✓

Source:Author

7 FINAL REMARKS

The Software-defined networking paradigm centralizes control decisions and enables the network to be intelligently and centrally programmed. These characteristics simplify network management and provide tools for infrastructure sharing. The aforementioned benefits motivated the discussion of implementing SDN in LLNs as a way to solve different limitations, but in particular concerning flexibility and resource reuse.

SDN has inspired novel security strategies but the control centralization and planes separation turn these networks prone to DoS attacks. This vulnerability is critical in SDN-based LLNs, since the resource constraints limit the implementation of security solutions for these attacks. This problem has been investigated and there are proposals in the literature that are effective detecting DoS attacks but at the cost of high resources usage.

From the literature revision, we identified that the main problem of current solutions is scalability. Centralized approaches impose high traffic overhead, requiring out-of-band communication in some cases, and large datasets for training. Distributed or hybrid approaches aid to reduce communication resources at the cost of lower detection performance or devices with high capabilities, hence a complexity challenge. From this, we asked the question: "What are the benefits of solving the complexity problem in hybrid approaches to increase scalability of intrusion detection in SDN-based resource-constrained networks?"

First, a lightweight anomaly detection that could run on resource-constrained devices (complexity challenge) was proposed and the metrics that maximized accuracy were studied. Then, the communication resources required by a centralized application were adjusted according to the network traffic characteristics. By running the detection algorithm on every node, we relieve the centralized monitoring from all the responsibility of anomaly detection. Lastly, a Security Manager analyzes centralized and distributed information to detect DoS attacks and identify attackers, supported by the information provided by the SDN controller and a Performance Manager.

Our results showed that, adjusting the centralized monitoring, we were able to scale the solution to large networks with low impact on the network performance. Combining both centralized and distributed approaches we improved detection accuracy and reduced detection delay. Moreover, the cooperation among the SDN controller and the Security Manager was instrumental to identify the attackers when the network was under FDFF attack. It is worth mentioning that, rather than concentrate the distributed detection on few nodes with high capabilities, we decided that all nodes participate in the detection at the cost of a low increase in the processing overhead. Therefore, this solution can be

implemented in homogeneous networks with resource constraints.

The results from this research are an important step to develop new intrusion detection strategies that scale according to new technology trends. Industry 4.0, Smart Cities and 5G are examples of technologies or paradigms that integrate SDN, WSNs and IoT in large scale. Cooperative intrusion detection is one option to turn these networks more secure without a significant impact on their performance.

7.1 Challenges and limitations

During this research, we had to face multiple challenges. Most of the works about intrusion detection for SDWSN and SD-IoT are well explained about the mathematics and algorithms implementation but lack experimentation information that limits their reproducibility. Moreover, looking for a way to access source codes, datasets and results was also a challenge.

Covid-19 pandemic was also a problem that limited this research. Network simulations with 225 nodes require several time. One set of 10 replications could take between 4 and 10 days, depending on the computer used. Despite most of the work being conducted remotely, electrical energy issues in the laboratory limited the number of computers that we could use for simulation, which limited the number of metrics that we could analyze.

7.2 Publications

Five research papers were published from the results presented in this thesis, enumerated below in chronological order:

1. "Understanding the Performance of Software Defined Wireless Sensor Networks under Denial of Service Attack", published at the Open Journal of Internet of Things and presented at the International Workshop on Very Large Internet of Things 2019 (SEGURA; MARGI; CHORTI, 2019). This paper contains the results about network performance impact when the network is under attack, present in this document in Chapter 4.

2. "Denial of Service Attacks Detection in Software-Defined Wireless Sensor Networks", presented at the SecSDN: Secure & Dependable Software Defined Networking for Sustainable Smart Communities, in Conjunction with IEEE ICC 2020 (SEGURA et al., 2020). This paper contains the results of applying (SKAPERAS; MAMATAS; CHORTI, 2018) algorithm to detect DoS attacks in SDWSNs.

3. "Multimetric Online Intrusion Detection in Software-Defined Wireless Sensor Networks", presented at the IEEE Latin-American Conference on Communications 2020 (SEGURA; MARGI; CHORTI, 2020). This paper contains the results of applying the

modified version of (SKAPERAS; MAMATAS; CHORTI, 2018) algorithm to detect DoS attacks in SDWSNs. These results are present in this document in Subsection 6.2.1.

