UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO GERMANO DA SILVA

# A One-Class NIDS for SDN-based SCADA Systems

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Alberto Egon
Schaeffer-Filho

Porto Alegre
January 2017

**CIP – CATALOGING-IN-PUBLICATION**

Silva, Eduardo Germano da

A One-Class NIDS for SDN-based SCADA Systems / Eduardo Germano da Silva. – Porto Alegre: PPGC da UFRGS, 2017.

100 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Alberto Egon Schaeffer-Filho.

1. Supervisory control and data acquisition. 2. Software-defined networking. 3. Smart grids. 4. Network-based intrusion detection system. 5. One-class classification. I. Schaeffer-Filho, Alberto Egon. II. Título.

## ACKNOWLEDGMENTS

Well... There were so many good moments, so many important people, so much learning to thank that it is hard to describe everything in only one page. I will start where everyone usually starts, thanking the parents. I'd like to thank them, for the foundations, care giving, and opportunities that they provided me.

During my time at UFRGS, I was sure that I left the teenage years and entered in the adult life. I'm grateful to Alberto, my advisor, and every professor of the Computer Networks Group... Really, they are awesome. Alberto and Lisandro, I'm grateful for every opportunity that you offered me... I didn't give up of the PhD course, I just need some time to help who need me today.

I want to thank (really, I'm very grateful) to my lab friends (they were much more than simple colleagues), many times I wanted to give up and they didn't let me. They were my family here in Porto Alegre. They were there to listen me, to help me, and to make me continue in this master's degree. I won't mention names, they know how important they were for me. I thank to the evaluation committee's professors, CNPq, ProSeG, INF, the Computer Networks Group, UFRGS, and for everything good that will happen in my life henceforth.

Finally, I would like to emphasize that... This work was supported by ProSeG - Information Security, Protection and Resilience in Smart Grids, a research project funded by MCTI/CNPq/CT-ENERG # 33/2013.

Without further ado ... Thanks for everything!

## AGRADECIMENTOS

Poxa... Foram tantos momentos bons, tantas pessoas importantes, tantos ensinamentos pra agradecer que fica até difícil descrever tudo em uma página. Irei começar por onde geralmente todos começam, agradecendo aos pais. Gostaria de agradecer aos meus pais, pela criação e carinho que me deram, além das oportunidades que me proporcionaram.

Em meu período na UFRGS, tive a certeza de que saí da adolescência e ingressei na "vida adulta". Devo isso ao Alberto, meu orientador, e a cada professor do Grupo de Redes... Sério eles são demais. Alberto e Lisandro, obrigado por todas as oportunidades que vocês me ofereceram... Eu não desisti do doutorado, só preciso de um tempo pra poder ajudar quem mais precisa de mim hoje.

Agradeço (e agradeço muito mesmo) aos meus amigos de laboratório (eles foram muito mais do que colegas), muitas vezes pensei em desistir e eles não deixaram. Eles foram a minha família aqui em Porto Alegre. Eles estavam lá pra me escutar, me ajudar, e me fazer continuar nesse mestrado. Não citarei nomes, eles sabem o quanto foram importantes pra mim. Agradeço aos professores da banca, ao CNPq, ao ProSeG, ao INF, ao Grupo de Redes, à UFRGS, e por tudo de bom que irá acontecer na minha vida a partir de agora.

Pra deixar tudo bunitinho, nos conformes, gostaria de salientar que... Este trabalho foi apoiado pelo ProSeG - *Information Security, Protection and Resilience in Smart Grids*, um projeto de pesquisa financiado pelo MCTI/CNPq/CT-ENERG # 33/2013.

Sem mais delongas... Obrigado por tudo!

*"O Filho da empregada também vai virar Doutor".*

— EDUARDO GERMANO

## ABSTRACT

Power grids have great influence on the development of the world economy. Given the importance of the electrical energy to our society, power grids are often target of network intrusion motivated by several causes. To minimize or even to mitigate the aftereffects of network intrusions, more secure protocols and standardization norms to enhance the security of power grids have been proposed. In addition, power grids are undergoing an intense process of modernization, and becoming highly dependent on networked systems used to monitor and manage power components. These so-called Smart Grids comprise energy generation, transmission, and distribution subsystems, which are monitored and managed by Supervisory Control and Data Acquisition (SCADA) systems. In this Masters dissertation, we investigate and discuss the applicability and benefits of using Software-Defined Networking (SDN) to assist in the deployment of next generation SCADA systems. We also propose an Intrusion Detection System (IDS) that relies on specific techniques of traffic classification and takes advantage of the characteristics of SCADA networks and of the adoption of SDN/OpenFlow. Our proposal relies on SDN to periodically gather statistics from network devices, which are then processed by One-Class Classification (OCC) algorithms. Given that attack traces in SCADA networks are scarce and not publicly disclosed by utility companies, the main advantage of using OCC algorithms is that they do not depend on known attack signatures to detect possible malicious traffic. As a proof-of-concept, we developed a prototype of our proposal. Finally, in our experimental evaluation, we observed the performance and accuracy of our prototype using two OCC-based Machine Learning (ML) algorithms, and considering anomalous events in the SCADA network, such as a Denial-of-Service (DoS), and the failure of several SCADA field devices.



**Keywords:** Supervisory control and data acquisition. software-defined networking. smart grids. network-based intrusion detection system. one-class classification.

**Um NIDS baseado em OCC para sistemas SCADA baseados em SDN**

## RESUMO

Sistemas elétricos possuem grande influência no desenvolvimento econômico mundial. Dada a importância da energia elétrica para nossa sociedade, os sistemas elétricos frequentemente são alvos de intrusões pela rede causadas pelas mais diversas motivações. Para minimizar ou até mesmo mitigar os efeitos de intrusões pela rede, estão sendo propostos mecanismos que aumentam o nível de segurança dos sistemas elétricos, como novos protocolos de comunicação e normas de padronização. Além disso, os sistemas elétricos estão passando por um intenso processo de modernização, tornando-os altamente dependentes de sistemas de rede responsáveis por monitorar e gerenciar componentes elétricos. Estes, então denominados Smart Grids, compreendem subsistemas de geração, transmissão, e distribuição elétrica, que são monitorados e gerenciados por sistemas de controle e aquisição de dados (SCADA). Nesta dissertação de mestrado, investigamos e discutimos a aplicabilidade e os benefícios da adoção de Redes Definidas por Software (SDN) para auxiliar o desenvolvimento da próxima geração de sistemas SCADA. Propomos também um sistema de detecção de intrusões (IDS) que utiliza técnicas específicas de classificação de tráfego e se beneficia de características das redes SCADA e do paradigma SDN/OpenFlow. Nossa proposta utiliza SDN para coletar periodicamente estatísticas de rede dos equipamentos SCADA, que são posteriormente processados por algoritmos de classificação baseados em exemplares de uma única classe (OCC). Dado que informações sobre ataques direcionados à sistemas SCADA são escassos e pouco divulgados publicamente por seus mantenedores, a principal vantagem ao utilizar algoritmos OCC é de que estes não dependem de assinaturas de ataques para detectar possíveis tráfegos maliciosos. Como prova de conceito, desenvolvemos um protótipo de nossa proposta. Por fim, em nossa avaliação experimental, observamos a performance e a acurácia de nosso protótipo utilizando dois tipos de algoritmos OCC, e considerando eventos anômalos na rede SCADA, como um ataque de negação de serviço (DoS), e a falha de diversos dispositivos de campo.

**Palavras-chave:** scada. sdn. smart grids. nids. occ.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| DNP3 | *Distributed Network Protocol 3* |
| DoS | *Denial-of-Service* |
| DPI | *Deep Packet Inspection* |
| EPA | *Enhanced Performance Architecture* |
| FCAPS | *Fault, Configuration, Accounting, Performance, and Security* |
| ForCES | *Forwarding and Control Element Separation* |
| FSP | *File Service Protocol* |
| GOOSE | *Generic Object Oriented Substation Event* |
| HMI | *Human-Machine Interface* |
| I2RS | *Interface to the Routing System* |
| ICT | *Information and Communication Technology* |
| ICS | *Industrial Control System* |
| IDMEF | *Intrusion Detection Message Exchange Format* |
| IDS | *Intrusion Detection System* |
| IEC | *International Electrotechnical Commission* |
| IED | *Intelligent Electronic Device* |
| IEEE | *Institute of Electrical and Electronics Engineers* |
| IP | *Internet Protocol* |
| ISP | *Internet Service Provider* |
| IT | *Information Technology* |
| KPCA | *Kernel Principal Component Analysis* |
| KSDMS | *Korean Smart Distribution Management System* |
| LAN | *Local Area Network* |
| ML | *Machine Learning* |
| MMS | *Manufacturing Message Specification* |
| MR | *MapReduce* |

| | |
|---|---|
| MTP | *Multipurpose Transaction Protocol* |
| MTU | *Master Terminal Unit* |
| NIDS | *Network-based Intrusion Detection System* |
| NIST | *National Institute for Standards and Technology* |
| NoSQL | *Non Structured Query Language* |
| OCC | *One-Class Classification* |
| OCRF | *One-Class Random Forests* |
| OCSVM | *One-Class Support Vector Machine* |
| OPC | *Object Linking and Embedding for Process Control* |
| OPEX | *Operational Expenditure* |
| OSI | *Open Systems Interconnection* |
| PC | *Personal Computer* |
| PCECP | *Path Computation Element Communication Protocol* |
| PES | *Power Engineering Society* |
| PMU | *Phasor Measurement Units* |
| QoS | *Quality of Service* |
| RBF | *Radial Base Function* |
| RTU | *Remote Terminal Unit* |
| SCADA | *Supervisory Control and Data Acquisition* |
| SDN | *Software-Defined Networking* |
| SMV | *Sampled Measured Values* |
| SVDD | *Support Vector Data Description* |
| SVM | *Support Vector Machine* |
| TC57 | *Technical Committee 57* |
| TCP | *Transmission Control Protocol* |
| WAN | *Wide Area Network* |

## CONTENTS

# 1 INTRODUCTION

Power grids are undergoing an intense modernization process through the use of Information and Communication Technology (ICT), evolving the traditional power system into the so-called Smart Grids (FARHANGI, 2010). Smart Grids have the potential to improve the generation, transmission, and distribution of electrical energy (YAN et al., 2013)(WANG; XU; KHANNA, 2011). Smart Grids allow a more resilient, secure, and reliable power supply for end-users, such as industries, schools, hospitals, and residences. Typically, power grids are complex environments, comprising thousands of equipment (*e.g.*, transformers, relays, fuses, or disconnectors) and devices that assist in the monitoring and control of resources, which rely on automated processes for the operation of the grid (CAHN et al., 2013). An important component of a power grid is the Supervisory Control and Data Acquisition (SCADA) system. SCADA systems are widely distributed systems responsible for monitoring, controlling, and managing automated processes and components, *e.g.* power substation and field devices, in the power grid (STOUFFER; FALCO; SCARFONE, 2011). Just as power grids are becoming Smart Grids, SCADA systems also require efforts and technologies that facilitate resource management and allow the monitoring of the proper operation of their communication networks (CAHN et al., 2013).

## 1.1 Problem and Motivation

Power grids are cyber-physical systems that have great influence on the economy and development of a country (FARHANGI, 2010). Thence, it is indispensable that power grids provide uninterrupted services to their end-users. Due to the importance of a power grid in the modern society, it is natural that its components may be targeted by cyber-attacks which aim to damage its operation (LI et al., 2012). Unfortunately, companies that manage the power grid are concerned only with maintaining their components in operation. In other words, devices are not frequently updated, and the security level of the power grid is not commonly reassessed for facing emerging intrusion techniques (CHIKUNI; DONDO, 2007). This fact can be confirmed by analyzing the SCADA systems of power grid maintainers. Differently from traditional Information Technology (IT) systems, in which their components have a lifetime on the order of 3-5 years, SCADA systems use devices that uninterruptedly operate from 15 to 20 years (ZHU; JOSEPH; SASTRY, 2011).

Most of the existing SCADA systems used in power grids were designed many years ago, without meeting basic requirements of information security, such as confidentiality, integrity, and availability (ZHU; JOSEPH; SASTRY, 2011). In the past, these systems operated in completely isolated environments, without any kind of connection to the Internet (IGURE; LAUGHTER; WILLIAMS, 2006). Theoretically, this scenario ensured that no malicious individual or organization could invade the power grid, since there were *"air gaps"* that separated SCADA systems from the Internet (CHIKUNI; DONDO, 2007). However, with the technology mod-

ernization, also came the necessity to access and perform remote maintenance of SCADA components. Thus, SCADA systems have started to become connected directly or indirectly to the Internet (FARHANGI, 2010). Although this tendency brings benefits, it allows hackers to easily collect information, invade, or damage the power grid operation, through techniques such as Denial-of-Service (DoS), eavesdropping, brute force, or even using malicious softwares (LI et al., 2012).

Currently, there is a concern with the security and the management of cyber-physical systems, *e.g.* power grids and SCADA systems. According to Dell Security Annual Threat Report (2015), the number of attacks against SCADA systems in general has doubled in 2014 if compared with the previous year. In countries such as the United States, United Kingdom, and Finland, where SCADA systems are more likely to be connected to the Internet, 202,322 attacks were registered only in 2014 (DELL, 2015). Unfortunately this number of attacks can be even higher because the companies usually do not disclose information about the suffered attacks, such as attack traces, or the impact of an attack. In addition, the rise of new techniques of networked invasion and complex viruses/malwares for SCADA systems is increasingly usual. Even with the appearance of Smart Grids, there is still a lack of adequate research in this context, because: (*i*) cyber-terrorists are turning their attention to attack these infrastructures, propagating malwares or executing cyber-attacks. If these threats are not properly detected and neutralized, they can cause outages in power supply, destroy power grid equipment, or even put lives in danger (ZHU; JOSEPH; SASTRY, 2011); and, (*ii*) it is necessary the development of solutions that improve the management of power grids and their components to reduce operational expenditures (OPEX) and to provide services with more quality to their final users.

Given the importance of power grid infrastructures, more secure communication protocols (MOHAGHEGHI; STOUPIS; WANG, 2009) and even standardization norms to enhance the security of SCADA systems were proposed (IGURE; LAUGHTER; WILLIAMS, 2006). In addition, recently, some research efforts to merge Software-Defined Networking (SDN) with SCADA systems have been carried out (CAHN et al., 2013; DONG et al., 2015; RINALDI et al., 2015). SDN is a promising network paradigm that can support the evolution of SCADA communication networks as well. SDN introduces an architecture that simplifies network operations by relying on a logically centralized element often referred to as controller (FEAMSTER; REXFORD; ZEGURA, 2013). The SDN paradigm adds the flexibility required to quickly deploy and configure new field devices and to develop more complex network services (*e.g.*, to detect cyber-attacks and misconfigurations) in the context of SCADA networks (RINALDI et al., 2015).

## 1.2 Aims and Main Contributions

Given the importance of the electrical energy to our society, power grids are often target of network intrusion motivated by several causes, such as to cause financial losses for the people

and organizations. In this context, Intrusion Detection Systems (IDSes) are essential to assist in detecting and mitigating the threats that may damage equipment or cause outages in power supply. For this reason, the main aim of this Master thesis is to propose an IDS that relies on specific techniques of traffic classification and takes advantage of the characteristics of SCADA networks and of the adoption of SDN/OpenFlow. To achieve the main aim, this document has other three secondary aims, that are:

1. to investigate the applicability and benefits of using SDN in Smart Grids environments;
2. to investigate the basic requirements of IDSes for SCADA systems;
3. to investigate the existing techniques for traffic classification in order to identify an approach that can be used for detecting anomalous behaviors in SCADA networks without relying on third-party SCADA attack traces;

The IDS proposed in this document is able to detect anomalous networking behavior of SCADA systems used in power generation, transmission, and distribution. Our solution does not rely on third-party SCADA attack traces and learns the network behavior of a SCADA system to infer possible threats, such as malware propagation, failures in field devices, or intrusion of malicious individuals in the power grid. We mainly rely on the OpenFlow protocol (FEAMSTER; REXFORD; ZEGURA, 2013) to periodically gather information about the SCADA network, and on the use of One-Class Classification (OCC) algorithms. The OCC technique can detect anomalies in datasets using initially only a homogeneous training set (KHAN; MADDEN, 2010), that in the case of SCADA systems can be the normal system functioning.

To the best of our knowledge, this is the first time that a solution that uses OCC algorithms for detecting network anomalies is presented in the specific context of SDN-based SCADA systems. Thus, the main contributions of this Masters thesis are:

1. a strategy based on SDN/OpenFlow to periodically collect information about a SCADA network;
2. a flexible NIDS based on OCC techniques to detect anomalies in SCADA networks without using third-party SCADA attack traces;
3. two case-studies that rely on simulated traditional power grids to prove the concept of the proposed NIDS's prototype.

In addition, to demonstrate the benefits of our proposed Network-based Intrusion Detection System (NIDS), we present an analysis comparing OCC-based Machine Learning (ML) algorithms. This comparison shows the efficiency of our approach to detect cyber-attacks targeted at a large-scale SCADA system for power grids. In our experimental evaluation, we observed the performance and accuracy of our prototype, considering anomalous events in the SCADA network, such as a DoS, and the failure of several SCADA field devices.

## 1.3 Document Outline

The remaining of this document is presented as follows. In Chapter 2, we present background on SCADA systems, on the SDN paradigm and its benefits in Smart Grids environments, as well as fundamental concepts of OCC techniques. In Chapter 3, we present some research efforts related to this Masters thesis, and a brief discussion about the advantages of the proposed NIDS in relation to the State-of-the-Art. In Chapter 4, we present the requirements for the proper operation of our solution. In addition, we detail the OCC algorithms used in our prototype, and we also present an overview of the architecture and its components. In Chapter 5, we describe the technologies employed in our prototype, as well as case-studies presenting the evaluation results and a performance analysis of our approach. Finally, in Chapter 6, we conclude this thesis, listing our contributions to the State-of-the-Art and lessons learned in the development of this work. In addition, we also present the final remarks and future work.

# 2 BACKGROUND

This chapter presents the necessary theoretical background for a better understanding of our work. The characteristics of SCADA systems, as well as the main architectures, communication protocols, and the main components are described in Section 2.1. Concepts about the SDN paradigm and the benefits that the OpenFlow protocol can bring for Smart Grids are presented in Section 2.2. Finally, Section 2.3 explains concepts about traffic classification, OCC techniques, and their advantages in SCADA environments.

## 2.1 Supervisory Control and Data Acquisition (SCADA)

Supervisory Control and Data Acquisition (SCADA), also known as Industrial Control System (ICS), is a kind of cyber-physical system used for controlling automated processes, monitoring resources and devices, and acquiring data on the environment in which it operates (STOUFFER; FALCO; SCARFONE, 2011). SCADA systems are largely used in industries and critical infrastructures, such as oil refineries, gas pipelines, telecommunication systems, water treatment and distribution stations, and power grids (IGURE; LAUGHTER; WILLIAMS, 2006). This kind of system has a high-level of distributed components, and is ideal for controlling geographically dispersed assets that are scattered over thousands of square kilometers, where the centralized data acquisition and control are critical to the system operation as a whole (STOUFFER; FALCO; SCARFONE, 2011).

SCADA systems are essential to improve the quality, to make flexible, and to enhance the productivity of a factory (SCIACCA; BLOCK, 1995). The main benefit of SCADA systems is the meaningful reduction of OPEX because they improve the reliability and performance of supervised processes, since humans are no longer needed for collecting information about the operational environment. Furthermore, SCADA systems permit the remote operation in a process directly from a control center, generating alarms that indicate in real time possible system faults. In addition, SCADA systems typically are able to generate reports and charts about alarms and trends. These benefits are essential to enhance the factory availability because the SCADA features permit the optimization of decision-making. Thus, SCADA systems are a fundamental piece in an automated process, and for this, they are very important to their maintainers (ZHU; JOSEPH; SASTRY, 2011).

### 2.1.1 Main Components

Generally, SCADA systems may comprise thousands of equipment to control processes and acquire information (CAHN et al., 2013). However, SCADA systems are composed, basically, of two main types of components (STOUFFER; FALCO; SCARFONE, 2011), the Master Station and its substations, which are discussed in the following:

- ***Master Station***: Also named Control Center, the Master Station comprises *SCADA servers*, *Human-Machine Interfaces* (HMIs), and the *Master Terminal Unit* (MTU). Usually, the data gathered from the operational environment is stored on SCADA servers (BOYER, 2009). In turn, HMIs are devices usually linked to SCADA servers to provide field data, detailed schematics for a particular sensor or machine, trending, and management information to the system operators. SCADA operators can send directly instructions to the automated processes through HMIs. Furthermore, HMIs are responsible for showing charts and reports about the system operation (BAILEY; WRIGHT, 2003). Finally, the MTU device is the main component of a Master Station. An MTU device receives periodically field data and operational data from intermediate stations and substations, and it permits the control of remote devices by human operators (STOUFFER; FALCO; SCARFONE, 2011). On the one hand, in smaller SCADA systems, the Master Station may be composed of a single Personal Computer (PC). On the other hand, in larger SCADA systems, the master station may include multiple servers, distributed software applications, and disaster recovery sites to enhance system resilience;
- ***Substations***: A substation is a standalone data acquisition and control unit which monitors and controls equipment at some remote location from the Master Station (BAILEY; WRIGHT, 2003). Normally, a SCADA system may contain several substations gathering information simultaneously. Substations comprise specific equipment, *e.g. field devices* and *Remote Terminal Units* (RTUs). Field devices are directly connected to the equipment that are being monitored and controlled by the SCADA system. There are two types of field devices, *sensors* (responsible for monitoring physical parameters, *e.g.* resources and the operational environment) and *actuators* (that control system modules, applying actions directly on the processes monitored by the SCADA system) (NIST, 2014). Field devices convert the physical parameters (such as, speed, level, *etc*.) into electrical signals that will be accessed by the RTU. Such signals can be analog or digital, depending on the equipment that are monitored (STOUFFER; FALCO; SCARFONE, 2011). Field devices are connected to the RTUs, which in turn, retrieve information about components and transfer this information to the MTU. An RTU is a control device, generally micro-processed, that monitors and controls equipment located in substations. RTUs are flexible, programmable, and precise devices, and they are the principal SCADA device of a substation (CHIKUNI; DONDO, 2007).

It is important to note that depending on the topology adopted (the main SCADA topologies are discussed on Section 2.1.2), in some cases, a SCADA system may use intermediate stations (also named sub-MTUs) that are similar to master stations. A sub-MTU can autonomously control the substations within its range and provide the storage of longer-term data related to their subordinate devices. However, differently from master stations, sub-MTUs periodically report their operational information and are under the control of an MTU (BAILEY; WRIGHT,

2003).

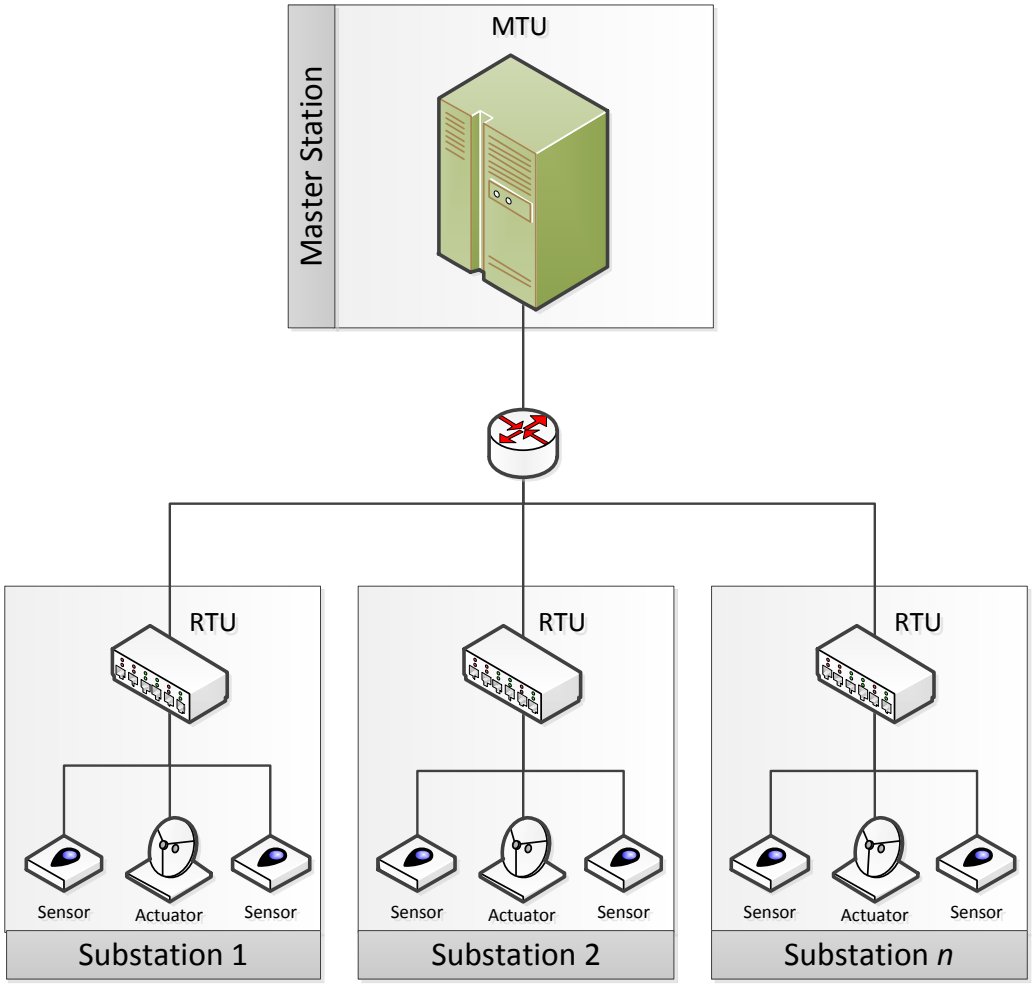
### 2.1.2 SCADA Topologies

SCADA systems are largely adopted by private companies and public-sector service providers. SCADA works well in many different types of enterprises because they can range from simple configurations to large and complex projects. Depending on the application and the environment, SCADA systems can exchange data and send commands to their processes and components in real-time (IGURE; LAUGHTER; WILLIAMS, 2006). Hence, there are different topologies used in current SCADA systems (STOUFFER; FALCO; SCARFONE, 2011). A SCADA network topology must be defined based on aims and characteristics of the system that will be controlled, the data that will be transmitted, as well as the connection speed requirements (KANG et al., 2009).

According to Kang et al. (2009), there are four basic SCADA topologies that are largely used: Monopoint, Multipoint, Multipoint with sub-MTUs, and Multiple MTUs. These topologies are discussed below:

- ***Monopoint***: This is the simplest SCADA topology (STOUFFER; FALCO; SCARFONE, 2011). In this kind of SCADA system, the data is exchanged between only two stations, *i.e.*, only one MTU can monitor an unique RTU. The stations in this topology can communicate in full duplex mode (transmitting and receiving on two separate frequencies), or in simplex mode (with only one frequency) (BAILEY; WRIGHT, 2003);
- ***Multipoint***: Similar to the previous topology, Multipoint SCADA is functionally simple; however, this topology is expensive because of the individual transmission channels required for each connection (STOUFFER; FALCO; SCARFONE, 2011). In this configuration, there is one MTU to monitor multiple RTUs. If two RTUs need to exchange information between each other, they need the arbitration of an MTU (BAILEY; WRIGHT, 2003). The Multipoint topology is presented in Figure 2.1, and it is mainly used for SCADA systems for utility services, especially in power grids (KANG et al., 2009);
- ***Multipoint with sub-MTUs***: Large-scale SCADA systems, containing hundreds of RTUs, often employ sub-MTUs to alleviate the burden on the MTU (STOUFFER; FALCO; SCARFONE, 2011). Very similar to Multipoint SCADA, this topology relies on sub-MTUs for directly controlling RTUs. In this topology, sub-MTUs also report their data to an MTU. The Multipoint with sub-MTUs topology also is largely adopted in SCADA systems for power grids (KANG et al., 2009);
- ***Multiple MTUs***: This topology permits the adoption of sub-MTUs if necessary. The Multiple MTUs topology is often used to enhance the resilience of the SCADA system. Thus, it is possible to use a second MTU that provides redundancy in the event of a primary MTU malfunction (STOUFFER; FALCO; SCARFONE, 2011). Furthermore,

Figure 2.1: A traditional Multipoint SCADA system.




Source: by author (2016).

this topology also is used, in specific cases, to enable the control of an RTU by more than one MTU. In addition, the MTUs can act in a peer-to-peer communication manner. This is a more complex arrangement requiring sophisticated protocols to handle packet collisions between different stations wanting to transmit data at the same time (BAILEY; WRIGHT, 2003).

### 2.1.3 Types of SCADA

Over the years, SCADA systems were also adapted to follow technological innovations (*e.g.*, the appearance of new transmission techniques, and more sophisticated equipment), to increase the security level of components, and to provide more accurate monitoring (SAYEGH et al., 2014). According to McClanahan (2002), the development of SCADA systems are divided in three generations that are described below:

- ***First Generation***: This generation is also known as monolithic, and the concept of SCADA

systems were centered in mainframe systems. In this context, SCADA systems only used proprietary communication protocols, and operated in completely isolated environments, without any kind of connection to the Internet (IGURE; LAUGHTER; WILLIAMS, 2006). This environment ensured that malicious individuals did not invade the master station and manipulate the system, since there were *"air gaps"* separating SCADA systems from the Internet (CHIKUNI; DONDO, 2007). Furthermore, the connectivity with the master station was limited. Thus, RTUs only sent information to the MTU when requested (BAILEY; WRIGHT, 2003);

- ***Second Generation***: The second generation of SCADA systems was characterized by the miniaturization of components and the distribution of processing across multiple systems. Distributed SCADA systems contained multiple stations, each one with a specific function. These stations were connected to a Local Area Network (LAN) and shared information with each other in real-time. As well as the first generation, distributed SCADA systems used proprietary components (such as, hardware, software, and peripherals) based on closed platforms that were incompatible with other available solutions (MCCLANAHAN, 2002);
- ***Third Generation***: The major improvement in the third generation was the utilization of open standards and communication protocols. This characteristic enabled to use of supervisory functions not only in LANs, but also in long-range networks, such as Wide Area Networks (WANs) (STOUFFER; FALCO; SCARFONE, 2011). The adoption of open standards eliminated several limitations inherited from previous generations. In addition, there was a large adoption of IP-based protocols for communicating SCADA components. The disaster survivability is another advantage enabled through the distribution of SCADA systems in WANs. In other words, distributing the processing across distinct locations allowed to develop SCADA systems that can continue in operation even if an MTU or RTU is completely destroyed (MCCLANAHAN, 2003).

### 2.1.4 Communication Protocols

When the first SCADA systems were developed, their communication protocols possessed the goal to only provide good performance, ensuring that the procedural requirements would be met (IGURE; LAUGHTER; WILLIAMS, 2006). Thus, oil refineries, power grids, other industries, and the manufacturers of control devices developed their own protocols and communication structures. Over the years, with different kinds of clients requesting for personalized equipment, adequate industrial protocols had to be developed, generating an immense diversity of available solutions (ALMALAWI, 2014). Nowadays, there are about over 200 protocols being used in the various types of SCADA in the world (IGURE; LAUGHTER; WILLIAMS, 2006).

The pioneers SCADA protocols were designed to operate over serial media. Most of these

protocols were proprietary standards developed by individual companies (IGURE; LAUGHTER; WILLIAMS, 2006), such as Modbus (HUITSING et al., 2008), Distributed Network Protocol 3 (DNP3) (CLARKE; REYNDERS; WRIGHT, 2004), Fieldbus (THOMESSE, 2005), and Profibus (BENDER, 1993). These protocols were simple and easy to implement, and they met the requirements of legacy SCADA systems. However, due to the increasing number of interconnected devices and the complexity level of these cyber-physical systems, the industry has moved into accepting common open standard protocols (IGURE; LAUGHTER; WILLIAMS, 2006), *e.g.* the adoption of TCP/IP-based communication. Thus, many protocols have been ported to operate over the TCP/IP stack, and new protocols have been proposed. Although there are a wide range of SCADA protocols *e.g.* the emerging WirelessHART protocol, EtherCAT, ControlNet, the BSAP protocol, nowadays, the most used communication protocols in SCADA systems are Modbus, DNP3, and IEC-61850 (BARBOSA, 2014). These three last protocols are discussed in more detail below:

- ***Modbus***: Introduced in 1979 by Schneider Electric, Modbus is one of the oldest, but most widely used industrial control protocols (HUITSING et al., 2008). The Modbus protocol is an open communication standard, supported by a large number of products and vendors on the market today. The open specification and the TCP extension of Modbus have contributed to its popularity, especially in the oil and gas sector (EDMONDS; PAPA; SHENOI, 2008). This protocol is the *de facto* standard for process control networks. Modbus establishes the rules and the structure of messages used by SCADA equipment to communicate amongst themselves (FOVINO et al., 2010). The Modbus application layer defines the mechanisms by which devices exchange data for operating and controlling industrial processes (SWALES, 1999). As Modbus is a protocol which is independent of the physical network layer, Modbus serial line can be integrated seamlessly into Modbus TCP networks, using simple gateways (AL-DALKY et al., 2014). This is transparent for the application. The Modbus protocol relies on a simple request/reply communication mechanism between a master device and slave devices (FOVINO et al., 2010). For example, an MTU (master) might send a "read" message to an RTU (slave) to obtain the value of a process parameter (*e.g.*, the temperature of a particular equipment). Alternatively, the MTU might send a "write" message to an RTU to perform a control action (*e.g.*, open a specific valve). Furthermore, this protocol supports unicast and multicast messages and it is relatively easy to implement (EDMONDS; PAPA; SHENOI, 2008).
- ***DNP3***: Over the last decade, DNP3 has emerged as one of the most prevalent international standard protocols in the SCADA market. DNP3 is an open protocol developed by Harris Controls Division, Distributed Automation Products in the early 1990s and it was released to the industry based DNP3 Users Group in November 1993 (CLARKE; REYNDERS; WRIGHT, 2004). The DNP3 protocol was developed to achieve interoperability among systems in the electric utility, oil and gas, water/waste water and security indus-

tries, and it is the dominant protocol for SCADA system in North America, Australia, and China (FOVINO et al., 2010). DNP3 is specifically developed for interdevice communication involving RTUs, and provides for both Field Device to RTU and MTU to Field Device/RTU. DNP3 permits the multipoint communication between a master device and one or more slave devices. Each device is identified by an unique address, that ranges from 0 to 65519 (LU et al., 2011). In addition, this protocol permits the emission of frames in diffusion, *i.e.*, when a master device sends a message, this message propagates to all other nodes of the topology. The DNP3 protocol was designed based on Enhanced Performance Architecture (EPA) (STRAYER; WEAVER, 1988). EPA is a simplified variation of the Open Systems Interconnection (OSI) model which contains only three layers, physical, data link, and application (STRAYER; WEAVER, 1988). However, this protocol contains segmentation and rebuilding functions to enable the transmission of messages with 2 or more kbytes, creating a pseudo-layer of transport (LU et al., 2011). Different from Modbus, the DNP3 data link layer manages the communication in an equilibrated mode, which means that either device, master or slave, can initiate the transmission of messages. To solve possible conflicts in the emission of messages, the DNP3 protocol contains a mechanism of collision management (CLARKE; REYNDERS; WRIGHT, 2004).

- ***IEC-61850***: IEC-61850 was released in 2004 and it is a part of the International Electrotechnical Commission's (IEC) Technical Committee 57 (TC57) reference architecture for electric power systems (MACKIEWICZ, 2006). The scope of IEC-61850 was originally focused on substations, however, there are currently discussions to define IEC-61850 for the RTU to MTU communication protocol (DORSCH et al., 2014). The IEC-61850 norm is a standardized data model based on principles of object orientation (MACKIEWICZ, 2006). Different from the other protocols, IEC-61850 defines several aspects of the substation communication network, such as the requirements, the system and project management, communication requirements, basic communication structures, and so on (BRUNNER, 2008). Although this standard defines several aspects in a substation, in this section we will only discuss the protocols used by IEC-61850. The communication paradigm of IEC-61850 relies abstract data models that can be mapped to a number of protocols. Hence, instead of determining a specific protocol for each layer, IEC-61850 permits the mapping of objects and services to other protocols that attend the required data and services (MACKIEWICZ, 2006). Currently, the abstract data models defined in IEC-61850 are mapped into three three protocols, *e.g.* Manufacturing Message Specification (MMS) (WANG et al., 1994), Generic Object Oriented Substation Event (GOOSE) (MOHAGHEGHI; STOUPIS; WANG, 2009), and Sampled Measured Values (SMV) (LEE et al., 2008). The MMS protocol relies on unicast messages to exchange analog and digital data. Generally, the MMS messages indicate the state of a particular equipment. In turn, GOOSE messages are used for multicast communications. The GOOSE protocol is responsible for information about the operation of any protection or

digital signal. Differently from MMS, GOOSE is based on UDP datagrams. Finally, SMV is used for analog readings of equipment into the substation (SIDHU; YIN, 2007).

Unfortunately, legacy SCADA protocols were developed focused only on performance, placing in background security aspects (CHIKUNI; DONDO, 2007). For example, the IP-versions of Modbus and DNP3 offer weak packet confidentiality assurance. Thus, these protocols allow, for example, the information exchange between SCADA devices to be transmitted in clear text. Although SCADA systems were originally conceived to operate in completely isolated environments, nowadays these systems are highly interconnected and often linked to the Internet. This trend was strongly motivated by the necessity to perform maintenance and to access remotely SCADA components but it also potentially enables malicious individuals to access the system (MCCLANAHAN, 2003). Due to the importance of these cyber-physical systems, an targeted cyber-attack can result in catastrophic consequences.

In addition to the IEC, there are also other efforts that focus on standardizing cyber-physical system and several professional organizations have been developing standards to improve the security of SCADA systems. For example, The Institute of Electrical and Electronics Engineers Power Engineering Society (IEEE-PES) has a working group for addressing issues of risk assessment of information security in SCADA networks. The Object Linking and Embedding for Process Control (OPC) Foundation is also another organization working towards open connectivity in industrial automation using open standards. OPC has developed standards for implementing data access, alarms, event management, and even Web access to SCADA network devices (IGURE; LAUGHTER; WILLIAMS, 2006). Furthermore, The National Institute for Standards and Technology (NIST) released a complete guide to ICS security. NIST also focuses efforts on enhancing the security of master stations (STOUFFER; FALCO; SCARFONE, 2011).

## 2.2 Software-Defined Networking (SDN)

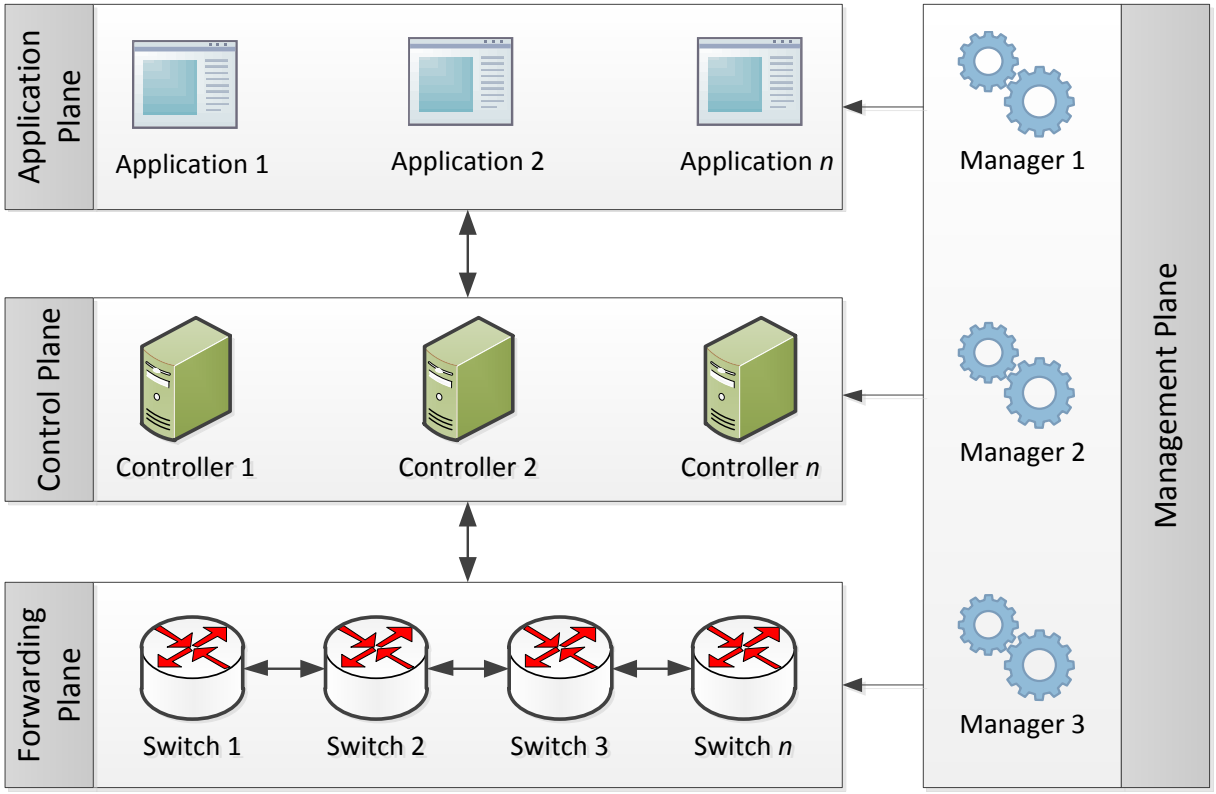
In Section 2.2.1 is presented fundamental concepts of the SDN paradigm, such as the advantages of this architecture, the main SDN implementations, and its main components. In addition, the advantages of merging SDN in the context of SCADA systems and Smart Grids is described in Section 2.2.2.