4. "Distributed DoS Attack Detection in SDN: Trade offs in Resource Constrained Wireless Networks", presented at the IEEE Statistical Signal Processing Workshop 2021 (SEGURA; MARGI; CHORTI, 2021). This paper contains our first results about distributed detection using change point analysis in every node of the network. Part of these results are present in Subsection 6.2.2.

5. "Centralized and Distributed Intrusion Detection for Resource-Constrained Wireless SDN Networks", published at the IEEE Internet of Things Journal (SEGURA; CHORTI; MARGI, 2021). This paper extend the results about centralized and distributed DoS detection and explains the cooperation between both approaches to improve detection performance.

Additionally, three other papers about performance and management of SDWSNs, were published during the PhD in collaboration with research group colleagues:

- "Software-defined wireless sensor networks approach: Southbound protocol and its performance evaluation", published at the Open Journal of Internet Of Things 2018 (MARGI et al., 2018)

- "In-network performance measurements for Software Defined Wireless Sensor Networks", presented at the IEEE International Conference on Networking, Sensing and Control 2019 (Luz et al., 2019).

- "The cost of software-defining things: A scalability study of software-defined sensor networks", published at the IEEE Access Journal (Alves et al., 2019).

7.3 Future work

Some ideas to improve and continue these research are the following:

- implementation of mitigation strategies based on the attackers and type of attack identification, completing the cooperative solution;

- investigate solutions for FNI attackers identification, since our algorithms were effective only when the network was under FDFF attack;

- implement multimetric anomaly detection in WSN nodes to study the relation between the metrics and the type of the attack;

- implement and evaluate the classification algorithms for complex anomalies explained in Chapter 5.

This work could also be extended to other type of attacks that are not exclusive of SDN, such as wormhole attack or selective forwarding. In this work we focus on the control plane but attacks that attain the data plane could also be investigated. At last, during the whole work we consider the controller as a secure entity and all the attacks went through the WSN devices. One next step could be to analyze the case where the controller has been attacked through the NSAL.

REFERENCES

AHMAD, I. et al. Security in Software Defined Networks: A Survey. **IEEE Commun. Surveys Tuts.**, v. 17, n. 4, p. 2317–2346, Fourthquarter 2015. ISSN 1553-877X.

ALEXANDER, R. et al. **RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks**. RFC Editor, 2012. RFC 6550. (Request for Comments, 6550). Disponível em: https://rfc-editor.org/rfc/rfc6550.txt.

ALVES, R. C. et al. Ws3n: wireless secure sdn-based communication for sensor networks. **Security and Communication Networks**, Hindawi, v. 2018, 2018.

ALVES, R. C. A.; MARGI, C. B.; KUIPERS, F. A. No way back? An SDN protocol for directed IoT networks. *In*: **Wireless On-demand Network systems and Services Conference, 15. Proceedings[...]**. [*S.l.: s.n.*]: IEEE, 2019.

ALVES, R. C. A.; MARGI, C. B.; KUIPERS, F. A. Know when to listen: Sdn-based protocols for directed iot networks. **Computer Communications**, v. 150, p. 672–686, 2020. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366419303275.

ALVES, R. C. A. et al. IT-SDN: Improved architecture for SDWSN. **BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS AND DISTRIBUTED SYSTEMS, 35. Proceedings [...]**, 2017.

Alves, R. C. A. et al. The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks. **IEEE Access**, v. 7, p. 115093–115108, Aug 2019. ISSN 2169-3536.

AUE, A.; HORVATH, L. Structural breaks in time series. **Journal of Time Series Analysis**, v. 34, n. 1, p. 1–16, 2013. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.2012.00819.x.

BADDELEY, M. et al. Evolving SDN for Low-Power IoT Networks. *In*: **IEEE CONFERENCE ON NETWORK SOFTWARIZATION AND WORKSHOPS (NetSoft), 4. Proceedings [...]**. [*S.l.: s.n.*], 2018. p. 71–79.

BAGAA, M. et al. A Machine Learning Security Framework for Iot Systems. **IEEE Access**, v. 8, p. 114066–114077, 2020.

BANOS-GONZALEZ, V. et al. Ieee 802.11ah: A technology to face the iot challenge. **Sensors**, v. 16, n. 11, 2016. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/16/11/1960.