### 2.2.1 The SDN Paradigm

Software-Defined Networking (SDN) is an emerging architecture for controlling, managing, and monitoring network traffic and switching devices. It is a dynamic, adaptable, controllable, and flexible architecture that provides an extensible platform for delivery of network services, capable of responding quickly to requirement changes (MONSANTO et al., 2013). Differently from traditional networks, SDN decouples the network control and the forwarding planes. In

addition to this characteristic, SDN also provides other two planes, the application plane, and the management plane. Thus, the control plane is responsible for decision making, updating flow tables of switching devices, and for implementing routing protocols. In turn, the forwarding are responsible for switching packets. The application plane executes SDN applications on the network, such as network visualization, load balancers, and firewalls (KREUTZ et al., 2015). Finally, the management plane is responsible for monitoring, configuring, and maintaining the components of each plane in an SDN environment (WICKBOLDT et al., 2015). Figure 2.2 illustrates the SDN architecture and its components.

Figure 2.2: SDN architecture.




Source: by author (2016).

In traditional networks, the control plane is coupled and is executed in each network device. Thus, often it is not possible to manage situations that have not been anticipated (SEZER et al., 2013), such as the network growing, or the addition of a new communication protocol. However, SDN can simplify the network management because it offers to network programmers a comprehensive view of the network and the ability to control devices from a centralized control plane, since the decision-making logic is not in network devices but in external controllers (FEAMSTER; REXFORD; ZEGURA, 2013). Further, each traditional device has its proprietary protocols. This makes the network devices complex to configure. The OpenFlow protocol (MCKEOWN et al., 2008) solves this problem. OpenFlow is an SDN implementation proposed to standardize the communication between the control plane and the forwarding plane.

From the traditional network management point-of-view, there are several benefits in adopting SDN. It simplifies or even solves critical management tasks. For example, a traditional network management task, network discovery, is naturally solved with SDN because its devices need to be registered or discovered by the network controller in order to establish a communication path between control and forwarding planes (WICKBOLDT et al., 2015).

Due to the benefits of SDN in relation to traditional networks, there are others initiatives that implement this promising network paradigm, such as Forwarding and Control Element Separation (ForCES) (SEZER et al., 2013), Path Computation Element Communication Protocol (PCECP) (PAOLUCCI et al., 2013), and Interface to the Routing System (I2RS) (HARES; WHITE, 2013). However, OpenFlow stands out as the most widespread and important protocol for SDN implementation. OpenFlow was proposed for standardizing the communication between control and forwarding planes. This protocol defines how applications running on the control plane can program the behavior of each network switch. Usually, an OpenFlow-based SDN network consists of the following components (MCKEOWN et al., 2008):

- ***OpenFlow Switches***: Data forwarding devices that use a flow table to forward packets;
- ***Flow Table***: A table that contains a list of flow entries and associated actions to be applied to the respective flows;
- ***OpenFlow Controller***: Software component that manipulates and controls the flow tables of switches;
- ***Secure Channel***: Communication channel that connects each switch to a controller and allows the OpenFlow controller to install flow rules. This secure channel enables the controller to manage and control all network switches, and to send and receive control messages to and from the switches.

### 2.2.2 SDN-based SCADA Systems

The incorporation of SDN into SCADA systems for Smart Grids emerges as a promising research area, since SDN can help in the evolution of SCADA communication networks, facilitating the development of network applications, and collaborating with the power grid modernization. Thus, in this document we advocate the use of SDN to assist in the management of SCADA systems. SDN can enable more flexible SCADA networks, since the addition of new policies and services requires changing the control plane only. Arguably, the use of SDN in SCADA will support more resilient systems, as solutions to detect and mitigate cyber-attacks and other threats can be more easily implemented in the controller (SILVA et al., 2015b). SCADA systems can benefit from the characteristics of SDN in several ways:

- ***Centralized Management***: The centralized control plane offers a global view of the network (WICKBOLDT et al., 2015). Thus, an SDN-based SCADA master station will be able to manage not only SCADA devices, but also monitor and control the network that

interconnects these devices;

- ***Flexibility***: SDN enables more flexible systems (FEAMSTER; REXFORD; ZEGURA, 2013), in which network applications and communication protocols can be modified via a logically centralized controller. In SCADA systems, this will permit easily adding new equipment (such as, HMIs, RTUs, and field devices) or upgrading existing network applications in SCADA networks;
- ***Programmability***: It is possible to easily add new functionality to the network on demand *via* the SDN controller. In SCADA, this will allow creating a range of customized services, *e.g.* to control the reading frequency of field devices at a specific time of day, to perform load balancing between communication links, to optimize the operation of system components, or even to identify and mitigate traffic anomalies;
- ***Standard API***: The OpenFlow protocol provides a standard API for controlling the behavior of network switching devices. In SCADA networks, this standardization will permit a better integration of geographically dispersed equipment from different vendors.

In addition, the characteristics of SDN can also enhance *Fault, Configuration, Accounting, Performance, and Security* (FCAPS) management in Smart Grids (SILVA et al., 2015b). The possible benefits of SDN-based SCADA systems for each FCAPS property are presented below:

- ***Fault***: SDN enables the implementation of mechanisms for increasing the resilience of SCADA systems. The centralized view of the SDN controller allows more efficient fault detection techniques, isolation of compromised components, and remediation of abnormal operation in Smart Grid networks caused by both misconfigured equipment or malicious softwares;
- ***Configuration***: The OpenFlow protocol provides a standard API for the correct configuration of new devices added to the SCADA network and their communication protocols. This can reduce the configuration overhead of these components, since an unique distribution substation is usually composed of thousands of field devices, such as sensors, circuit breakers, actuators, relays, and transformers (CAHN et al., 2013);
- ***Accounting***: The measurement capabilities of the OpenFlow controller provides the ability to collect metrics and statistics about the network traffic through native counters (SILVA et al., 2015a). This information can be used in dimensioning the capacity of the SCADA network, to plan the growth of the power grid, or to detect abuses in resource usage and intruders in the power grid;
- ***Performance***: An SDN-based SCADA system can facilitate the use of Quality of Service (QoS) policies in Smart Grid environments, to perform load balancing between communication links and to optimize the operation of system components;
- ***Security***: The controller also permits the network implementation of applications that can add more security to Smart Grids, *e.g.* in terms of protecting the information exchanged in

SCADA networks, or mainly creating sophisticated mechanisms for traffic classification that detect malicious activities on SCADA environments.

## 2.3 Traffic Classification

In Section 2.3.1 the definition of traffic classification and the techniques based on Machine Learning is presented. Finally, fundamental concepts of One-Class Classification and its examples are described in Section 2.3.2.

### 2.3.1 Traffic Classification and Machine Learning

In computing, classification is the task of learning a target function that maps each new instance into one of the predefined classes (KHAN; MADDEN, 2010). Thus, traffic classification can be defined as the task of associating a particular network traffic according to the application that generated it. Traffic classification techniques can be used for clustering IP traffic flows into groups that have similar traffic patterns, or for classifying one or more applications of interest (NGUYEN; ARMITAGE, 2008). Traffic classification techniques are capable of identifying patterns in the sampled network traffic that, for example, may indicate malicious traffic, such as DoS attacks, or malware propagation.

Many security-related tools, such as anti-virus and anti-malware applications rely on traffic classification to detect the application behind a given IP flow (ESTE; GRINGOLI; SALGARELLI, 2009). In the context of Internet Service Providers (ISPs), traffic classification enables the identification of traffic patterns, and what classes of applications are being used by a user at any given point in time (NGUYEN; ARMITAGE, 2008). Hence, the ability of assigning traffic flows to classes of service is seen as a priority by many ISPs. The algorithms for traffic classification are essential for advanced network management and traffic engineering (ESTE; GRINGOLI; SALGARELLI, 2009).

Traditional traffic classification techniques rely on the inspection of TCP packets or UDP port numbers, or the reconstruction of protocol signatures in packet's payload (ERMAN; ARLITT; MAHANTI, 2006). These approaches suffer several limitations, *e.g.* if an application adopt an unknown port to avoid detection, or if the application's packets are encrypted (DAINOTTI; PESCAPE; CLAFFY, 2012). To address the aforementioned drawbacks, many researches proposed the adoption of Machine Learning (ML) techniques for traffic classification, creating an inter-disciplinary blend of IP networking and data mining techniques (NGUYEN; ARMITAGE, 2008). Traffic classification techniques based on ML fall into two main categories:

- ***Supervised Learning:*** In supervised learning, classes are defined *a priori*, and samples are given to the system already labeled with classes (CARUANA; NICULESCU-MIZIL, 2006). Supervised algorithms require an initial step named training step. In this step,

the classifier learns the profile of one or more predefined target classes. Thus, the classifier will be ready for classifying new samples. This technique has achieved results comparable to Deep Packet Inspection (DPI) (DAINOTTI; PESCAPE; CLAFFY, 2012). Support-Vector Machines (SVM) (CORTES; VAPNIK, 1995) and Naive Bayes (LEWIS, 1998) are examples of supervised algorithms;

- ***Unsupervised Learning:*** Differently from supervised learning, unsupervised algorithms do not require a training step, and they identify distinct classes and assign samples to the classes (clustering) (DAINOTTI; PESCAPE; CLAFFY, 2012). Unsupervised learning requires manual input to determine the classes of the clustered data. However, this technique does not use historical information or a data model to produce the data clustering, but only the similarities observed in the samples (ALMALAWI, 2014). Examples of unsupervised algorithms are K-Means (JAIN, 2010), DBSCAN (BORAH; BHATTACHARYYA, 2004), and AutoClass (CHEESEMAN et al., 1993).

The number of classes also defines a classifier. Binary and multiclass classifiers are the most known ML techniques for traffic classification. On the one hand, binary classifiers are algorithms trained with positive and negative target classes. Thus, if a binary classifier has a training set containing two classes (*e.g.*, FTP and HTTP traffic), the classifier will be able to classify a new sample into one of these two classes (ESTE; GRINGOLI; SALGARELLI, 2009). On the other hand, a multiclass classifier can use a training set composed of several target classes (such as, HTTP, FTP, BGP, and DNS) and, consequently, it will be able to predict a new instance into one of the target classes (CARUANA; NICULESCU-MIZIL, 2006). However, binary and multiclass classifiers require the set of instances that characterize the target classes in the training step to operate. These approaches are not feasible if, in the classification step, a new traffic profile (a novelty) in relation to the existing classes on the training set emerges (KHAN; MADDEN, 2010). Thereby, a new protocol may be erroneously classified into an existing category by a multiclass classifier specialized for traffic classification.

### 2.3.2 One-Class Classification (OCC)

ML offers a wide range of mechanisms that can be applied for traffic classification and for detecting intrusions in different scenarios (CHANDOLA; BANERJEE; KUMAR, 2009). In this context, OCC algorithms were designed to alleviate the restriction that traditional classifiers (binaries and multiclasses) have regarding the training step and novelty detection. The OCC technique creates a model featuring a single class and it can be used when only one class of training samples is available (ESTE; GRINGOLI; SALGARELLI, 2009). In other words, differently from binary and multiclass classifiers, the OCC approach is a special case of supervised learning that only examples of the unique class are available in the training set. Thus, instances of this class are named target instances. However, all other instances are *per* definition

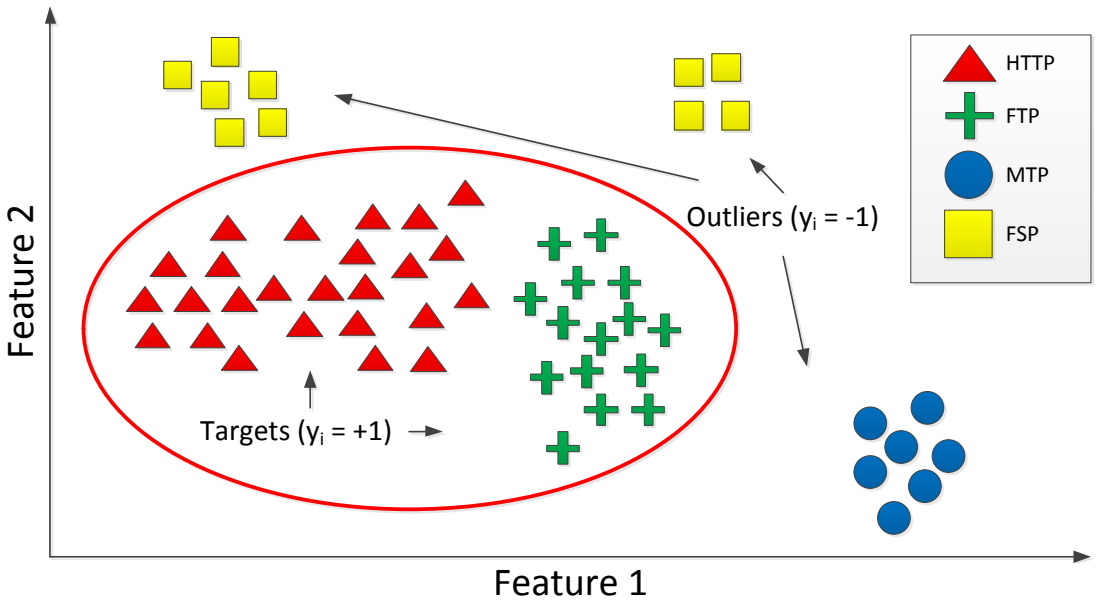
outliers (TAX, 2001).

The OCC problem is to describe instances of the target class and to detect new objects that resemble the training set (LENG et al., 2015). We chose to study OCC because they produce very accurate classifiers and, there are several real-world applications based on OCC techniques responsible for detecting: frauds in payrolls and in e-commerce operations, clinical anomalies, and anomalies in communication networks (KHAN; MADDEN, 2010). Mainly in the context of traffic classification, the OCC paradigm can be used to detect unexpected behaviors in a communication, *i.e.* OCC can be used as an anomaly detection mechanism (JANSSENS, 2013). In communication networks, unexpected behaviors may represent the occurrence of malicious activities, creating a real necessity for sophisticated detection mechanisms to prevent the network from service degradation (BARBOSA, 2014). Thus, IDSes that rely on OCC algorithms are trained to recognize the normal network activities, being able to determine what is an attack or misconfiguration on the data traffic.

In Figure 2.3 there is an example of a one-class classifier projected in a feature space. In this example, TCP-based protocols (HTTP and FTP) are the target instances and are labeled $y = +1$. In turn, the instances that represent UDP-based traffic are classified as outliers $y = -1$. In our example, File Service Protocol (FSP) and Multipurpose Transaction Protocol (MTP) datagrams are classified as outliers. The red line shows a possible one-class classifier which distinguishes between TCP-based traffic and outliers. Note that this explanation serves as the basis to the OCC approach. A more detailed explanation and the formalization of OCC algorithms is presented in Section 4.2.

IDSes based on OCC algorithms can enhance the security level of SCADA environments. In the context of SCADA systems, we intend to investigate the development and implementation of OCC-based anomaly detection mechanisms will bring the same benefits achieved by existents techniques based on ML for general networks. OCC algorithms do not rely on malicious traffic signatures, but instead they need only the expected traffic (benign behavior) for building a classifier model, making the detection process faster and more accurate. Since an OCC-based NIDS does not require attack signatures, it is well suited for intrusion detection in SCADA systems (NADER; HONEINE; BEAUSEROY, 2013). Assuming SDN allows periodically gathering precise statistics in SCADA networks, it is possible to use this information for creating a model of the normal and expected behavior of a SCADA system. Thus, this behavioral model in combination with OCC algorithms can be used for building a resilience mechanism for detecting cyber-threats in SCADA networks.

Figure 2.3: Example of a one-class classifier for traffic classification.




Source: by author (2016).

# 3 RELATED WORK

In this chapter, we present some efforts related to this Masters thesis. Firstly, in Section 3.1, we discuss about investigations which present the benefits that can be achieved by adding SDN in Smart Grid environments. In addition, Section 3.2 lists efforts that present solutions for detecting cyber-intrusions in SCADA systems. Finally, Section 3.3 presents a brief discussion about the advantages of our solution in relation to the state-of-the-art.

## 3.1 SDN in Smart Grids

In the past few years, the number of researches that exploit the benefits of merging SDN in Smart Grid environments has grown considerably. This fact together with the power grid importance for the modern society makes this research topic highly relevant, and any initiative aimed to make this environment more reliable, more flexible, and safer, is highly important. It is important to emphasize that there are researches presenting the benefits of SDN in Smart Grids and their several components, such as Zhang et al. (2013), Dorsch et al. (2014), Dong et al. (2015), and Rinaldi et al. (2015), including SCADA systems. However, although there are many efforts generically discussing the impact and consequences of using SDN in Smart Grids, to the best of our knowledge, these are still scarce initiatives that debate the applicability of SDN in specific parts of the power grid.

Dong et al. (2015) investigate *(i)* how SDN can enhance the resilience of Smart Grids against malicious attacks, *(ii)* additional risks that can be introduced adopting SDN and how to manage them, and *(iii)* how to evaluate and validate solutions for SDN-based Smart Grids. In addition, this paper also discusses concrete security issues and their possible countermeasures in the context of Smart Grids. In turn, Dorsch et al. (2014) present and analyze an SDN-based approach for dynamic network control, meeting the specific communication requirements of power grid transmission and distribution subsystems. In this work it is also discussed challenges related to the adoption of SDN in Smart Grid communication networks. Song et al. (2013) define and classify smart control functions that can be implemented in Smart Grid environments through the adoption of SDN. According to the authors, some of the operation strategies proposed in the paper will be implemented into the Korean Smart Distribution Management System (KSDMS). In addition, the effectiveness of these solutions are evaluated through case studies.

In a previous work, we discussed the potential benefits that SDN can bring to the power grid and more specifically to the SCADA systems (SILVA et al., 2015b). The aforementioned paper presents a multipath approach for SDN-based SCADA system in which communication of SCADA devices is performed by more than one route, in order to prevent possible eavesdroppers from fully capturing messages exchanged between SCADA devices of the distribution power system. In this context, Cahn et al. (2013) also present a solution for power distribution subsystems. The authors use SDN for allowing the network to be auto-configurable, secure and

reliable against possible system misconfiguration. A prototype was developed using the Ryu OpenFlow controller and evaluated in a testbed with real SCADA devices. Focusing on the transmission subsystem, Goodney et al. (2013) propose the use SDN to control the communication between devices responsible for measuring electrical waves in the grid, known as Phasor Measurement Units (PMUs). The authors developed an SDN-based network application to facilitate the management of PMUs and provide support for multicast and multi-rate, essential features for PMU networks.

To solve particular problems of power grids, Gyllstrom, Braga and Kurose (2014) present algorithms to make fast recovery from link failures. The proposed algorithms permit the solution to detect and report where and when occurred link failure using OpenFlow. Furthermore, the authors present algorithms for computing backup multicast trees, and fast backup tree installation. Pfeiffenberger et al. (2015) demonstrate an efficient solution that can be used to solve the problem of robust multicast in power grid substations. The solution proposed in this paper uses the fast-failover groups feature of OpenFlow to provide one-link fault tolerance. The results showed that this solution provides little packet loss and routes that are less susceptible to fail. Finally, Kim et al. (2014) present an SDN-based solution for creating virtual network slices in a Smart Grid environment. The main idea of this proposal is to enable the network to be self-configurable, defining virtual network slices. Each slice can support one application or a group of similar applications. Thus, the authors argue that their solution will provide secure and cost-efficient communications for Smart Grid applications.

## 3.2 SCADA IDSes

Unfortunately, several SCADA systems currently in operation have vulnerabilities in their mechanisms of basic security, such as access control and user authentication. Although the adoption of reliable mechanisms of authentication and access control in this context is feasible, costs of development and deployment restrain the incorporation of those service to the SCADA systems. In addition, even secure systems might have vulnerabilities occasioned by misconfigurations, errors, or by intruders. Hence, the number of researches that propose IDSes as a complementary approach of security for protecting SCADA systems is increasing (BARBOSA, 2014).

Bigham, Gamez and Lu (2003) were pioneers in evaluating how the accuracy and security of SCADA systems can be improved using anomaly detection to identify incidents caused by faults and cyber-attacks. This paper compares the performance of invariant induction and n-gram anomaly-detectors. In addition, the authors proposed the integration of the output from several anomaly-detecting techniques using Bayesian Networks. Linda, Vollmer and Manic (2009) use Neural Networks for detecting intrusion in a SCADA environment. The Neural Network learns the normal contents of a window that is calculated over a sequence of $N$ packets.

Several classical ML methods are tested by Duessel et al. (2010). This approach consists

of extracting traffic information using Bro[1], then applying a different combination of feature extraction methods, similarity measures and anomaly detection methods. Maglaras and Jiang (2014) presented an intrusion detection module based on OCSVM capable of detecting malicious network traffic in SCADA systems. The OCSVM module developed is trained by off-line network traces and detect anomalies in real time. This module is part of an IDS developed under CockpitCI project[2] and communicates with other components by IDMEF (Intrusion Detection Message Exchange Format) messages. These messages contain information about the source of incident, time, and a classification of the alarm.

Cheung et al. (2007) proposed an IDS based on behavioral models for SCADA networks. This IDS creates models that represent the expected network behavior of the devices that are connected to a SCADA system. The authors point out that SCADA systems have topologies that hardly change over time, and thus the behavior of the devices maintains a pattern. This facilitates the detection of possible attacks that may cause changes to the expected network behavior. Oman and Phillips (2008) proposed a hybrid between an IDS and a configuration tool. The IDS internally uses models representing the allowed traffic patterns. The authors propose the use of the Telnet protocol to configure and test the connectivity of field devices. The configuration of these devices is periodically retrieved and stored, so it can later be restored in case of misconfiguration or to recover from cyber-attacks. A proof-of-concept was implemented and deployed in a testbed environment.

In the approach described by Valdes and Cheung (2009), flow-level metrics extracted from the network traffic are compared to historical values. If the difference between current and historical values is discrepant, the flow is marked as anomalous. In addition, alarms are also defined as *new* flows (flows for which no historical data is known) and *missing* flows (flows not observed after a certain time). This approach is implemented in a testbed environment. D'Antonio, Oliviero and Setola (2006) presented a distributed architecture to secure the communication network upon which the critical infrastructure relies. This architecture is composed by an IDS that is built on the top of a customizable flow monitor. The authors also proposed a method to extrapolate real-time information about user behavior from network traffic. This method consists in monitoring traffic flows at different levels of granularity in order to discover ongoing cyber-attacks.

Premaratne et al. (2010) proposed a signature-based approach focused on SCADA environments. By observing traffic generated by field devices, the authors manually derive rules characterizing the cyber-attack behavior (for example, ICMP packets larger than 100 bytes indicate a Ping DoS). In turn, Yang et al. (2013) presented a rule-based IDS that relies on DPI. This IDS uses signature-based and model-based approaches tailored for SCADA systems. The proposed signature-based rules can detect known suspicious or malicious attacks. In addition, model-based detection is proposed as a complementary method for detecting unknown cyber-attacks.

[1] https://www.bro.org/

[2] http://www.cockpitci.eu/

Sayegh et al. (2014) proposed an IDS that uses temporal packet data to identify traffic anomalies. In this work, different packet signatures are generated for a given protocol, and probability functions are used to identify if a given packet is expected to arrive to the SCADA system. The paper targets the BACnet protocol but mentions that new protocols can be supported without changing the IDS core functions.

In a series of papers, Carcano et al. (2010), Fovino et al. (2010), Carcano et al. (2011) proposed an IDS capable to analyze Modbus/DNP3 packets. This approach relies on two detection strategies: *(i)* a single packet signature-based strategy, allowing to detect illicit packets sent to PLCs and RTUs; and, (*ii*) a state-based intrusion detection technique, allowing to keep track of the industrial system state, and to identify if a set of licit SCADA commands sent to field devices is able to bring the system into a critical state. In addition, the authors presented a rule language designed in order to describe network signatures and field device states.

Other techniques are also proposed for detecting intrusions in SCADA environments. A process-aware approach to detect intrusion is proposed by Cardenas et al. (2011). An interesting aspect of this paper is the evaluation of the impact of different realistic attack scenarios and the discussion of responses to these attacks. Parthasarathy and Kundur (2012) exploit the predictable and regular nature of SCADA communication patterns to detect intrusion in field devices. The authors proposed a distributed and lightweight IDS suitable for implementation across multiple resource constrained SCADA devices in the Smart Grid. This approach uses the Bloom Filter data structure for memory efficiency and incorporates the physical state of the power grid for greater robustness. Asif and Al-Harthi (2014) designed an IDS that relies on the Honey Token based Encrypted Pointers to protect SCADA networks from cyber-attacks. These honey tokens inside the frame serve as a trap for the cyber-attacker. This IDS is designed for detecting intrusions and for recovering the system using reverse engineering approach.

In his thesis, Almalawi (2014) proposed an unsupervised SCADA data-driven anomaly detection approach intended to be used as a passive SCADA IDS. This IDS has two main steps: (*i*) the identification of consistent and inconsistent states from unlabeled SCADA data traffic generated by system sensors and actuators using the density factor for the $k$-nearest neighbors of the observation; and, (*ii*) the extraction of proximity-based detection rules for normal and anomalous behavior using statistically determined micro-clusters (ALMALAWI et al., 2014). Barbosa, Sadre and Pras (2012) investigated the main traffic characteristics in SCADA networks and presented a NIDS capable of detecting data injection and DoS attacks (BARBOSA, 2014). This NIDS specifically explores the periodicity of traffic generated in SCADA systems.

## 3.3 Related Work Discussion

As showed in sections 3.1 and 3.2, there are several papers that propose NIDSes for SCADA systems. Although this research topic has been growing, there are few researches that exploit large-scale SCADA topologies in their experimental evaluations. It is also important to note that

is scarce the papers that present clearly results, highlighting the accuracy of their proposals. The proposal presented by Maglaras and Jiang (2014) is a exception, presenting an aproximately acurracy of 98% using OCSVM in a traditional SCADA system. However, Maglaras and Jiang (2014) does not present detailed results, with specific ML evaluation metrics and performance metrics.

We investigated IDSes currently available for SCADA systems and the network behavior of SCADA devices to contribute to this research topic. Several detection mechanisms have been proposed to mitigate anomaly behaviors in SCADA systems, however, many solutions require to know *a priori* the system inconsistent state. To collect information about a system inconsistent state can be expensive, or just impossible. Few proposals rely on ML to detect intrusions in SCADA systems (*e.g.*, Linda, Vollmer and Manic (2009) and Duessel et al. (2010)), but these papers also require to learn an inconsistent state in the training step. In this context, the utilization of OCC algorithms in IDSes for SCADA systems is still starting, Maglaras and Jiang (2014) is an example. For this reason, we will propose on Section 4.3 a complete solution to detect anomalies in SCADA networks that relies on OCC and SDN, with high accuracy, differently from the NIDSes proposed until then.

We know that merging SDN in SCADA add new vulnerabilities to these cyber-physical systems, but SDN can be used to map the network behavior of SCADA devices. Unfortunately, there are few IDSes that exploit the network behavior of SCADA, and the systems that use these aspects do not employ the characteristics of SDN to collect more accurate information of the network (*e.g.*, Barbosa (2014) and Almalawi (2014)). Furthermore, the papers that propose SDN for SCADA systems do not present implementations that prove the benefits of SDN applied in SCADA environments. For this reason, on the next sections we: (*i*) list the requirements of NIDSes for SDN-based SCADA systems (Section 4.1); (*ii*) propose a NIDS based on OCC algorithms (Section 4.3) that periodically verifies the SCADA network, through SDN, in order to find anomalous behaviors that differ from the expected SCADA behavior; and (*iii*) we evaluate the prototype of our solution, emulating a large-scale SCADA network topology (Section 5.2).

# 4 ONE-CLASS NIDS FOR SDN-BASED SCADA SYSTEMS

Initially, in Section 4.1, we list the requirements for the proper operation of our solution. Next, in Section 4.2, we present a brief background on the OCC algorithms that we adopted in the proposed NIDS for SDN-based SCADA systems. Finally, in Section 4.3, we introduce our strategy to detect intrusions in SCADA communication networks, detailing the architecture and the components of our NIDS.

## 4.1 NIDS Requirements

In the power sector, NIDSes are responsible for detecting anomalies in the communication networks of SCADA systems (LI et al., 2012). Unfortunately, the number of cyber-threats that exploit the vulnerabilities of cyber-physical systems is increasing, and frequently new types of malware arise, which are more complex and more difficult to detect (such as, Stuxnet) (DELL, 2015). Moreover, the maintainers of SCADA systems do not usually disclose details about detected cyber-attacks. As a result, SCADA attack traces are scarce and not publicly disclosed (IGURE; LAUGHTER; WILLIAMS, 2006). Hence, a NIDS should detect anomalous behaviors known by operators, as well as anomalies that exploit previously unknown vulnerabilities, and consequently unpatched (Zero-Day exploit). Furthermore, an accurate NIDS generates fewer false-alarms during the SCADA network monitoring. A high accuracy avoids unnecessary efforts from the power grid operators and the maintenance staff (BARBOSA, 2014). However, the design of a NIDS must consider the characteristics of a SCADA network to achieve a considerable accuracy.

The sampling period of a SCADA system may vary depending on the environment that is monitored (ALMALAWI, 2014). For example, a SCADA system applied on the context of power distribution uses a sampling period that ranges from 2 to 4 seconds. On the other hand, a SCADA for water distribution systems has a larger sampling period, requesting data every 10 to 15 minutes (HADLEY; HUSTON, 2007). In addition, the data stored in an RTU may be updated independently from the MTU's sampling period. Hence, a NIDS for SCADA systems **must respect the sampling period** of the monitored system (STOUFFER; FALCO; SCARFONE, 2011). This means that, if a SCADA system has a sampling period of 1 second, the NIDS must collect network statistics on the same frequency, or at least close to the SCADA's sampling period. A NIDS for SCADA systems that monitors the network considering the sampling period can minimize or even avoid the incidence of faults in the system. In the context of power grids, a NIDS that attends the time requirements imposed by the environment can avoid possible outages in the power supply (BARBOSA, 2014).

In the power sector, SCADA systems can be used for monitoring small centers of power distribution, as well as large-scale power grids that supply energy for entire countries (CHIKUNI; DONDO, 2007). Thus, there is a wide range of equipment available for sale and protocols

that can be used in SCADA systems. Although there are norms that standardize components and protocols for SCADA systems (such as, IEC-61850), it is still common that manufacturers develop equipment incompatible with some technologies or devices of another manufacturer (FERNANDEZ; LARRONDO-PETRIE, 2010). A NIDS used in this environment **must be scalable**, and it also **must adapt itself to the diversity of available technologies for SCADA systems**. Our solution must be generic to operate independently of the amount of equipment, the protocol used, or the behavior of the network devices.

Usually, a power substation is composed by several transformers, relays, or fuses. In turn, each of these substation equipment is monitored and controlled by tens of sensors and actuators (CHIKUNI; DONDO, 2007). So, a single substation easily can comprise thousands of fields devices directly or indirectly (*via* RTU) connected to the SCADA master station (CAHN et al., 2013). Due to the large number of equipment connected to a SCADA network, each data request can generate bursts of large data amounts (BARBOSA; SADRE; PRAS, 2012). Consequently, these data bursts will generate network statistics for the entire system. Thus, a NIDS for SCADA systems **must constantly manipulate large datasets**. A NIDS that adopts techniques of parallel processing will present advantages in SCADA environments. The parallel processing of information permits the manipulation of large amounts of information generated by data bursts, minimizing possible bottlenecks in the processing of statistics periodically gathered on a SCADA network. In addition, the selection of a traffic classification mechanism is also important because NIDSes that rely on algorithms with fast classification process will obtain advantages in SCADA environments (ALMALAWI, 2014). Table 4.1 lists the requirements presented above in this section.

Table 4.1: NIDS requirements for SCADA systems.

| Requirement | Description |
|---|---|
| **Respect the SCADA's sampling period** | It can minimize or even avoid the incidence of faults in the monitored system |
| **Respect the SCADA's scalability** | It permits that the NIDS operates independently of the amount of equipment |
| **Respect the diversity of techonlogies that may coexist in a SCADA system** | It permits that the NIDS operates independently of the protocol used, or the behavior of the netowrk devices |
| **Manipulate large datasets** | It minimizes possible bottlenecks in the processing of statistics periodically gathered |

## 4.2 OCC Algorithms

Given a SCADA system that controls several power distribution substations, assume it is composed of various equipment that are constantly operating to supply energy for final users.

The SCADA system can possibly comprise thousands of field devices for monitoring the substations of that power grid. Thus, the SCADA's master station periodically receives information about the equipment in operation, such the machinery temperature, or if the components are generating unknown vibrations. In this case, it is relatively easy to gather training information about the expected functioning of the system. But on the other side, collecting data about a system fault or a malware spreading can be expensive, or just impossible. If a system fault could be simulated, there is no guarantee that the faults that possibly will occur will respect the pattern generated by the simulated case. To cope with this problem, OCC solutions are introduced.

We verified in the literature the classification algorithms utilized for detecting anomalies in the network to analyze the advantages and disadvantages of each approach in accordance with the listed requirements in Section 4.1. In our review, the SVM technique stood out in relation to others techniques used for traffic classification. SVMs are among the most popular methods of supervised learning. SVMs are a set of supervised learning methods that analyze datasets and recognize patterns (CORTES; VAPNIK, 1995). They are widely used for text classification, recognizing patterns in images, and network traffic classification. Although SVMs are computationally expensive, **it is possible to build fast classifiers** setting some parameters, respecting the first requirement listed in Table 4.1. Furthermore, SVMs produce very accurate classifiers that are robust to noise (CHERKASSKY; MA, 2004). SVMs also attend the last requirement of NIDSes for SCADA systems, because **their algorithms deal well with large datasets** and in most cases perform better in comparison to other supervised learning methods (CARUANA; NICULESCU-MIZIL, 2006), even in problems that are not linearly separable.

There are SVM adaptations to deal with the OCC problem, such as OCSVM (SCHöLKOPF; SMOLA, 2001) and SVDD (TAX; DUIN, 2004). These variations of SVM specialized in OCC have an extensive documentation available and are widely applied in applications that need to detect novelties. By the characteristics of these algorithms and the requirements of our application, we adopted these SVM-based algorithms, OCSVM and SVDD, in our NIDS. However, to comprehend the operation of SVM-based OCC algorithms it is necessary, firstly, to understand the operation of the traditional binary SVM.

SVMs creates a non-linear decision boundary by projecting the data through a non-linear function $\phi$ to a space with a higher dimension. In other words, the data not linearly separable in an original space $I$ is reprojected in a feature space $F$, where there can be a hyperplane that separates the data according to its respective class. When the hyperplane is projected back in $I$, it assumes the shape of a non-linear curve. The hyperplane of the traditional SVM is represented with the equation $w^T x + b = 0$, where $w \in F$ and $b \in R$. The hyperplane determines the margin between classes. Thus, the instances of class $-1$ will be in one side of the hyperplane, whilst the instances of class $+1$ will be in the opposite side. Furthermore, the hyperplane maximizes the margin between the classes. So, the distance from the nearest instance from each class to the hyperplane is the same.

Slack variables $\xi_i$ are introduced to enable the projection of some instances in the opposite

side to the their class, in order to prevent the SVM classifier from over-fitting with noise data. Slack variables make the optimization restrictions imposed by support vectors more flexible. In addition, the constant $C > 0$ determines the trade-off between maximizing the margin and the amount of training data inside this margin (training errors). The objective function of the SVM classifier is the minimization formulation in Equation 4.1.

$$\min_{w,\ b,\ \xi_i} \frac{\parallel w \parallel^2}{2} + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to :} \tag{4.1}$$

$$\begin{aligned} y_i(w^T \phi(x_i) + b) &\geq 1 - \xi_i \quad \text{for all } i = 1, \ \dots, n \\ \xi_i &\geq 0 \quad \text{for all } i = 1, \ \dots, n \end{aligned}$$

This minimization problem comprises quadratic programming (FRANK; WOLFE, 1956) and is solved by introducing the Lagrangian Function (BREZZI, 1974). The Lagrangian Function involves the restrictions to the objective function associated to the parameters named Lagrange multipliers. Thus, Equation 4.2 presents how is formulated the classification rule for each instance $x$.

$$f(x) = \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b) \tag{4.2}$$

In this case, $\alpha_i$ are Lagrange multipliers. Lagrange multipliers are applied in optimization problems, and they find extremes (maximum and minimum) of a function susceptible to one or more restrictions (BREZZI, 1974). The Lagrangian function must be minimized, and this implies in maximizing $\alpha_i$ and minimizing $w$ and $b$. Each $\alpha_i > 0$ is weighted in the decision function that generates the classifier support vectors.

The function $K(x, x_i) = \phi(x)^T \phi(x_i)$ is named as Kernel Function (BAUDAT; ANOUAR, 2001). The outcome of the decision function only relies on the dot-product of the vectors in $F$. This is not necessary to perform an explicit projection of $F$. In this case, a kernel function can be used to project $F$ and to obtain the same results (kernel trick) because kernel functions permit the representation of abstract spaces. The kernel trick technique allows the classification of data that originally is non-linearly separable (CHERKASSKY; MA, 2004). The feature space $F$ may comprise an unlimited number of dimensions, making the hyperplane building process that separates the data complex (SCHöLKOPF; SMOLA, 2001). The most used kernel functions are linear, polynomial, sigmoidal, and the Radial Base Function (RBF). The formulation of these kernel functions are presented in Table 4.2.

However, our application relies on the OCC paradigm to detect anomalies in SCADA networks. Thence, we are interested on OCC versions of SVMs. As said before, there are two main approaches of SVM-based OCC algorithms, OCSVM and SVDD. These algorithms are

discussed below, in sections 4.2.1 and 4.2.2 respectively.

### 4.2.1 One-Class Support Vector Machine (OCSVM)

OCSVM is a supervised machine learning algorithm based on SVM presented by Schölkopf *et al.* (SCHöLKOPF; SMOLA, 2001). OCSVM is the most popular technique for OCC and is indicated for problems that involve anomaly detection. Basically, OCSVM infers the properties of normal cases and from these properties can predict which examples differ from the normal examples (SCHöLKOPF; SMOLA, 2001). This OCC algorithm resembles the traditional two-class SVM, where the training set is composed of two groups, one positive and one negative. However, OCSVM is an adaptation of the traditional SVM, where the training data contains only members of the target class, and the origin in $F$ is characterized as the only member of the outlier class. OCSVM uses a homogeneous set of instances in its training phase, and separates the training data from the origin (in the feature space $F$), maximizing the distance from the hyperplane to the origin.

The OCSVM algorithm builds a binary function that captures regions in the original space $I$ where there are density of training data. This binary function recognizes an small region as the target class (mapped through the training data coordinates) and the remaining of $I$ as the outlier class. Thus, when a validation instance $z$ is classified, if $f(z) < 0$, then $z$ is labeled as an anomaly (outlier), otherwise it is labeled normal (target class). The OCSVM minimization function, presented in Equation 4.3, has similarities in relation to the SVM traditional function:

$$\min_{w,\ \xi_i,\ \rho} \frac{1}{2} \parallel w \parallel^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho$$

$$\text{subject to :} \tag{4.3}$$

$$\begin{aligned} (w \cdot \phi(x_i)) &\geq \rho - \xi_i \quad \text{for all } i = 1, \ldots, n \\ \xi_i &\geq 0 \quad \text{for all } i = 1, \ldots, n \end{aligned}$$

Table 4.2: Most popular Kernel Functions.

| Kernel | Function |
|---|---|
| **Linear** | $K(x, x_i) = (x \cdot x_i)$ |
| **Polynomial** | $K(x, x_i) = (\delta(x \cdot x_i) + \kappa)^2$ |
| **Sigmoidal** | $K(x, x_i) = \tanh(\delta(x \cdot x_i) + \kappa)$ |
| **RBF** | $K(x, x_i) = \exp\left( - \frac{\parallel x - x_i \parallel^2}{2\sigma^2} \right)$ |

However, instead of using the penalty parameter $C$ to smooth the training errors, the OCSVM uses the $\nu$ parameter to characterize the solution. The $\nu$ parameter is responsible for: (*i*) defining an upper bound of outliers in the training set; and, (*ii*) defining a lower bound on the number of instances used as support vectors.

As the traditional SVM for binary classification, the OCSVM algorithm also uses kernel functions to classify data that is non-linearly separable. OCSVM can use traditional kernel functions, or customized kernels defined by the user. Thus, as showed in Equation 4.4, by using kernel functions and Lagrange techniques, the OCSVM decision function becomes:

$$f(x) = \text{sgn}((w \cdot \phi(x_i)) - \rho) = \text{sgn}(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho) \tag{4.4}$$

This function builds a hyperplane characterized by the variables $w$ and $p$. This function maximizes the distance from the origin in $F$ and separates the instances from the origin. However, the user needs to experiment which kernel fits better to the data that will be classified in order to achieve more accurate results.

### 4.2.2 Support Vector Data Description (SVDD)

Also known as Support Vector Domain Description, the SVDD algorithm was introduced by Tax and Duin (TAX; DUIN, 1999) (TAX; DUIN, 2004) and is another type of SVM-based OCC algorithm. SVDD is a useful method for novelty detection and has been applied to a variety of applications that need to monitor the rise of novelties. SVDD adopts a different approach to separate data in a feature space. Instead of building a hyperplane, this algorithm uses the training set to define a hypersphere with minimum radius, which is used for binary classification of instances of a validation set. In other words, the purpose of the SVDD technique is not to find an optimal separating hyperplane, but a spherically shaped boundary around the dataset with minimal volume containing all target data (BENKEDJOUH et al., 2012).