BASSEVILLE, M.; NIKIFOROV, I. V. et al. **Detection of abrupt changes: theory and application**. [*S.l.: s.n.*]: Prentice Hall Englewood Cliffs, 1993. v. 104.

BENZEKKI, K.; FERGOUGUI, A. E.; ELALAOUI, A. E. Software-defined networking (sdn): a survey. **Security and Communication Networks**, v. 9, n. 18, p. 5803–5833, 2016.

Bhunia, S. S.; Gurusamy, M. Dynamic attack detection and mitigation in IoT using SDN. *In*: **INTERNATIONAL TELECOMMUNICATION NETWORKS AND APPLICATIONS CONFERENCE (ITNAC), 27. Proceedings [...]**. [*S.l.: s.n.*], 2017. p. 1–6.

BIN-YAHYA, M.; ALHUSSEIN, O.; SHEN, X. Securing software-defined wsns communication via trust management. **IEEE Internet of Things Journal**, p. 1–1, 2021.

BRAGA, R.; MOTA, E.; PASSITO, A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. *In*: **IEEE Local Computer Network Conference**. Denver, CO, USA: IEEE, 2010. p. 408–415.

Butun, I.; Morgera, S. D.; Sankar, R. A survey of intrusion detection systems in wireless sensor networks. **IEEE Commun. Surveys Tuts.**, v. 16, n. 1, p. 266–282, First 2014. ISSN 1553-877X.

CABAJ, K. et al. Sdn architecture impact on network security. *In*: GANZHA L. MACIASZEK, M. P. M. (ed.). **Position Papers of the 2014 Federated Conference on Computer Science and Information Systems**. PTI, 2014. (Annals of Computer Science and Information Systems, v. 3), p. pages 143–148. Disponível em: http://dx.doi.org/10.15439/2014F473.

Changlong Chen; Song, M.; Hsieh, G. Intrusion detection of sinkhole attacks in large-scale wireless sensor networks. *In*: **IEEE INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND INFORMATION SECURITY, Proceedings [...]**. [*S.l.: s.n.*], 2010. p. 711–716.

CHOI, S.; KWAK, J. Enhanced SDIoT Security Framework Models. **International Journal of Distributed Sensor Networks**, v. 12, n. 5, p. 4807804, 2016.

CONTI, M.; GANGWAL, A.; GAUR, M. A comprehensive and effective mechanism for ddos detection in sdn. *In*: **IEEE INTERNATIONAL CONFERENCE ON WIRELESS AND MOBILE COMPUTING, NETWORKING AND COMMUNICATIONS (WIMOB), 13. Proceedings [...]**. [*S.l.: s.n.*], 2017. p. 1–8.

COSTANZO, S. et al. Software Defined Wireless Networks: Unbridling SDNs. *In*: **European Workshop on Software Defined Networking. Proceedings [...]**. [*S.l.: s.n.*], 2012. p. 1–6.

CROSSBOW. **TelosB mote platform**. [*S.l.*], 2004. Revision B.

DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. *In*: IEEE. **Local Computer Networks, 2004. 29th Annual IEEE International Conference on**. [*S.l.: s.n.*], 2004. p. 455–462.

ENNS, R. et al. **Network configuration protocol (NETCONF)**. [*S.l.*], 2011.

FARHIN, F. et al. Attack detection in internet of things using software defined network and fuzzy neural network. *In*: **2020 Joint 9th International Conference on Informatics, Electronics Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision Pattern Recognition (icIVPR)**. [*S.l.: s.n.*], 2020. p. 1–6.

FOULADI, R. F.; ERMIS, O.; ANARIM, E. A ddos attack detection and defense scheme using time-series analysis for sdn. **Journal of Information Security and Applications**, v. 54, p. 102587, 2020. ISSN 2214-2126. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214212620307560.

FREMDT, S. Asymptotic distribution of the delay time in page's sequential procedure. **Journal of Statistical Planning and Inference**, v. 145, p. 74 – 91, 2014. ISSN 0378-3758. Disponível em: http://www.sciencedirect.com/science/article/pii/S0378375813002139.

GALLUCCIO, L. et al. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. *In*: **2015 IEEE Conference on Computer Communications (INFOCOM)**. [*S.l.: s.n.*], 2015. p. 513–521.

GENI. 2018. Disponível em: https://www.geni.net/.