The SVDD hypersphere is characterized by a center **a** and a radius ($R > 0$) as distance from the center to the boundary, of which the volume $R^2$ will be minimized. The center **a** of the hypersphere is a linear combination of the support vectors, that are the training instances for which the Lagrange multiplier is not zero. As well as the formulation of the traditional two-class SVM, it can be required that the hypersphere margins are softened. For this, slack variables $\xi_i$ and the penalty parameter $C$ are also introduced in SVDD. The Equation 4.5 describes the SVDD minimization problem.

$$\min_{R,\, \mathbf{a}} R^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to :} \tag{4.5}$$

$$\| x_i - \mathbf{a} \|^2 \leq R^2 + \xi_i \quad \text{for all } i = 1, \ldots, n$$
$$\xi_i \geq 0 \quad \text{for all } i = 1, \ldots, n$$

A new instance $z$ can be validated as a member of the target class (or an outlier) after introducing Lagrange multipliers $\alpha_i$ in this formulation. Thus, the SVDD hypersphere is modeled to involve the majority of training samples. In the validation stage, new samples that are not inside the hypersphere area are classified as novelties, whereas samples that are classified inside the hypersphere are considered normal samples. That is, as depicted in Equation 4.6 a new sample is considered member of the target class if the distance to the center is smaller than or equal to the hypersphere radius, by using the RBF Kernel as a distance function between two data points in $F$.

$$\| z - \mathbf{x} \|^2 = \sum_{i=1}^{n} \alpha_i \exp\left(\frac{- \| z - \mathbf{x}_i \|^2}{\sigma^2}\right) \geq -R^2/2 + C_R \tag{4.6}$$

## 4.3 Architecture Overview

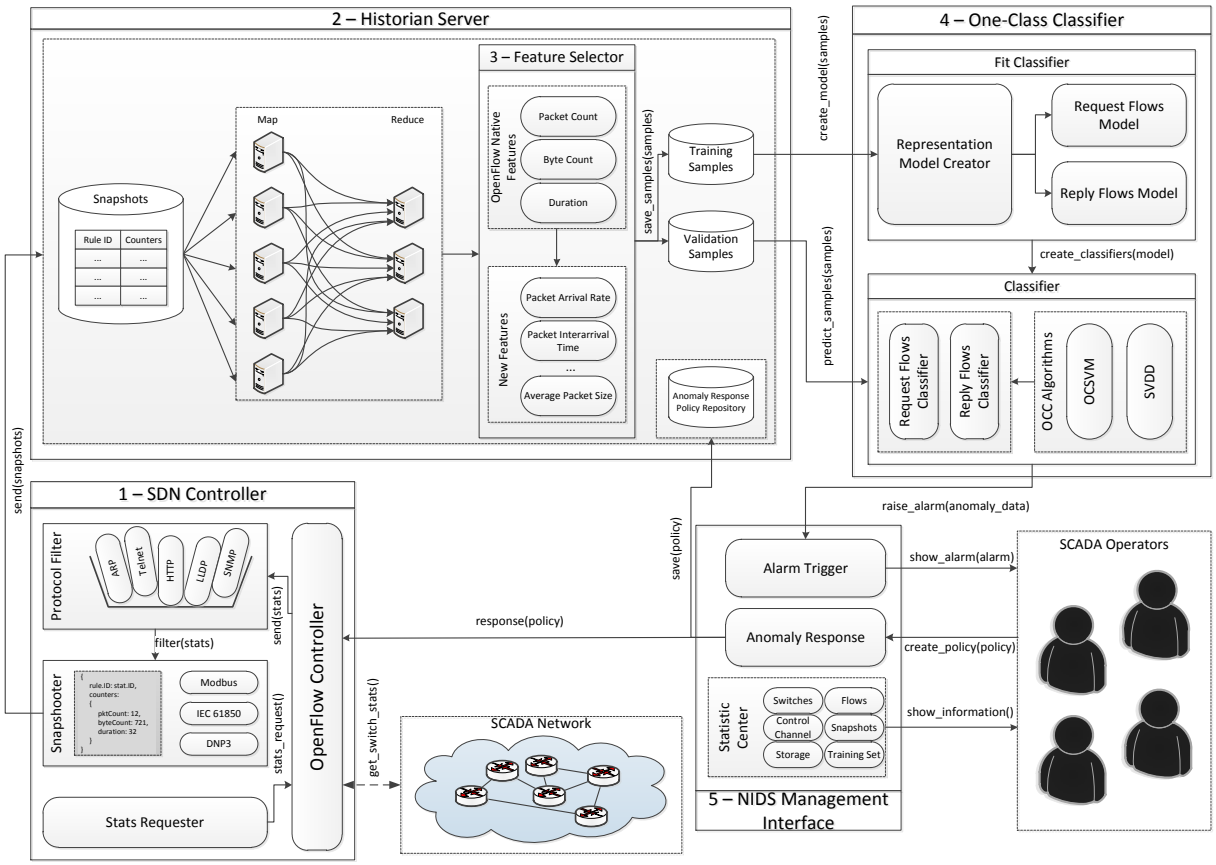
Differently from previous approaches that propose IDSes for traditional SCADA systems (BARBOSA, 2014)(ALMALAWI, 2014), we present a NIDS designed and developed for SDN-based SCADA systems. Our approach takes advantage from the characteristics of SCADA networks and the benefits that SDN will bring for these systems to accomplish *novelty* detection. Considering the periodicity and the behavior of SCADA networks (BARBOSA, 2014), novelties (*i.e.*, unexpected behaviors) in the communication environment of SCADA systems may indicate anomalous behaviors. We rely on the assumption that a NIDS needs to know the normal system behavior to predict anomalous and dangerous behaviors to the services provided by the monitored system. So, as our classifier knows the expected behavior of a SCADA network, any data that represents novelty in relation to the classifier model can be reported to the operator as a network anomaly. These network anomalies may represent the beginning of cyber-attacks, the propagation of malwares, or even the misconfiguration of field devices.

OpenFlow facilitates anomaly mitigation, enabling the redirection of malicious network traffic, or even dropping the packets of intruders. To route packets, OpenFlow installs rules in each network switch. Each rule may have unique information about a communication flow, such as: (*i*) MAC and IP address of source and destination devices; (*ii*) packet and byte counters; (*iii*) rule duration (in seconds and nanoseconds); and, (*iv*) the switch in which the rule is installed. The proposal uses OpenFlow for periodically extracting statistics from the SCADA network. Thus, our strategy can continuously gather statistics on the same frequency of the SCADA sampling period. Our NIDS generates samples from statistics gathered on the SCADA network. In turn, samples serve as basis to build the component that predicts the network behavior of SCADA devices. By default, our proposed solution provides an ID for each generated sample, however, the composition of IDs can be defined by the SCADA system operator. The Open-

Flow native counters also compose a sample and it is possible to use other features extracted from the native ones. The selection of an optimal set of features can increase the classifier accuracy (SILVA et al., 2015a), decreasing the rate of false-positive alarms generated during the network monitoring.

Every component of the proposed solution has been designed to attend the requirements listed on Section 4.1. We have planned our architecture **to be scalable**, permitting the detection of anomalies in SCADA systems responsible for monitoring small regions, as well as large-scale SCADA systems with thousands of equipment. In addition, the architecture contemplates the adoption of mechanisms that enable the parallel and distributed processing of samples, such as the MapReduce programming model (DEAN; GHEMAWAT, 2008). MapReduce enables our NIDS **to promptly process large amounts of data** generated by SCADA equipment, permitting, for example, the NIDS **to operate according to the sampling period** requirements of SCADA for power distribution environments. Figure 4.1 presents an overview of the proposed NIDS architecture. Our NIDS comprises five components that intercommunicate to monitor the network and to report anomalous behaviors in SCADA systems. A more detailed description of these components is presented below:

Figure 4.1: Architecture overview of the proposed NIDS for SDN-based SCADA systems.




Source: by author (2016).

### 4.3.1 SDN Controller

This component acts directly on the SCADA network. SDN Controller is responsible for monitoring and for applying routing strategies to the SCADA network switches. This component was developed to directly attend the requirements 1 and 3 of Table 4.1 because it relies on parameters that define the periodicity of statistics gathering, and which protocols will be monitored by our NIDS. SDN controller is composed of *Stats Requester*, *OpenFlow Controller*, *Protocol Filter*, and *Snapshooter*:

- *Stats Requester*: Stats Requester is responsible for: (*i*) creating request messages of flow statistics; and, (*ii*) for controlling the periodicity that these messages are sent to the network switches. SCADA systems can use specific protocols for each substation. For example, a modern substation can use IEC-61850 to exchange messages within its area, whilst a legacy substation of the same SCADA can use the traditional Modbus serial protocol. In addition, each field device can report data in different frequencies. Thus, Stats Requester receives as parameter, values that define the frequency at which statistics requests are sent to the network. The SCADA operator must know the system characteristics that will be monitored to define values that are compatible with the periodicity of specific parts of the SCADA system. Our approach permits the SCADA operator to choose a monitoring strategy that, for example, is more frequent on critical network points and sparser on non-critical substations;
- *OpenFlow Controller*: As mentioned before, OpenFlow is currently the most important protocol for SDN implementation (MCKEOWN et al., 2008). OpenFlow Controller applies routing strategies on the SCADA network. This module acts as intermediary for other components, interfacing the NIDS with the SCADA network. The OpenFlow Controller module receives the Stats Requester's requisitions and forwards these messages to the SCADA switches. However, in this module, the SCADA operator must define which switches will be monitored on the network. Moreover, the SCADA operator needs to define the sending frequency of statistics requests for each switch on the topology. So, our NIDS can monitor the network in its totality in accordance with the different polling times used by the SCADA system. In addition, OpenFlow Controller receives the statistics of SCADA switches and forwards this data to the module responsible for handling this information. Finally, OpenFlow Controller also is responsible for enforcing anomaly response policies on the SCADA network, such as to limit the traffic to control network protocols, or to block unexpected protocols to exchange messages in the SCADA environment, avoiding the propagation of malware or the the occurrence of exploits in the network;
- *Protocol Filter*: When actioned, Protocol Filter blocks statistics regarding non-specific SCADA protocols. So, this module allows the monitoring of only previously defined

types of flows and generates statistics for the NIDS. Protocol Filter offers a list of protocols that can be observed on a SCADA network (such as, LLDP, ARP, and ICMP, and specific SCADA protocols, such as Modbus, DNP3, and IEC-61850). As well as the Stats Requester module, Protocol Filter enables the manual selection of which protocols will be monitored on the SCADA network. Thus, the SCADA operator can decrease the amount of information processed by our NIDS;

- *Snapshooter*: In turn, the Snapshooter module organizes the information received from Protocol Filter into snapshots. The snapshots generated by this module are sent and stored on Historian Server. In our approach, a snapshot is composed of an ID and the OpenFlow native counters. Our NIDS offers a standard ID (ruleID) that is showed below:

$$ruleID = (srcip, dstip, srcport, dstport, protocol, switch)$$

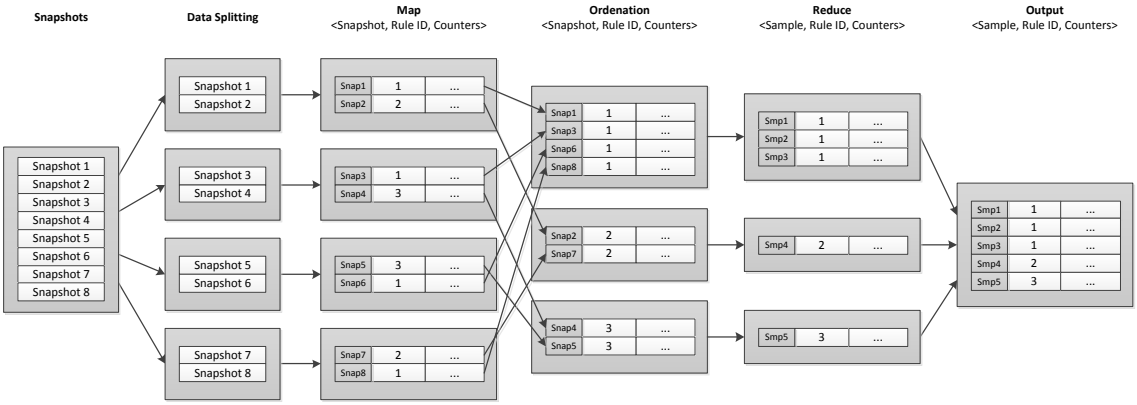
However, our proposal permits the SCADA operator to define which information will be used to compose a ruleID.

### 4.3.2 Historian Server

Historian Server is a component that is typically present in the master station of several traditional SCADA systems (ZHU; JOSEPH; SASTRY, 2011). This server stores logs about devices, events, and alarms which can be used to populate graphic trends in HMIs (STOUFFER; FALCO; SCARFONE, 2011). We extended the Historian Server component to store the snapshots extracted from the network by SDN Controller. As large-scale SCADA systems can interconnect thousands of devices that constantly generate large amounts of data, Historian Server needs to be able to store and process larger datasets. For this reason, in order to respect the fourth requirement listed in Table 4.1 and to allow large-scale data processing, the proposed NIDS relies on techniques of distributed and parallel processing, such as the MapReduce programming model (DEAN; GHEMAWAT, 2008), to reduce possible bottlenecks in SCADA servers.

In this context, we designed a scalable version of Historian Server, using a processing queue and MapReduce. The processing queue is periodically verified if there are snapshots that can be processed via MapReduce. As presented in Figure 4.2, MapReduce permits the processing of data by multiple processors, named workers. Initially, the data stored on the Snapshots database is split (Data Splitting) into smaller parts that will be processed separately by workers specialized in data mapping. Each worker contains the Mapper code, a function that analyzes the snapshots and maps this information into tuples $< RuleID, Counters >$ (Mapping process). The pseudocode of the Mapper function used in the mapping process is presented in Algorithm 1. The snapshots processed by the Map Workers are ordered and grouped (Ordenation) according to their identifiers, and they are sent to the workers specialized in reduce this information. In turn, the Reduce workers are responsible for reducing the mapped snapshots

Figure 4.2: Diagram of the MapReduce algorithm of the proposed NIDS.




Source: by author (2016).

into native samples (Reduction process). Native samples are generated from the subtraction of snapshots with the same IDs but collected on distinct times, considering only the OpenFlow native counters ($Pkt$ = Packet Count, $Byt$ = Byte Count, and $Dur$ = Duration). An example of the conversion process of two snapshots ($S_t$ and $S_{t-1}$) in one native sample ($nativeSample_x$) is demonstrated below:

$$nativeSample_x = \begin{cases} Pkt(S_t) - Pkt(S_{t-1}) \\ Byt(S_t) - Byt(S_{t-1}) \\ Dur(S_t) - Dur(S_{t-1}) \end{cases}$$

In addition, the Reduce Workers emit the output of the MapReduce process to the next component, Feature Selector. The pseudocode of the reduction process is presented in Algorithm 2 and the Feature Selector component is detailed in Section 4.3.3. Furthermore, Historian Server comprises the *Anomaly Response Policy Repository*, which is a component that stores strategies for anomaly mitigation. Anomaly response policies will be discussed later on in this chapter, when the *Anomaly Response* component is presented. Thus, our NIDS can proactively act on the network without direct intervention of an operator, for example: (*i*) redirecting an anomalous flow to a HoneyPot device; (*ii*) reducing the priority of a harmful flow to the system; or (*iii*) dropping undesirable packets.

### 4.3.3 Feature Selector

The OpenFlow protocol only provides native flow features, namely packet count, byte count, and flow duration. These features may not be sufficient or adequate to describe the nature of a specific traffic profile, as the profile of the SCADA network (BARBOSA, 2014). Using

**Algoritmo 1** Mapper PseudoCode

```
1:  function MAPPER(snapshots)
2:
3:      for each snapshot in snapshots do
4:          ruleID ← getID(snapshot)
5:          counters ← getCounters(snapshot)
6:
7:          if (!isIDMapped(ruleID)) then
8:              save(ruleID)
9:          end if
10:
11:         keyRuleID.add(counters)
12:
13:     end for
14:
15: end function
```

**Algoritmo 2** Reducer PseudoCode

```
1:  function REDUCER(ruleIDs)
2:
3:      for each ruleIDMapped in ruleIDs do
4:          counters ← getMappedCounters(ruleIDMapped)
5:
6:          if (getCountersAmount(counters) < 2) then
7:              continue
8:          end if
9:
10:         if (!isIDMapped(ruleIDMapped)) then
11:             save(ruleIDMapped)
12:         end if
13:
14:         for i ← 1 until getCountersAmount(counters) with step ← 1 do
15:             nativeFeatures ← extractNativeFeatures(counters(i), counters(i - 1))
16:             mappedKeyRuleID.add(nativeFeatures)
17:         end for
18:
19:     end for
20:
21: end function
```

appropriate features to describe traffic behavior may increase the accuracy of our NIDS. Thus, the Feature Selector component (SILVA et al., 2015a) was proposed to offer an extensive set of features with 33 entries extracted from the OpenFlow native counters, *e.g.* packet inter-arrival-time, packets per second, mean packet length, and so forth. Besides, this component uses feature selection techniques, such as Principal Component Analysis (PCA) (HOFFMANN, 2007) and Genetic Algorithm (GA) (WHITLEY, 1994), to determine the optimal set of features for traffic

classification.

Feature Selector was strategically placed inside the Historian Server component, more precisely at the end of the conversion process of snapshots into native samples, in order to avoid unnecessary computational processing. Feature Selector directly receives each native sample (formatted with its ID and counters) and transforms them into more detailed samples, as showed in the example below:

$$\langle (ruleID) : (pkt\_count, byte\_count, duration) \rangle$$
$$\downarrow$$
$$\langle (ruleID) : (pkt\_count, byte\_count, duration, arrival\_rate, average\_packet\_size...) \rangle$$

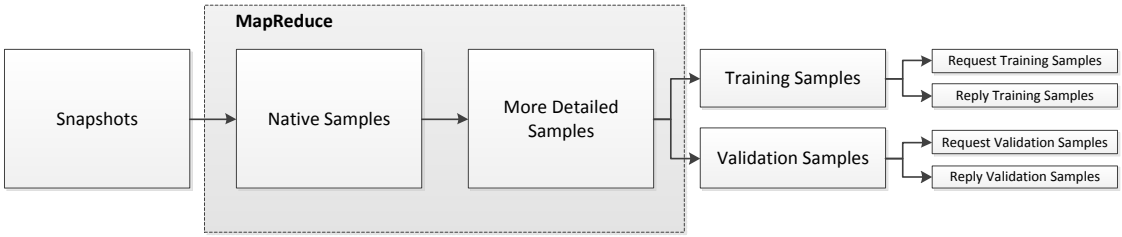
Further, this component splits the more detailed samples into two types, *Training Samples* and *Validation Samples*. Both types of samples are stored on the Historian Server. On the one hand, Training Samples are used for fitting the NIDS. Training Samples are fundamental to build the representation models responsible for classifying the SCADA network traffic in real-time. On the other hand, Validation Samples are generated during the network monitoring. Validation Samples are posteriorly analyzed by the One-Class Classifier component.

In addition to Feature Selector, we adopted another strategy to enhance the accuracy of our NIDS. In the case of SCADA networks, the direction of a communication flow (from master to slave device, or from slave to master device) directly influences on building a representation model. For example, messages sent by the master device maintain a standard profile, whilst packets sent by a slave (*e.g.*, RTU or field devices) constantly vary because the reported data from sensors changes among the measurements. Thus, in order to increase the NIDS classification accuracy, we also split samples into two classes: request and reply samples. Request samples have the master device as source device, whilst reply samples have an RTU or a field device as source. Note that, the activation of the Feature Selector component is optional because there are cases in which the SCADA operator needs a faster traffic classification (but with less descriptive samples), near to the SCADA sampling time. Thus, the NIDS will use the native samples to describe the profile of the SCADA traffic. A summary about the life-cycle of how the information is handled on the Historian Server and the Feature Selector is showed in Figure 4.3.

### 4.3.4 One-Class Classifier

The One-Class Classifier component is the central element in the proposed architecture. This component analyzes the samples stored in Historian Server to find anomalous behaviors in the SCADA network. The functions of this component are: (*i*) to analyze the samples stored in Historian Server to find anomalous behavior in the SCADA traffic; and, (*ii*) to emit alarms to the next component about possible threats detected in the network. This component is divided

Figure 4.3: The information life-cycle inside the Historian Server and the Feature Selector.