GIOTIS, K.; ANDROULIDAKIS, G.; MAGLARIS, V. Leveraging sdn for efficient anomaly detection and mitigation on legacy networks. *In*: **2014 Third European Workshop on Software Defined Networks**. [*S.l.: s.n.*], 2014. p. 85–90.

GRIGORYAN, G. et al. Enabling cooperative iot security via software defined networks (sdn). *In*: IEEE. **2018 IEEE International Conference on Communications (ICC)**. [*S.l.: s.n.*], 2018. p. 1–6.

GUSTAFSSON, F. The marginalized likelihood ratio test for detecting abrupt changes. **IEEE Transactions on Automatic Control**, v. 41, n. 1, p. 66–78, Jan 1996. ISSN 0018-9286.

GUTIERREZ, J. A.; CALLAWAY, E. H.; BARRETT, R. **IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks**. USA: IEEE Standards Office, 2003. ISBN 0738135577.

HALEPLIDIS, E. et al. **Software-defined networking (SDN): Layers and architecture terminology**. [*S.l.*], 2015.

HASHEMI, S. Y.; ALIEE, F. S. Dynamic and comprehensive trust model for iot and its integration into rpl. **The Journal of Supercomputing**, Springer, v. 75, n. 7, p. 3555–3584, 2019.

HIRAYAMA, T.; KENTAROH, T.; SASASE, I. Fast target link flooding attack detection scheme by analyzing traceroute packets flow. *In*: **2015 IEEE International Workshop on Information Forensics and Security (WIFS)**. [*S.l.: s.n.*], 2015. p. 1–6.

Jia, Y. et al. Flowguard: An intelligent edge defense mechanism against iot ddos attacks. **IEEE Internet of Things Journal**, v. 7, n. 10, p. 9552–9562, 2020.

KALKAN, K.; GUR, G.; ALAGOZ, F. Sdnscore: A statistical defense mechanism against ddos attacks in sdn environment. *In*: **2017 IEEE Symposium on Computers and Communications (ISCC)**. [*S.l.: s.n.*], 2017. p. 669–675.

KHARRUFA, H.; AL-KASHOASH, H. A. A.; KEMP, A. H. Rpl-based routing protocols in iot applications: A review. **IEEE Sensors Journal**, v. 19, n. 15, p. 5952–5967, 2019.

Kobo, H. I.; Abu-Mahfouz, A. M.; Hancke, G. P. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. **IEEE Access**, v. 5, p. 1872–1899, 2017.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, v. 78, n. 9, p. 1464–1480, 1990.

Kreutz, D. et al. Software-Defined Networking: A Comprehensive Survey. **Proc. IEEE Proc.**, v. 103, n. 1, p. 14–76, Jan 2015.

LATIF, Z. et al. A comprehensive survey of interface protocols for software defined networks. **Journal of Network and Computer Applications**, v. 156, p. 102563, 2020. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804520300370.

LEE, T.-H. et al. A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan. *In*: HUANG, Y.-M. et al. (ed.). **Advanced Technologies, Embedded and Multimedia for Human-centric Computing**. Dordrecht: Springer Netherlands, 2014. p. 1205–1213. ISBN 978-94-007-7262-5.

LEVIS, P. et al. Tinyos: An operating system for sensor networks. *In*: _____. **Ambient Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 115–148. ISBN 978-3-540-27139-0. Disponível em: https://doi.org/10.1007/3-540-27139-2_7.

LI, J. et al. Ai-based two-stage intrusion detection for software defined iot networks. **IEEE Internet of Things Journal**, v. 6, n. 2, p. 2093–2102, 2019.

Liu, K. et al. Self-diagnosis for large scale wireless sensor networks. *In*: **2011 Proceedings IEEE INFOCOM**. [*S.l.: s.n.*], 2011. p. 1539–1547. ISSN 0743-166X.

LUO, T.; TAN, H.-P.; QUEK, T. Q. S. Sensor openflow: Enabling software-defined wireless sensor networks. **IEEE Communications Letters**, v. 16, n. 11, p. 1896–1899, 2012.

Luz, T. et al. In-network performance measurements for Software Defined Wireless Sensor Networks. *In*: **16th IEEE Int. Conf. Netw., Sens. and Control (ICNSC 2019)**. [*S.l.: s.n.*], 2019.

LUZ, T. C.; MARGI, C. B.; VERDI, F. L. Network metrics detection to support internet of things application orchestration. **Open Journal of Internet Of Things (OJIOT)**, RonPub, v. 7, n. 1, p. 93–103, 2021. ISSN 2364-7108. Disponível em: http://nbn-resolving.de/urn:nbn:de:101:1-202108291933420773657 5.

MAHMUD, A.; RAHMANI, R. Exploitation of openflow in wireless sensor networks. *In*: **Proceedings of 2011 International Conference on Computer Science and Network Technology**. [*S.l.: s.n.*], 2011. v. 1, p. 594–600.

MAHRACH, S. et al. Sdn-based syn flooding defense in cloud. **Journal of Information Assurance and Security**, v. 13, n. 1, 2013.

MARDINI, W.; EBRAHIM, M.; AL-RUDAINI, M. Comprehensive performance analysis of rpl objective functions in iot networks. **International Journal of Communication Networks and Information Security**, Kohat University of Science and Technology (KUST), v. 9, n. 3, p. 323–332, 2017.

MARGI, C. B. et al. Software-Defined Wireless Sensor Networks Approach: Southbound Protocol and Its Performance Evaluation. **Open Journal of Internet Of Things (OJIOT)**, RonPub, v. 4, n. 1, p. 99–108, 2018. ISSN 2364-7108. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil. Disponível em: http://nbn-resolving.de/urn:nbn:de:101:1-201808051930571018960 7.

MCKEOWN, N. et al. OpenFlow: Enabling Innovation in Campus Networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

Miranda, C. et al. A collaborative security framework for software-defined wireless sensor networks. **IEEE Transactions on Information Forensics and Security**, p. 1–1, 2020. ISSN 1556-6021.

MORALES, L. V.; MURILLO, A. F.; RUEDA, S. J. Extending the floodlight controller. *In*: **2015 IEEE 14th International Symposium on Network Computing and Applications**. [*S.l.: s.n.*], 2015. p. 126–133.

MORIN, E. et al. Comparison of the device lifetime in wireless networks for the internet of things. **IEEE Access**, v. 5, p. 7097–7114, 2017.

Mudzingwa, D.; Agrawal, R. A study of methodologies used in intrusion detection and prevention systems (idps). *In*: **2012 Proceedings of IEEE Southeastcon**. [*S.l.: s.n.*], 2012. p. 1–6. ISSN 1558-058X.

NAOUS, J. et al. Delegating network security with more information. *In*: **Proceedings of the 1st ACM Workshop on Research on Enterprise Networking**. New York, NY, USA: Association for Computing Machinery, 2009. (WREN '09), p. 19–26. ISBN 9781605584430.

NAYAK, A. K. et al. Resonance: Dynamic access control for enterprise networks. *In*: **Proceedings of the 1st ACM Workshop on Research on Enterprise Networking**. New York, NY, USA: Association for Computing Machinery, 2009. (WREN '09), p. 11–18. ISBN 9781605584430.

NOORIBAKHSH, M.; MOLLAMOTALEBI, M. A review on statistical approaches for anomaly detection in ddos attacks. **Information Security Journal: A Global Perspective**, Taylor & Francis, v. 29, n. 3, p. 118–133, 2020.

OLIVEIRA, B. T. de; MARGI, C. B.; GABRIEL, L. B. Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks. *In*: **2014 IEEE Latin-America Conference on Communications (LATINCOM)**. [*S.l.: s.n.*], 2014. p. 1–6.

OLIVEIRA, D. A. G.; MARGI, C. B. Combining metrics for route selection in sdwsn: Static and dynamic approaches evaluation. *In*: **2018 IEEE 10th Latin-American Conference on Communications (LATINCOM)**. [*S.l.: s.n.*], 2018. p. 1–6.

Osterlind, F. et al. Cross-Level Sensor Network Simulation with COOJA. *In*: **Proc. IEEE Conf. Local Comput. Netw. (LCN)**. [*S.l.: s.n.*], 2006. p. 641–648.

PONGLE, P.; CHAVAN, G. Real time intrusion and wormhole attack detection in internet of things. **International Journal of Computer Applications**, Foundation of Computer Science, v. 121, n. 9, 2015.

POOR, H. V.; HADJILIADIS, O. **Quickest detection**. [*S.l.: s.n.*]: Cambridge University Press, 2008.