Source: by author (2016).

into two modules, one for training the classifier (Fit Classifier), and other for classifying the SCADA network traffic (Classifier). The description of these two modules is presented below:

- *Fit Classifier*: The Fit Classifier module is responsible for the training step of the proposed NIDS. In other words, the training samples of request and reply stored in Historian Server are loaded for building representation models to each communication flow direction on the system. The Fit Classifier uses parameters that define how the training stage will be accomplished. To the NIDS can actuate properly, the training stage must occur before the traffic validation stage. However, our approach permits additional configurations that assist in fitting the classifier. For example, it is possible to execute new trainings during the monitoring of the SCADA network, and to define the periodicity in which our NIDS will be trained again. This functionality enables the proposed NIDS to learn new trends on the SCADA network, such as possible new commands (*e.g.*, to define the rotation of a turbine) and routing strategies inserted on the SCADA system (such as, multicast diffusion of messages to the slave devices), or even an increase in the amount of data reported from RTUs caused by the elevation in the energy demand. Fit Classifier provides to the SCADA operator two options of training: (*i*) from the normal operation of the SCADA system: where the system operator must define the time that will be used for gathering statistics about the network normal functioning. In this case, the classification of new traffic samples (validation stage) on the SCADA network is disabled during the training stage; or, (*ii*) from trace files: this option permits the training of the NIDS from stored training files. Thus, it is possible to save specific training files for sporadic events on the network, such as cases where the resource demand is larger in a set of substations at a certain time, or in cases in which the SCADA sampling time is modified to cover periods of maintenance of equipment or substations;
- *Classifier*: This module loads the representation models previously generated on the Fit Classifier module to build the traffic classifiers, Request Classifier and Reply Classifier, used to detect disturbances on the network considering the direction of a network flow. Classifier is a reconfigurable module that periodically loads the validation samples stored

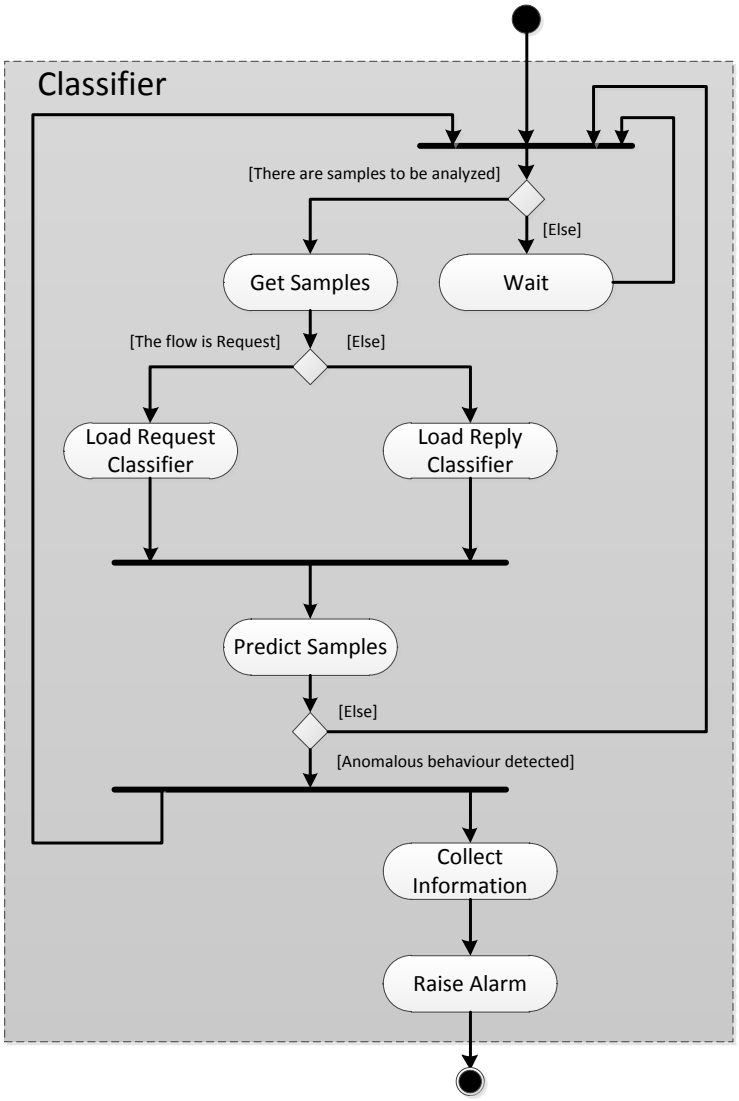
in Historian Server. This module receives parameters through the NIDS Management Interface component. The Classifier's parameters permit the mechanism to operate in accordance with the SCADA network behavior, considering the network periodicity or the periodicity of specific parts of the network. In this context, it is important to note that a more constant verification will generate smaller data bursts in smaller time-cycles, whilst sparser verification, in turn, will generate larger data bursts with larger periodicity. By default, the proposed NIDS offers two options of OCC algorithms, OCSVM and SVDD. However, the solution was conceived to permit the SCADA operator to install new OCC algorithms. In this case, drivers recognize the algorithms available on Classifier and offer the required information (*i.e.*, parameters) to the NIDS Management Interface. In addition, Classifier enables the combination of OCC algorithms to enhance the accuracy of the proposed NIDS. Thus, the network operator can choose the more accurate solution to detect anomalies in their SCADA environment. Finally, Classifier will send an alarm to the NIDS Management Interface if any sample indicates an inconsistent state in the SCADA system, notifying when and where the network anomaly occurred. Figure 4.4 presents an operation diagram of the Classifier module, illustrating when a set of samples is received until the raise of an alarm to the SCADA operators.

### 4.3.5 NIDS Management Interface

The NIDS Management Interface is the component responsible for providing information about the NIDS to the SCADA operators. This management interface also permits that operators send instructions to the NIDS. More precisely, the NIDS Management Interface enables the SCADA operators:

- **To be notified through alarms about possible disturbances on the network:** The anomalous behaviors detected by the One-Class Classifier are directly forwarded to the NIDS Management Interface. Thus, the Alarm Trigger module, that is responsible for generating alarms to the SCADA operators, retrieves the information contained in the validation samples that were classified as anomaly, such as the DPID of the switch that originated the information, the IP address of the devices that are involved in the message exchange, the moment when the information was collected and classified by the NIDS, and the direction in which the message was forwarded. With this set of information, the NIDS can inform the SCADA operators where and when the network anomaly occurred on the power grid, permitting the maintenance staff prepare a solution to solve this possible disturbance;
- **To build anomaly responses:** The NIDS Management Interface also offers a module to build anomaly response policies against eventual disturbances detected by our solution, the Anomaly Response module. This module permits the SCADA operators to create

Figure 4.4: Diagram of the Classifier module.

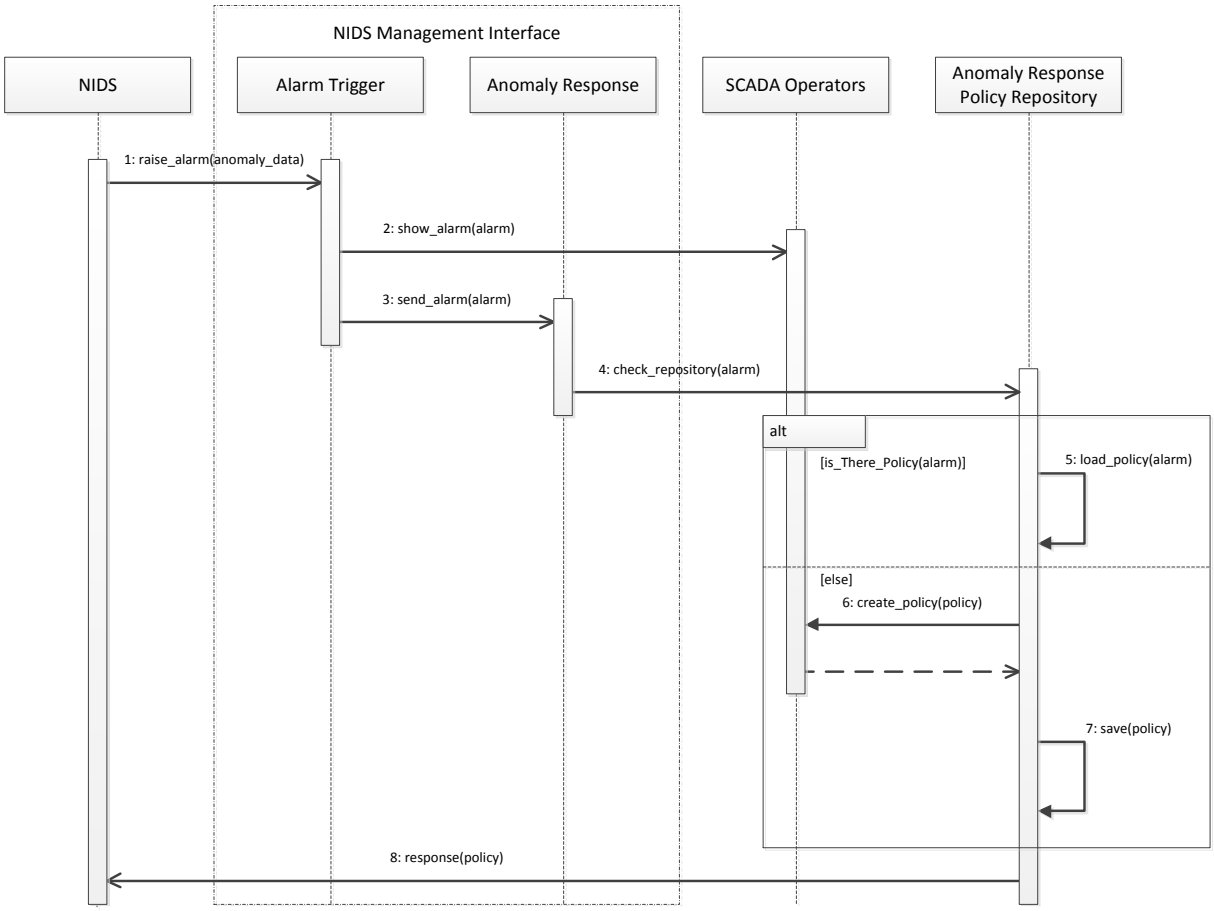



Source: by author (2016).

OpenFlow rules that minimize or mitigate the detected anomalous behaviors. So, it is possible to create an anomaly response that, for example, redirects a DDoS attack to a HoneyPot device, forwards an unknown flow to middleboxes that will analyze the unknown packets, and directly blocks anomalous traffic, avoiding the malicious instructions (such as, restart or shutdown commands) to be sent to RTUs or field devices. The SCADA operator can create an anomaly response immediately after the occurrence of a network disturbance in accordance to the event characteristics. In addition, the proposed NIDS also allows the anomaly responses to be stored on the Anomaly Response Policy Repository, that is situated in Historian Server. Thus, the SCADA operators can define if our NIDS can reuse the stored mitigation strategies and proactively actuate in the system re-

covery into a consistent state. Figure 4.5 presents the sequence diagram of the anomaly response mechanism of our NIDS;

Figure 4.5: Sequence diagram of the anomaly response mechanism.




Source: by author (2016).

- **To configure components:** The NIDS Management Interface permits the configuration of essential functionalities of the NIDS components, such as: (*i*) the periodicity that the SDN Controller will collect network statistics, and if the NIDS will use different sampling times in specific point of the SCADA system; (*ii*) which communication protocols will be monitored by our NIDS; (*iii*) how the IDs that identifies the snapshots will be composed and samples stored in Historian Server; (*iv*) if the native samples will be used in the network monitoring, or if the Feature Selector will create more detailed samples, and which features will be used for characterizing the SCADA traffic; (*v*) as well as the frequency that the NIDS will be trained, the periodicity that the One-Class Classifier will classify the validation samples; and, (*vi*) which algorithms will be used and how they will be configured for identifying anomalies;
- **To analyze information about the NIDS operation:** Finally, the NIDS Management Interface comprises the Statistic Center module. Statistic Center presents information about the operation of the SCADA network and the proposed NIDS. Thus, the operator

can visualize information, such as: (*i*) the traffic on the OpenFlow control channel; (*ii*) the amount of active flows in each switch of the SCADA system and on the network as a whole; (*iii*) the storage capacity of the Historian Server, as well as the amount of stored snapshots and samples; (*iv*) the training set size, and the amount validation samples that are waiting to be analyzed; and, (*v*) which features are used to describe the SCADA traffic.

# 5 PROTOTYPE AND EXPERIMENTAL EVALUATION

In this chapter, we describe a proof-of-concept prototype, the experimental setup used for the evaluation of our implementation, as well as the test scenarios used to simulate a SCADA environment. We also discuss the experimental results that validate our prototype and verify the accuracy of the classification techniques.

## 5.1 Prototype

In the following we describe which were the technologies adopted to develop a proof-of-concept prototype of our solution. To validate the proposed NIDS, we extended or implemented the following components: SDN Controller, Historian Server, and One-Class Classifier. The implementation choices are detailed below:

- **SCADA Network Emulator:** Given the importance, responsibility, and nature of SCADA systems for power grids, it is not feasible to conduct security experiments in real cyber-physical environments (FRIEDBERG; MCLAUGHLIN; SMITH, 2015). For this reason, we used the Mininet Emulator[1] version 2.2.1 to create our evaluation scenario. Mininet is an open source project that was developed in order to offer to researchers a reliable simulation environment for implementing SDN solutions. Mininet emulates a complete network on a single machine, comprising hosts, links, and switches. Each Mininet host has its own private network interface and can only see its own processes. In addition, Mininet emulates OpenFlow switches by default. Mininet is a widely accepted and recognized tool. Furthermore, the Mininet emulator offered the ideal toolset to simulate our SDN-based SCADA system;
- **SDN Controller:** As SDN controller, we chose POX[2] version 0.20. POX is defined as an open source networking software platform. POX permits developers to develop new POX components that implement network functionalities. Thus, POX is very useful for writing networking software in general. In addition, POX is a popular tool for teaching about and researching SDN and network application programming. POX offers programming interfaces in Python, and it started as an OpenFlow Controller, however now, POX also functions as an OpenFlow switch. POX officially supports Windows, Mac OS, and Linux. In our prototype we, used the *l2_learning.py* component to routing packets in the SCADA system. In addition, we used POX timer events to monitor the SCADA communication network, since the POX platform provides a framework for communicating with switches that use the OpenFlow 1.0. We are aware that there are newer versions of the OpenFlow protocol (such as version 1.5), however version 1.0 is currently the stable version of the

[1] http://mininet.org/
[2] http://www.noxrepo.org/pox

protocol[3]. Obviously, the use of newer versions can optimize issues such as processing, but we found in version 1.0 all required support to our solution;

- **Historian Server:** We implemented the Historian Server component on the Apache Cassandra[4] database version 2.2 (The stable version). Apache Cassandra is an open source distributed database management system designed to handle large amounts of structured data across many nodes. Cassandra relies on the NoSQL (Non Structured Query Language) paradigm. NoSQL databases do not use relational models and permit the implementation of scalable, resilient, and distributed high-performance solutions. In addition, Cassandra is a column-oriented database and has a flexible data model. Thus, different from a table in a traditional relational database, distinct rows in the same table do not have to share the same set of columns. The flexible data model of Cassandra enables the utilization of personalized RuleIDs by our solution. In addition, this database promotes design scalability and allows distributing system tasks to multiple clusters, decreasing possible processing bottlenecks. The Apache Cassandra database supports several language drivers to develop solutions, such as Python, C#, Java, and other programming languages. It is important to note that we did not use a MapReduce Framework in our prototype. Our prototype relies on an off-line classification process. However, we intend to improve the prototype, adding a MapReduce framework and turning on-line the classification process;
- **OCC algorithms implementation:** In order to obtain a reliable implementation of the OCC algorithms used in our NIDS, we adopted the LIBSVM[5] library in its version 3.18. LIBSVM is an open source ML library that offers a simple and efficient implementation of several SVM-based algorithms for binary, multiclass, and OCC problems. LIBSVM is written in C++ and offers interfaces to develop solutions in other languages, such as Python, Java, Ruby, and so forth. The algorithms available in LIBSVM typically requires two steps: (*i*) training a dataset to obtain a classification model; and (*ii*), using the classification model to predict information of a validation dataset. LIBSVM natively offers the OCSVM algorithm, however the version 3.18, specially, offers a plug-in that enables the utilization of the SVDD algorithm. It is important to note that both OCC algorithms used in our experiments, OCSVM and SVDD, were configured with their default parameters, using the RBF Kernel.

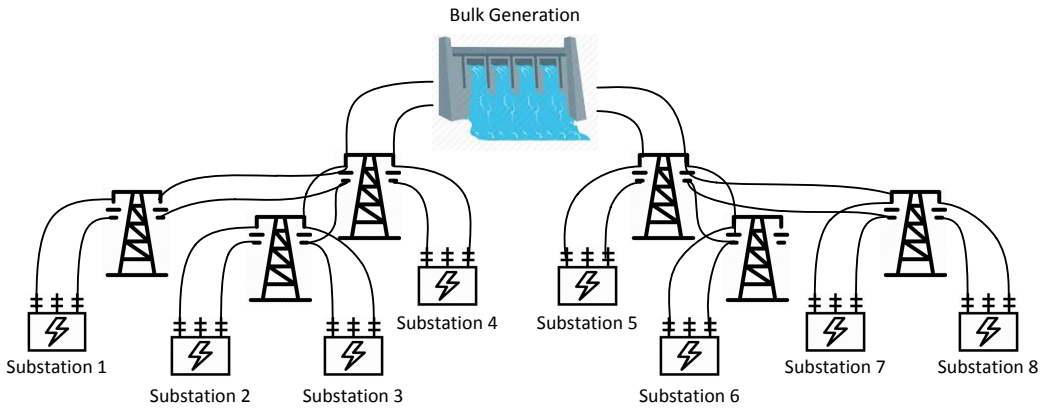
### 5.1.1 Evaluation Scenario

Our evaluation scenario (*e.g.*, the number of components, protocols, topology, commands, and the SCADA sampling time) was based on documents that detail characteristics of large-

[3]http://archive.openflow.org/wk/index.php/OpenFlow_Wiki
[4]http://cassandra.apache.org/
[5]https://www.csie.ntu.edu.tw/∼cjlin/libsvm/

scale SCADA systems (BAILEY; WRIGHT, 2003; BOYER, 2009; STOUFFER; FALCO; SCARFONE, 2011) and SCADA systems for power grids (THOMAS; MCDONALD, 2015; STOUFFER; FALCO; SCARFONE, 2011). Thus, based on these documents, we defined an environment that simulates a large-scale SDN-based SCADA system for power grids. In this context, we assume that the SDN-based SCADA system is maintained by a particular power grid company. This company controls a hydro-power plant, power transmission lines, and eight distribution substations. The power grid simulated in our evaluation scenario is responsible for supplying energy to eight different regions and is showed in Figure 5.1.

Figure 5.1: General overview of the power grid simulated in our experiments.




Source: by author (2016).

The simulated SDN-based SCADA system is composed by several equipment. More specifically, our SCADA comprises: 25 OpenFlow switches that interconnect the SCADA devices; 1 MTU to send commands and control the SCADA components; 4 subMTUs to receive MTU's commands and forward them to the subordinated devices, such as RTUs and IEDs. In addition, subMTUs also report their stored data to the MTU; 8 RTUs that control field devices, and centralize the data gathered in the distribution substation, forwarding this information to one subMTU; 6,000 traditional field devices reporting data about the substation equipment. These field devices were divided in the 8 distribution substations of the power company, totalizing 750 for each substation. Thus, the field devices of each substation are directly connected to one RTU; and, 800 independent field devices, also known as Intelligent Electronic Devices (IEDs). IEDs were placed in our SDN-based SCADA system for controlling the voltage in the transmission lines of the power grid. Different from the traditional field devices that are abstractions in our evaluation scenario, each IED was simulated independently. In addition, as well as RTUs, each IED directly exchanges information with one subMTU. In a nutshell, in our evaluation scenario the MTU requests data from its sub-MTUs, which consequently request data from their subordinated devices (RTUs or IEDs). Table 5.1 summarizes the set of SCADA devices used in our evaluation scenario.

Due to the large number of equipment, we adopted a network topology for large-scale

Table 5.1: Overview of our evaluation scenario.