Pritchard, S. W.; Hancke, G. P.; Abu-Mahfouz, A. M. Security in software-defined wireless sensor networks: Threats, challenges and potential solutions. *In*: **2017 IEEE 15th International Conference on Industrial Informatics (INDIN)**. Emden, Germany: IEEE, 2017. p. 168–173.

QAZI, Z. A. et al. Application-awareness in sdn. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 43, n. 4, p. 487–488, ago. 2013. ISSN 0146-4833. Disponível em: https://doi.org/10.1145/2534169.2491700.

QIN, S. J. An overview of subspace identification. **Computers & Chemical Engineering**, v. 30, n. 10, p. 1502 – 1513, 2006. ISSN 0098-1354. Papers form Chemical Process Control VII. Disponível em: http://www.sciencedirect.com/science/article/pii/S009813540600158X.

Ravi, N.; Shalinie, S. M. Learning-driven detection and mitigation of ddos attack in iot via sdn-cloud architecture. **IEEE Internet of Things Journal**, v. 7, n. 4, p. 3559–3570, 2020.

Rawat, D. B.; Reddy, S. R. Software defined networking architecture, security and energy efficiency: A survey. **IEEE Commun. Surveys Tuts.**, v. 19, n. 1, p. 325–346, Firstquarter 2017. ISSN 1553-877X.

SANGODOYIN, A. et al. An approach to detecting distributed denial of service attacks in software defined networks. *In*: **2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)**. [*S.l.: s.n.*], 2018. p. 436–443.

SAYJARI, T.; SILVEIRA, R. M.; MARGI, C. B. Control and data traffic isolation in sdwsn using ieee 802.15.4e tsch. *In*: **2021 IEEE Statistical Signal Processing Workshop (SSP)**. [*S.l.: s.n.*], 2021. p. 126–130.

SCARFONE, K.; MELL, P. et al. Guide to intrusion detection and prevention systems (idps). **NIST special publication**, Washington, D.C., 2007.

SCOTT-HAYWARD, S.; O'CALLAGHAN, G.; SEZER, S. Sdn Security: A Survey. *In*: **2013 IEEE SDN for Future Networks and Services (SDN4FNS)**. Trento, Italy: IEEE, 2013. p. 1–7.

SEGURA, G. A. N.; CHORTI, A.; MARGI, C. B. Centralized and Distributed Intrusion Detection for Resource-Constrained Wireless SDN Networks. **IEEE Internet of Things Journal**, p. 1–1, 2021.

SEGURA, G. A. N.; MARGI, C. B.; CHORTI, A. Understanding the Performance of Software Defined Wireless Sensor Networks Under Denial of Service Attack. **Open Journal of Internet Of Things (OJIOT)**, RonPub, 2019. Special Issue: Proc. Int. Workshop Very Large Internet of Things (VLIoT 2019) in conjunction with the VLDB 2019 Conf. Los Angeles, United States.

SEGURA, G. A. N.; MARGI, C. B.; CHORTI, A. Multimetric online intrusion detection in software-defined wireless sensor networks. *In*: **2020 IEEE Latin-American Conference on Communications (LATINCOM)**. Virtual Conference: IEEE, 2020. p. 1–6.

SEGURA, G. A. N.; MARGI, C. B.; CHORTI, A. Distributed dos attack detection in sdn: Tradeoffs in resource constrained wirelessnetworks. *In*: **2021 IEEE Statistical Signal Processing Processing Workshop (SSP)**. Rio de Janeiro, Brazil: IEEE, 2021. p. 1–6.

SEGURA, G. N. et al. Denial of Service Attacks Detection in Software-Defined Wireless Sensor Networks. *In*: **SecSDN IEEE Int. Conf. Commun. (ICC)**. Dublin, Ireland: IEEE, 2020.

SHAKHOV, V. V. Protecting wireless sensor networks from energy exhausting attacks. *In*: MURGANTE, B. et al. (ed.). **Computational Science and Its Applications – ICCSA 2013**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 184–193. ISBN 978-3-642-39637-3.

SHITAO, F. An Introduction to Krylov Subspace Methods. **arXiv e-prints**, p. arXiv:1811.09025, Nov 2018.

SILVA, A. Santos da et al. Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn. *In*: **NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium**. [*S.l.: s.n.*], 2016. p. 27–35.

SINGH, M. P.; BHANDARI, A. New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges. **Computer Communications**, v. 154, p. 509–527, 2020. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366419313830.

SKAPERAS, S.; MAMATAS, L.; CHORTI, A. Early Video Content Popularity Detection with Change Point Analysis. *In*: **IEEE Global Commun. Conf. (GLOBECOM)**. Abhu-Dhabi, United Arab Emirates: [*S.l.: s.n.*], 2018.

Skaperas, S.; Mamatas, L.; Chorti, A. Real-Time Video Content Popularity Detection Based on Mean Change Point Analysis. **IEEE Access**, v. 7, p. 142246–142260, 2019.

SORNIN, N. et al. **Lorawan specification**. 2015.

SULTANA, N. et al. Survey on sdn based network intrusion detection system using machine learning approaches. **Peer-to-Peer Networking and Applications**, Springer, v. 12, n. 2, p. 493–501, 2019.

TAKEUCHI, J.-i.; YAMANISHI, K. A unifying framework for detecting outliers and change points from time series. **IEEE Transactions on Knowledge and Data Engineering**, v. 18, n. 4, p. 482–492, April 2006. ISSN 1041-4347.

Thanigaivelan, N. K. et al. Distributed Internal Anomaly Detection System for Internet-of-Things. *In*: **2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [*S.l.: s.n.*], 2016. p. 319–320. ISSN 2331-9860.

THEODOROU, T.; MAMATAS, L. Coral-sdn: A software-defined networking solution for the internet of things. *In*: **2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [*S.l.: s.n.*], 2017. p. 1–2.

THEODOROU, T.; MAMATAS, L. A versatile out-of-band software-defined networking solution for the internet of things. **IEEE Access**, v. 8, p. 103710–103733, 2020.

THEODOROU, T. et al. A multi-protocol software-defined networking solution for the internet of things. **IEEE Communications Magazine**, v. 57, n. 10, p. 42–48, 2019.

TOLEDO, C. M. de et al. Enabling security in software-defined wireless sensor networks for internet of things. *In*: SBC. **Anais da XVIII Escola Regional de Redes de Computadores**. [*S.l.: s.n.*], 2020. p. 109–115.

WANG, R. et al. ETMRM: An Energy-efficient Trust Management and Routing Mechanism for SDWSNs. **Computer Networks**, v. 139, p. 119 – 135, 2018. ISSN 1389-1286.

WANI, A.; S, R.; KHALIQ, R. SDN-based intrusion detection system for IoT using deep learning classifier (IDSIoT-SDL). **CAAI Trans. on Intelligence Technol.**, v. 6, n. 3, p. 281–290, 2021.

WATTEYNE, T. et al. From manet to ietf roll standardization: A paradigm shift in wsn routing protocols. **IEEE Communications Surveys Tutorials**, v. 13, n. 4, p. 688–707, 2011.

WOOLDRIDGE, M. **An introduction to multiagent systems**. [*S.l.: s.n.*]: John wiley & sons, 2009.

WOOLLEY, M. **Bluetooth core specification v5**. 2019.

XIE, J. et al. A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. **IEEE Communications Surveys Tutorials**, v. 21, n. 1, p. 393–430, 2019.

XU, Y.; LIU, Y. DDoS attack detection under SDN context. *In*: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [*S.l.: s.n.*], 2016. p. 1–9.

YAO, X.; LIU, Y. Machine learning. *In*: _____. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. Boston, MA: Springer US, 2014. p. 477–517. ISBN 978-1-4614-6940-7. Disponível em: https://doi.org/10.1007/978-1-4614-6940-7_17.

Yin, D.; Zhang, L.; Yang, K. A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework. **IEEE Access**, v. 6, p. 24694–24705, 2018.

ZARCA, A. M. et al. Enhancing iot security through network softwarization and virtual security appliances. **International Journal of Network Management**, v. 28, n. 5, p. e2038, 2018. E2038 nem.2038. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2038.

ZARPELÃO, B. B. et al. A survey of intrusion detection in internet of things. **Journal of Network and Computer Applications**, v. 84, p. 25 – 37, 2017. ISSN 1084-8045. Disponível em: http://www.sciencedirect.com/science/article/pii/S1084804517300802.

Özçelik, M.; Chalabianloo, N.; Gür, G. Software-defined edge defense against iot-based ddos. *In*: **2017 IEEE International Conference on Computer and Information Technology (CIT)**. [*S.l.: s.n.*], 2017. p. 308–313.