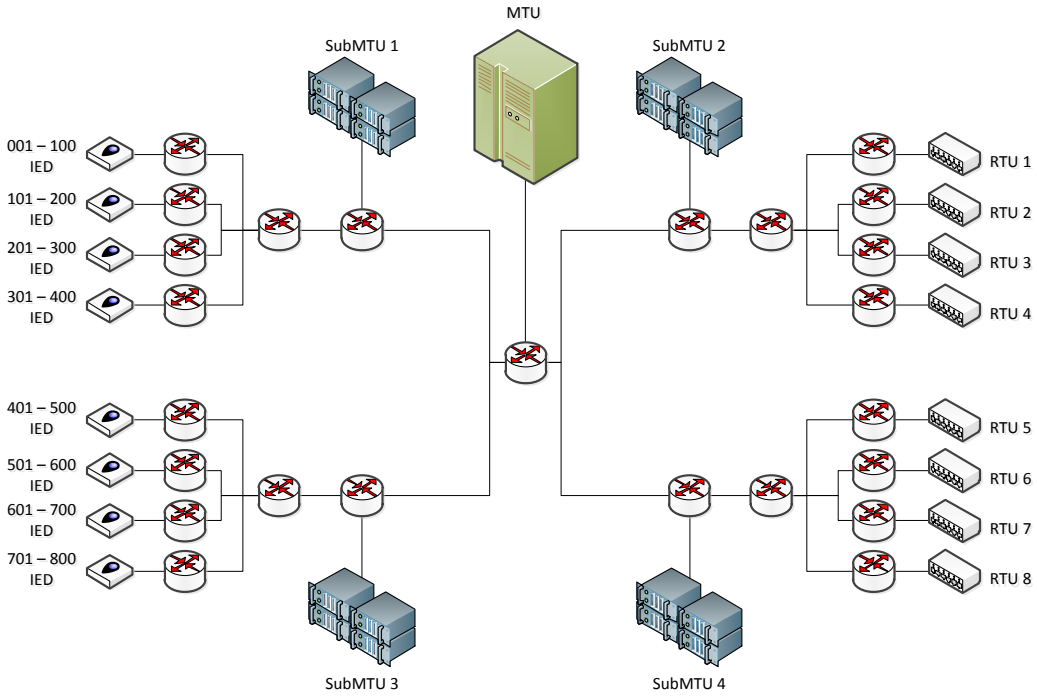
| **Device** | **Quantity** |
|---|---|
| MTU | 1 |
| subMTU | 4 |
| RTU | 8 |
| Field Device | 6,000 |
| IED | 800 |
| OpenFlow Switch | 25 |
| ***Total*** | 6,838 |

SCADA systems presented by Stouffer, Falco and Scarfone (2011). Furthermore, we used Modbus/TCP for device communication. We also used Mininet to emulate this SCADA network, and Figure 5.2 depicts the topology that we adopted in our evaluation scenario. To respect the time requirements of traditional SCADA systems for power grids, our evaluation scenario consisted of the MTU and subMTUs periodically requesting information from the subordinated devices, every 2 seconds. In this context, every 2 seconds, a master device sends a `READ_COILS` message to its slave devices. In turn, the slave devices reply the original message with the stored values into their registers. The traffic generation of our simulated SCADA system was based on the characteristics (*e.g.*, periodicity, behavior, and size of messages) presented by Barbosa (2014). In addition, focusing on our NIDS, we set the periodicity of network statistics gathering to 4 seconds. Furthermore, our implementation was configured to collect information from Historian Server every 4 seconds too. Thus, in our evaluation scenario, our NIDS was responsible for analyzing the network information stored in Historian Server and for generating alerts to system operators when a possible anomalous behavior in the SCADA communication network is detected. It is important to note that in our experiments, we used the native samples for traffic classification, keeping disabled the Feature Selector component.

## 5.2 Experimental Evaluation

In this section, we describe two case studies used to evaluate the accuracy of our proposed NIDS. Firstly, in Section 5.2.1, we analyze operation of the component when a DoS attack is occurring. Then, in Section 5.2.2, we show the results of our NIDS when used to detect misconfigured devices in the SCADA network. The accuracy of our prototype in discussed in both sections, comparing the ML metrics and analyzing the performance of our solution using the selected OCC algorithms, OCSVM and SVDD.

Figure 5.2: Configuration of the network topology used in our experiments.




Source: by author (2016).

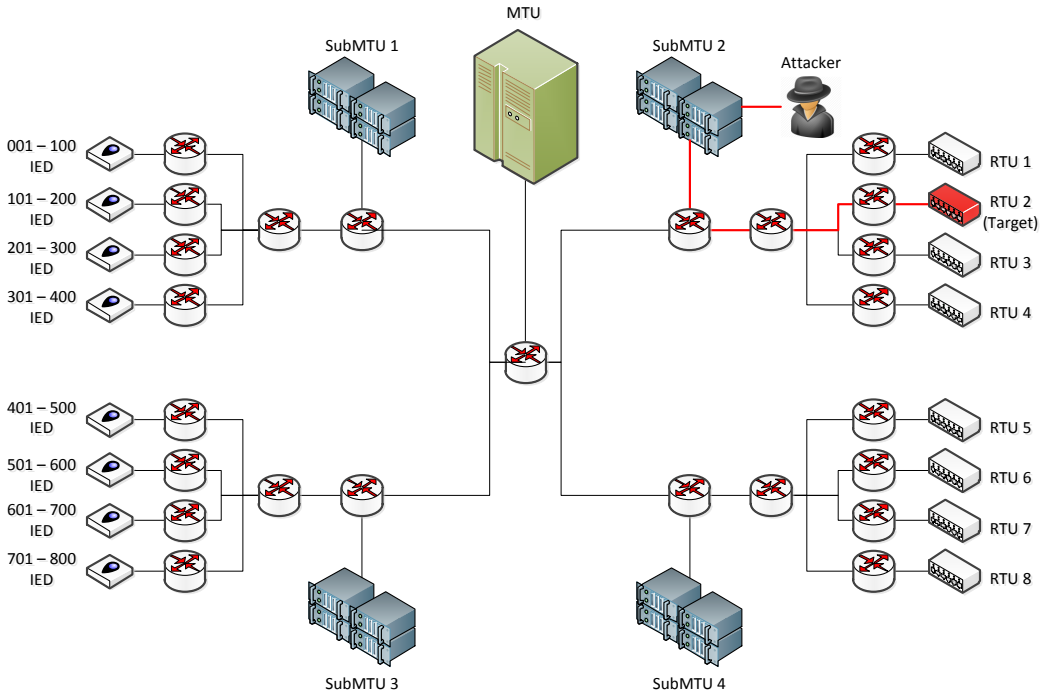
### 5.2.1 Case Study 1 - DoS Attack

In this first case study, we simulate an attack that affects the availability of some SCADA components. For this reason, we simulated the launch of a DoS attack targeted at one of the substations. In our experiments, we assume that this DoS could be launched by a disgruntled employee of the power distribution company who has remote access to one workstation of subMTU #2. As illustrated in Figure 5.3, the disgruntled employee intends to disrupt the communication between a particular distribution substation (RTU #2) and the rest of the power system, forcing a hard-reset in the SCADA system and causing financial losses to the company. The DoS attack is based on a particular Modbus vulnerability named Unauthorized Read Request[6]. This vulnerability allows an attacker with IP connectivity to a SCADA slave device to send unlimited data requests and consequently cause a buffer-overflow in the targeted equipment. Currently, according to Dell (2015), buffer-overflow is a major threat in modern and legacy SCADA systems. It is important to note that in this case study, our aim is to detect the DoS propagation. The experiment conducted in this case study is characterized by monitoring the SCADA network during 20 minutes. This experiment contained two types of traffic, where: 10 minutes represented the expected system functioning; and 10 bursts of 1 minute indicated the occurrence of the DoS attack on the SCADA network. This experiment was performed 30

[6]http://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=20674

times, in order to achieve a confidence level of 95% in the obtained results. The experiments were performed on an Intel Core i7-4790 3.6GHz with 16 GB of RAM memory.

Figure 5.3: Positioning of the disgruntled employee in Case Study 1.




Source: by author (2016).

We evaluated the accuracy of our implementation using the two available SVM-Based OCC algorithms for traffic classification, OCSVM and SVDD. Firstly, we calculated a confusion matrix from the results of our experiments. In this context, the confusion matrix generated four metrics that are described below:

- **True Positive (TP):** In the context of our experiments, a sample predicted as TP indicates benign traffic correctly classified;
- **True Negative (TN):** TN samples characterize the DoS attack that was correctly detected;
- **False Positive (FP):** Samples classified as FP indicate the DoS attack misclassified as benign traffic;
- **False Negative (FN):** FN samples are benign samples misclassified as outliers.

Table 5.2 presents the specific metrics for evaluating supervised ML algorithms. In absolute numbers, the experiments of Case Study 1 generated a total of 29,709 samples which were classified by the NIDS. Of the total of samples generated, 15,579 (52.439%) represented the normal functioning of our SCADA network, whilst the 14,130 (47.561%) remaining samples evidenced the DoS propagation in the power grid. Each repetition of our experiment generated on average 990.3 validation samples, which correspond to 519.3 positive samples and 471 negative samples

on average for each repetition.

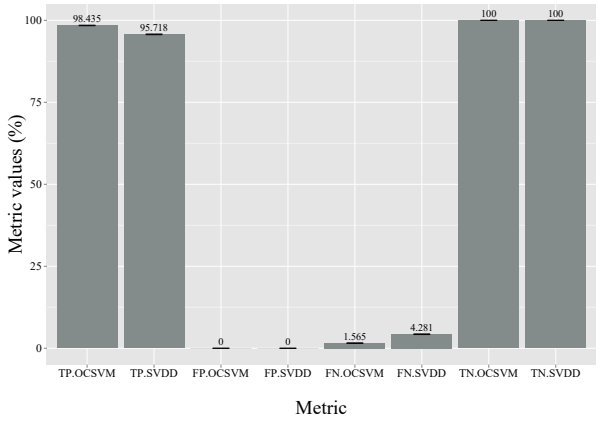
Table 5.2: ML metrics analyzed in our experiments.

| Metric | Formulation | Description |
|---|---|---|
| **True Positive Rate (TPR)** | $\frac{TP}{TP+FN}$ | The proportion of instances classified as positive that are correctly identified |
| **True Negative Rate (TNR)** | $\frac{TN}{FP+TN}$ | The proportion of instances classified as negative that are correctly identified |
| **Positive Predictive Value (PPV)** | $\frac{TP}{TP+FP}$ | The proportion of instances classified as positive that are TP |
| **Negative Predictive Value (NPV)** | $\frac{TN}{TN+FN}$ | The proportion of instances classified as negative that are TN |
| **False Positive Rate (FPR)** | $\frac{FP}{FP+TN}$ | The proportion of instances classified as negative that incorrectly receive a positive test result |
| **False Discovery Rate (FDR)** | $\frac{FP}{FP+TP}$ | The proportion of instances classified as positive that are FP |
| **False Negative Rate (FNR)** | $\frac{FN}{FN+TP}$ | The proportion of positives which yield negative test outcomes with the test |
| **Accuracy (ACC)** | $\frac{TP+TN}{(TP+FN)+(FP+TN)}$ | The proportion of true results among the total number of cases examined |

The confusion matrices presented in Figure 5.4 describe the traffic classification produced by the proposed NIDS. With both algorithms, OCSVM and SVDD, we can observe that our NIDS obtained significant results if we consider the validation samples classified as FP and TN. The confusion matrices show that, for all repetitions of the experiments, our NIDS detected correctly and instantly the validation samples that indicated the propagation of the DoS attack on the SCADA network. In other words, 100% of the validation samples that presented the evidence of DoS attack (TN bars in Figure 5.4) were classified as anomalous behavior, being reported immediately to the SCADA operators. However, if we only analyze the expected behavior, we can see that OCSVM obtained slightly better performance if compared to the SVDD algorithm. OCSVM classified correctly, that is, classified as TP, approximately 98.435% of the validation samples, whilst using SVDD this classification was 95.718%. Figure 5.5 presents a time series of the traffic classification of both OCSVM (Figure 5.6(a)) and SVDD (Figure 5.6(b)), respectively. As Figure 5.6(b) indicates, in the experiment using SVDD, the last three validation samples were classified as FP. This indicates that the RBF Kernel used by OCSVM fits better to the simulated traffic in our SCADA system in relation to the SVDD hypersphere. Note that figures 5.6(a) and 5.6(b), for better visualization of the traffic classification, present only part of an experiment, which contains 40 validation samples where 24 samples are the normal expected traffic, and 16 are the traffic samples evidencing the DoS attack.

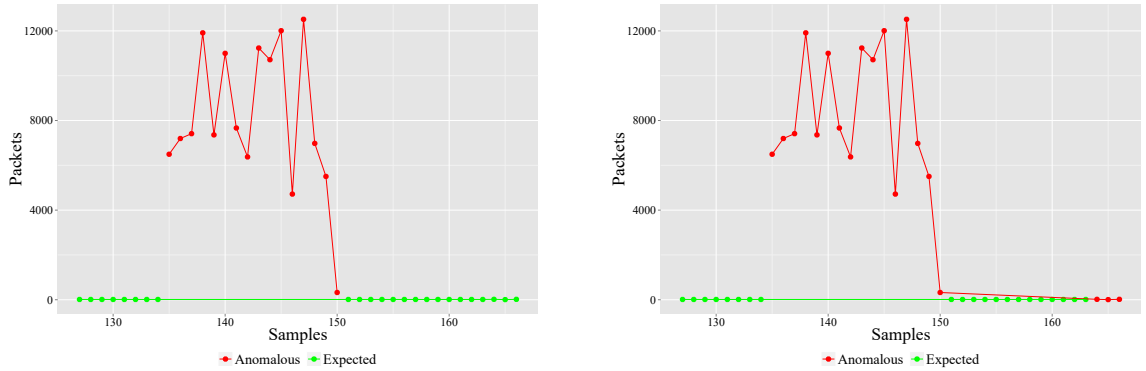
As both algorithms obtained similar FP and TN rates, these results also have influence on metrics for evaluating the accuracy of our NIDS. As can be seen in Figures 5.7(a) and 5.7(b), both algorithms achieved 100% of TNR and PPV. Consequently, OCSVM and SVDD also achieved 0% of FPR and FDR. These results show that the prototype accurately detected and reported the DoS attack in this first case study. Analyzing the remaining metrics, we can see that SVDD presented a higher FNR (4.281%) if compared to OCSVM (1.565%). OCSVM presented slightly higher TPR (98.435% against 95.718%). Furthermore, OCSVM also presented higher rate of NPV (98.31% against 95.501%). Ultimately, OCSVM obtained slightly better accuracy (99.18%) than SVDD (97.755%). Table 5.3 presents an overview of the obtained results in the

Figure 5.4: Confusion matrices generated for Case Study 1.




Source: by author (2016).

Figure 5.5: Traffic classification of our One-Class NIDS for SDN-Based SCADA systems.




(a) Traffic classification using OCSVM.

(b) Traffic classification using SVDD.

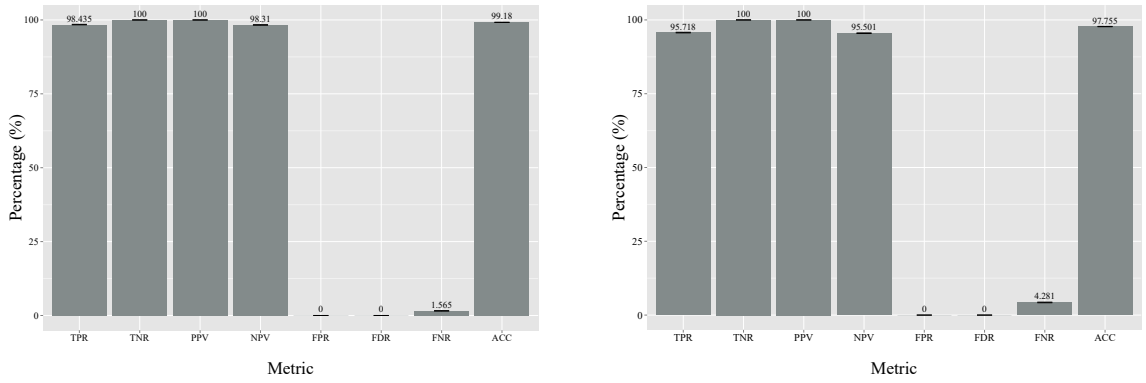
Source: by author (2016).

experiments of Case Study 1.

Table 5.3: Overview of the results obtained in Case Study 1.

| **Metrics / Algorithm** | **OCSVM** | **SVDD** |
|---|---|---|
| **TPR** | $\approx 98.435\%$ | $\approx 95.718\%$ |
| **TNR** | $100\%$ | $100\%$ |
| **PPV** | $100\%$ | $100\%$ |
| **NPV** | $\approx 98.31\%$ | $\approx 95.501\%$ |
| **FPR** | $0\%$ | $0\%$ |
| **FDR** | $0\%$ | $0\%$ |
| **FNR** | $\approx 1.565\%$ | $\approx 4.281\%$ |
| **ACC** | $\approx 99.18\%$ | $\approx 97.755\%$ |

Figure 5.6: ML metrics obtained from the experiments of Case Study 1.




(a) Metrics using OCSVM. (b) Metrics using SVDD.

Source: by author (2016).

The accuracy obtained in this case study is near to the accuracy achieved by Maglaras and Jiang (2014) using OCSVM in a traditional SCADA system (98.88%).Analyzing these results, we can state that in this first case study, our approach was able to fully detect the DoS attack. Although the prototype has classified validation samples as FP, it behaved well in relation to the attack detection. This can be proved by analyzing the results obtained on metrics directly based on TN samples or FP samples, such as TNR, PPV, FPR, and FDR. Our initial experiments also showed that OCSVM presented slightly better accuracy than SVDD in this scenario. This means that, given our assumption about the scarcity of public available attack traces in SCADA, it is possible to detect attacks targeted at the power system by using a classifier model that only represents the expected behavior of the SCADA network.

Finally, we also analyzed the time required for processing the validation samples for both algorithms, OCSVM and SVDD. To collect the processing time for each algorithm, we stored the traces generated in each repetition of this experiment, then we classified once the entire information collected in the experiment. Despite the OCSVM algorithm has obtained better accuracy, SVDD classifies validation samples a bit faster. In our experiments, SVDD classified the validation samples in approximately 6 milliseconds, whilst OCSVM achieved the time of 6.4 milliseconds. Thus, through the results achieved in this case study, we can show that, for example, SVDD may be more suitable for SCADA systems that generate large amounts of data and have small time requirements.

### 5.2.2 Case Study 2 - Misconfiguration of Slave Devices

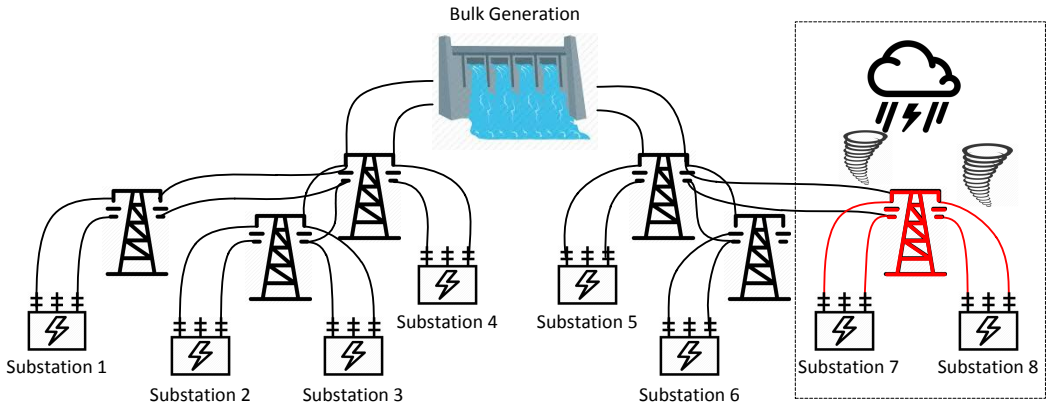
The majority of the proposed IDSes for SCADA systems are designed to detect only anomalies caused by cyber-attackers, such as DoS or malware propagation. Throughout this Masters Thesis, we stated that our solution is not only capable to detect anomalies caused by cyber-

attackers but it is also able to identify misconfigurations induced by both, human interference and natural causes. The ability to notify network anomalies independently of their nature is essential to guarantee the correct operation of SCADA systems, since several equipment (*e.g.*, field devices and IEDs) are operating in open environments, gathering, for example, information about external devices. Thus, SCADA field devices are often exposed to extreme climatic conditions, such as low temperatures, high humidity, sun rays, storms, and so forth.

Differently from the first experiment that focuses on network intrusions, in this case study we focused on simulating an anomaly situation that compromised the integrity of data reported by SCADA IEDs. In other words, the focuses of this case study is to detect network anomalies caused by, for example, misconfigurations or natural disasters. For this reason, we assumed that a natural disaster has damaged some IEDs in the transmission line of our SDN-based SCADA system. More precisely, as represented in Figure 5.7, the natural disaster was based on a tornado that struck the transmission lines that supply substations 7 and 8. This incident misconfigured some IEDs of the SubMTU 3 (IEDs 701-800). In our experiment, these damaged equipment started to report information that does not represent the real state of the transmission lines. For example, IEDs may start to send malformed packets, which are empty or greater than the expected packet size.

The aim of this case study is to detect the aftereffect of an incident. The experiment conducted in this case study is characterized by monitoring the SCADA network during 10 minutes. This experiment contained two types of traffic, where: (*i*) 5 minutes represented the expected system functioning; and (*ii*) the last 5 minutes indicated the occurrence of misconfigured equipment on the SCADA network. As well as Case Study 1, this experiment was performed 30 times, in order to achieve a confidence level of 95% in the obtained results. The experiments were performed on an Intel Core i7-4790 3.6GHz with 16 GB of RAM memory.

Figure 5.7: Area of impact in Case Study 2.

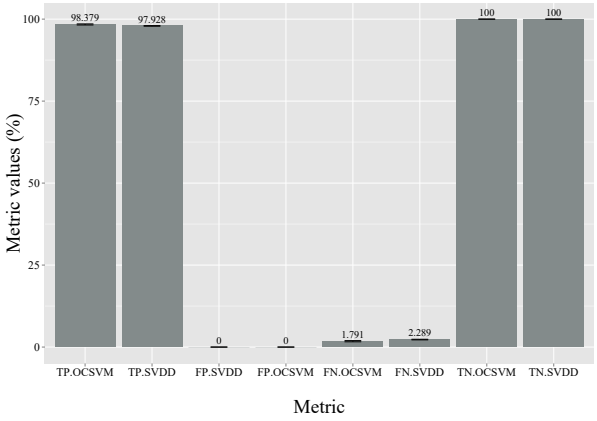



Source: by author (2016).

In this case study, we also evaluated the quality of our NIDS to identify the incident using OCSVM and SVDD. We calculated the confusion matrices resulted from our experiments.

Figure 5.8 presents the confusion matrices resulted from Case Study 2. With both algorithms, OCSVM and SVDD, we can observe that our NIDS obtained again good results considering samples classified as FP and TN. The confusion matrices show that, for all repetitions of the experiments, our NIDS detected correctly and instantly the validation samples that evidenced the equipment malfunction. 100% of the validation samples that represented misconfigurations on IEDs (TN bars in Figure 5.8) were classified as anomalous behavior. However, if we only analyze the expected behavior, we can see that OCSVM maintained a slightly better performance if compared to the SVDD algorithm. OCSVM classified as TP, approximately 98.379% of the validation samples, whilst the SVDD algorithm achieved 97.928% of TP.

Figure 5.8: Confusion matrices generated for Case Study 2.

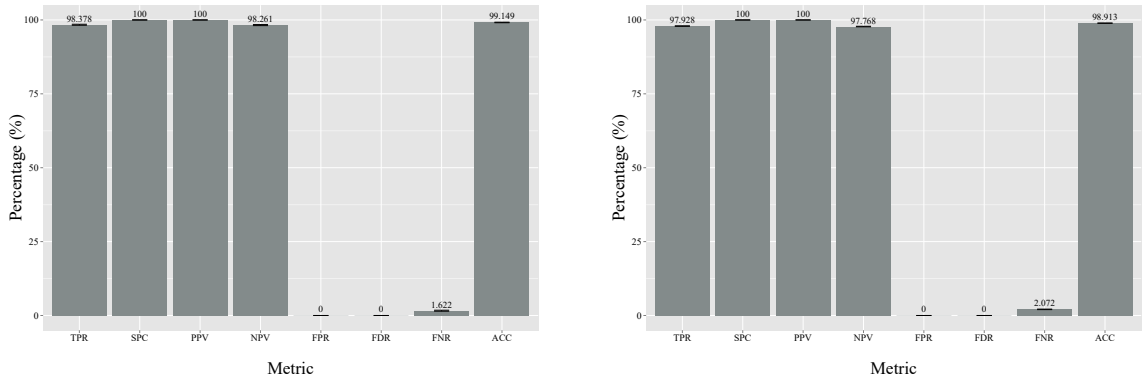



Source: by author (2016).

In this case study, both algorithms achieved again 100% of TNR and PPV, as is showed in Figures 4.10(a) and 4.11(b). Consequently, OCSVM and SVDD also achieved 0% of FPR and FDR. These results show that the One-Class NIDS accurately detected and reported the behavior of misconfigured IEDs in Case Study 2. Analyzing the remaining metrics, we can see that SVDD presented a higher FNR (2.072%) if compared to OCSVM (1.622%). OCSVM presented slightly higher TPR (98.379% against 97.928%). Furthermore, OCSVM also presented higher rate of NPV (98.261% against 97.768%). SVDD also was not was better than OCSVM that obtained an accuracy level of 99.149%. Table 5.4 presents an overview of the obtained results in the experiments of Case Study 2. Analyzing the results of this case study, we can state that: OCC algorithms also can be used in the detection of misconfigured devices in SCADA systems; and, our approach was able to fully detect the anomaly behavior that resulted from malfunctioning devices.

In this last case study, we also seek to evaluate how our NIDS behaves in relation to the training process of the representation model. In a first moment, we created 10 distinct training sets containing 10, 100, 500, 1,000, 5,000, 10,000, 15,000, 20,000, and 25,000 samples that represented the normal behavior of our evaluation scenario. These training sets were used to build the representation model by the prototype using both algorithms, OCSVM and SVDD. Then,

Figure 5.9: ML metrics obtained from the experiments of Case Study 2.




(a) Metrics using OCSVM. (b) Metrics using SVDD.

Source: by author (2016).

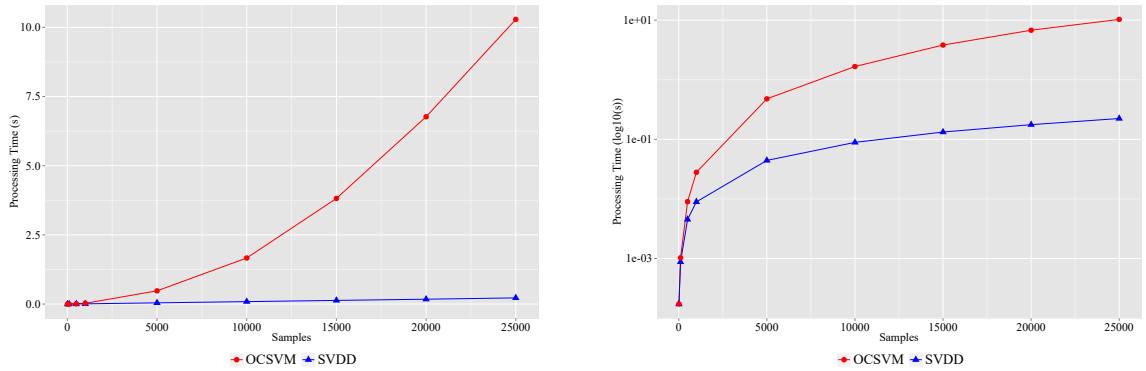
Table 5.4: Overview of the results obtained in Case Study 2.

| **Metrics / Algorithm** | **OCSVM** | **SVDD** |
|---|---|---|
| **TPR** | $\approx 98.378\%$ | $\approx 97.928\%$ |
| **TNR** | $100\%$ | $100\%$ |
| **PPV** | $100\%$ | $100\%$ |
| **NPV** | $\approx 98.261\%$ | $\approx 97.768\%$ |
| **FPR** | $0\%$ | $0\%$ |
| **FDR** | $0\%$ | $0\%$ |
| **FNR** | $\approx 1.622\%$ | $\approx 2.072\%$ |
| **ACC** | $\approx 99.149\%$ | $\approx 98.913\%$ |

we analyzed the processing time required to create the representation model for each training set. Figure 5.10 presents the achieved times in this experiment. The OCSVM algorithm requires more computing resources than SVDD. Analyzing the time for each training set, we can see that the OCSVM processing time increases exponentially, differently from the SVDD algorithm that grows linearly. For example, to the larger training set with 25,000 samples, OCSVM demanded approximately 10.287 seconds to create the representation model, whilst the SVDD algorithm required only 0.223 seconds for accomplishing the same task. To proportionate a better result comparison, Figure 4.11(a) presents the data in decimal scale, whilst Figure 4.11(b) uses logarithmic scale in base 10, permitting the visualization of processing times even for the smaller training sets.

Lastly, we still used the same training sets to calculate the amount of memory required for creating the representation model for each OCC algorithm. Although OCSVM is more accurate than SVDD, Figure 5.11 shows that OCSVM requires more memory to describe the representation model that classifies the SCADA traffic. This characteristic is due to the fact that

Figure 5.10: Processing Time to create the representation model in Case Study 2.



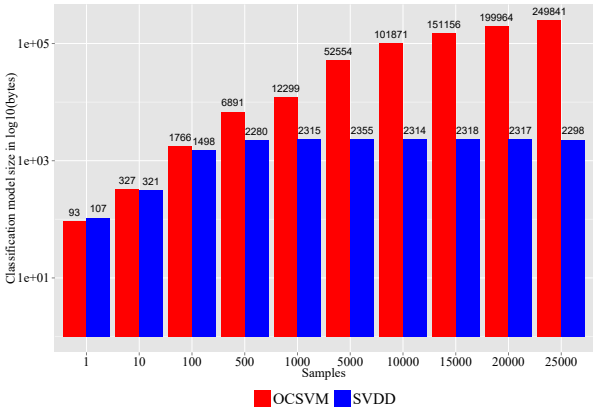

(a) Using decimal scale.

(b) Using $\log_{10}$ scale.

Source: by author (2016).

the RBF Kernel creates a decision boundary that fits better than a hypersphere to the training data. In addition, SVDD automatically removes unnecessary similar samples that are included in the training set, thus optimizing the size of the representation model. In other words, an SVDD representation model may have a smaller size if compared to an OCSVM representation model. In our experiments, as we can see in Figure 5.11 we obtained similar results for the smaller training sets (*e.g.*, with 10 and 100 samples). However, OCSVM requires much more memory than SVDD to the larger training sets. For example, in our experiments, to the training set with 25,000 samples, OCSVM created the representation model with sized approximately 250 kilobytes, whilst SVDD used only 2.3 kilobytes.

Figure 5.11: Memory used to create the representation model in Case Study 2.




Source: by author (2016).

# 6 CONCLUDING REMARKS

Power grids are responsible for the generation, transmission, and distribution of electricity to end-users. However, over the recent years, power grids are becoming more sophisticated, with the aim of increasing their safety, reliability, economical and energy efficiency, and reducing their environmental impact. Usually, power grids are controlled and monitored by large-scale SCADA systems. Because of their fundamental importance, any threat to the operation of SCADA systems may result in heavy economical losses or even put lives in danger. Therefore, in order to promote the modernization process of power grids, we investigate the development of a new generation of SCADA systems, named SDN-based SCADA systems. In this context, SDN-based SCADA systems can facilitate the design and development of Smart Grid network applications, by making them more secure and resilient.

By relying on the global view of the SDN-based SCADA network and on their ability to gather switch statistics, we presented a specific NIDS for this kind of environment. The NIDS proposed in this Masters Thesis relies on OCC algorithms, a specific kind of ML techniques that, with a unique inlier homogeneous training set, can detect anomalous behaviors in SCADA networks caused by human or natural causes, such as unauthorized system or network activity. We presented experimental results in a realistic large-scale SCADA environment that validate our prototype and verified the accuracy of the classification techniques, applied to the detection of network anomalies in two case-studies: (*i*) a DoS attack based on a real vulnerability of the Modbus/TCP protocol; and, (*ii*) a natural disaster that was responsible for misconfiguring several SCADA field devices. Our analysis was based on a comparison of two OCC algorithms, OCSVM and SVDD, which deal well with large datasets and have a fast classification process if compared to other ML techniques.

## 6.1 Summary of Contributions

In this document, we presented a list of contributions to the state-of-the-art, such as:

- The results of a comprehensive investigation of the applicability and benefits of applying SDN in Smart Grid environments, mainly in the context of SCADA systems. During our research, we detected that the characteristics of SDN, such as its flexibility, programmability, the centralized management, and its standard API can enhance the level of resilience and robustness of SCADA systems. In addition, we listed possible benefits of SDN-based SCADA systems for each FCAPS property;
- An investigation of the basic requirements of NIDSes for SCADA systems applied in the context of power grids. Based on the literature, we cited that for a NIDS to properly operate in the environment of a SCADA system, it must: (*i*) respect the SCADA sampling period; (*ii*) be scalable; (*iii*) constantly manipulate large datasets; and, (*iv*) adapt itself to the diversity of available technologies for SCADA systems;

- A review of the existing techniques utilized for traffic classification. Thus, we identified the OCC approach, a class of ML algorithms that can be used for detecting anomalous behaviors in SCADA networks without relying on third-party SCADA traces. In this document, we detailed the operation of two SVM-based OCC algorithms, OCSVM and SVDD. In our experiments, these algorithms were capable of detecting network anomalous behavior caused by human interference, and natural disasters. Furthermore, both, OCSVM and SVDD, achieved significant results, presenting an accuracy level of approximately 99%;
- A NIDS specific for SDN-based SCADA systems that addresses the basic requirements of traditional IDSes for power grids. Our proposal comprises a flexible traffic classifier based on OCC techniques to detect anomalies in SCADA networks. In addition, our strategy relies on SDN/OpenFlow to periodically collect information about the SCADA network. To the best of our knowledge, this is the first time that a solution that merges SDN, SCADA, and OCC is proposed to enhance the resilience of SCADA systems for power grids.

## 6.2 Discussion and Lessons Learned

SCADA systems are commonly used to aid the operation of critical infrastructures, including some that are considered to be essential for our society, *e.g.* power grids. In the past, these systems were completely isolated and relied on specific hardware and software components. However, SCADA systems are becoming increasingly interconnected to the Internet to increase productivity and to reduce the OPEX, thus bringing new security threats. In order to enhance the security level of SCADA systems, several traditional IT NIDSes have been adapted to detect unexpected behaviors in SCADA environments. Nevertheless, the different nature and characteristics of SCADA networks have motivated researchers to develop NIDSes specific for SCADA systems. Consequently, a number of NIDSes have been specifically developed to meet the requirements of SCADA systems.

With the appearance of the SDN paradigm, new researches efforts have aimed to insert SDN in the context of power grids and SCADA systems. Some proposals appeared to improve SCADA systems, and to provide better services to the power grid end-users. Thus, we investigated how SDN can assist to enhance the resilience level of SCADA systems, and we proposed the use of SDN to collect more accurate information from the SCADA network. Therefore, we could learn how a SCADA network behaves and how a cyber attack targeted to this environment will manifest. In addition, we proposed a NIDS that creates signatures of the expected network operation, *e.g.* the proposal generates behavior models of the correct functioning of SCADA devices.

Our proposal was designed to use OCC algorithms. This design definition allowed us to built a theoretical background about the main available OCC approaches. In this universe of

possibilities, we chose two SVM-based OCC algorithms, OCSVM and SVDD. In our experiments, we could state that these algorithms can be used to detect network anomalies in SCADA environments, and thus, we also could learn the characteristics of these two algorithms. In our experiments, we could observe that OCSVM adapted itself better than SVDD to the SCADA network traces. However, the SVDD algorithm is faster and requires less computing resources than OCSVM. For this reason, SVDD may be more indicated for SCADA systems with more rigorous time restrictions, whilst OCSVM may be more appropriate for SCADA that needs better accuracy.

## 6.3 Final Remarks and Future Work

As possible future work, we intend to expand the range of initially available OCC approaches in our proposal, inserting, for example, algorithms such as the Kernel Principal Component Analysis (KPCA) (HOFFMANN, 2007) and the One-Class Random Forests (OCRF) (DÉSIR et al., 2013). We plan to carry out a performance analysis of our NIDS using these new OCC algorithms against real SCADA threats, such as the Stuxnet worm. Still with respect to traffic classification, we also intend to provide solutions based on combined classifiers, implementing for example, a mix between a unsupervised algorithm (*e.g.*, K-Means) and an available OCC algorithm, in order to improve the accuracy of our proposal. In addition, we intend to incorporate the Feature Selector component in our solution to minimize possible false alerts. In order to improve the components of our prototype, we intend, for example, to permit that our NIDS classifies on-line the SCADA traffic. Furthermore, we intend to evaluate the utilization of different MapReduce frameworks in our solution, in order to detect the implementation that offers the best trade-off for our NIDS in terms of data processing and scalability. Finally, we plan to implement a user-friendly interface, for defining alternatives to mitigate the anomalous behaviors without compromising the functioning of SCADA devices.

# REFERENCES

AL-DALKY, R. et al. A modbus traffic generator for evaluating the security of scada systems. In: **Communication Systems, Networks Digital Signal Processing (CSNDSP), 2014 9th International Symposium on**. [S.l.: s.n.], 2014. p. 809–814.

ALMALAWI, A. **Designing unsupervised intrusion detection for SCADA systems**. Thesis (PhD) — RMIT University, Melbourne, Victoria, Australia., December 2014. Available from Internet: <https://researchbank.rmit.edu.au/view/rmit:161104>.

ALMALAWI, A. et al. An Unsupervised Anomaly-Based Detection Approach for Integrity Attacks on SCADA Systems. **Computers & Security**, v. 46, n. 0, p. 94 – 110, 2014. ISSN 0167-4048.

ASIF, M.; AL-HARTHI, Y. Intrusion detection system using honey token based encrypted pointers to mitigate cyber threats for critical infrastructure networks. In: **Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 1266–1270.

BAILEY, D.; WRIGHT, E. Background to {SCADA}. In: BAILEY, D.; WRIGHT, E. (Ed.). **Practical {SCADA} for Industry**. Oxford: Newnes, 2003. p. 1 – 10. ISBN 978-0-7506-5805-8. Available from Internet: <http://www.sciencedirect.com/science/article/pii/B9780750658058500015>.

BARBOSA, R.; SADRE, R.; PRAS, A. A first look into scada network traffic. In: **Network Operations and Management Symposium (NOMS), 2012 IEEE**. [S.l.: s.n.], 2012. p. 518–521. ISSN 1542-1201.

BARBOSA, R. R. R. **Anomaly detection in SCADA systems: a network based approach**. Thesis (PhD) — University of Twente, Enschede, April 2014. Available from Internet: <http://doc.utwente.nl/90271/>.

BAUDAT, G.; ANOUAR, F. Kernel-based methods and function approximation. In: **Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on**. [S.l.: s.n.], 2001. v. 2, p. 1244–1249 vol.2. ISSN 1098-7576.

BENDER, K. (Ed.). **Profibus: The Fieldbus for Industrial Automation**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. ISBN 0-13-012691-8.

BENKEDJOUH, T. et al. Fault prognostic of bearings by using support vector data description. In: **Prognostics and Health Management (PHM), 2012 IEEE Conference on**. [S.l.: s.n.], 2012. p. 1–7.

BIGHAM, J.; GAMEZ, D.; LU, N. Safeguarding scada systems with anomaly detection. In: GORODETSKY, V.; POPYACK, L.; SKORMIN, V. (Ed.). **Computer Network Security**. Springer Berlin Heidelberg, 2003, (Lecture Notes in Computer Science, v. 2776). p. 171–182. ISBN 978-3-540-40797-3. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-45215-7_14>.

BORAH, B.; BHATTACHARYYA, D. K. An improved sampling-based dbscan for large spatial databases. In: **Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on**. [S.l.: s.n.], 2004. p. 92–96.

BOYER, S. A. **Scada: Supervisory Control And Data Acquisition**. 4th. ed. USA: International Society of Automation, 2009. ISBN 1936007096, 9781936007097.

BREZZI, F. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. **Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique**, v. 8, n. 2, p. 129–151, 1974.

BRUNNER, C. Iec 61850 for power system communication. In: **2008 IEEE/PES Transmission and Distribution Conference and Exposition**. [S.l.: s.n.], 2008. p. 1–6. ISSN 2160-8555.

CAHN, A. et al. Software-defined energy communication networks: From substation automation to future smart grids. In: **Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on**. [S.l.: s.n.], 2013. p. 558–563.

CARCANO, A. et al. A multidimensional critical state analysis for detecting intrusions in scada systems. **Industrial Informatics, IEEE Transactions on**, v. 7, n. 2, p. 179–186, May 2011. ISSN 1551-3203.

CARCANO, A. et al. State-based network intrusion detection systems for scada protocols: A proof of concept. In: **Proceedings of the 4th International Conference on Critical Information Infrastructures Security**. Berlin, Heidelberg: Springer-Verlag, 2010. (CRITIS'09), p. 138–150. ISBN 3-642-14378-4, 978-3-642-14378-6. Available from Internet: <http://dl.acm.org/citation.cfm?id=1880551.1880563>.

CARDENAS, A. A. et al. Attacks against process control systems: Risk assessment, detection, and response. In: **Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security**. New York, NY, USA: ACM, 2011. (ASIACCS '11), p. 355–366. ISBN 978-1-4503-0564-8. Available from Internet: <http://doi.acm.org/10.1145/1966913.1966959>.

CARUANA, R.; NICULESCU-MIZIL, A. An Empirical Comparison of Supervised Learning Algorithms. In: **Proceedings of the 23rd International Conference on Machine Learning**. New York, NY, USA: ACM, 2006. (ICML '06), p. 161–168. ISBN 1-59593-383-2.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly Detection: A survey. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 41, n. 3, p. 15:1–15:58, jul. 2009. ISSN 0360-0300.

CHEESEMAN, P. et al. Readings in knowledge acquisition and learning. In: BUCHANAN, B. G.; WILKINS, D. C. (Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. chp. AutoClass: A Bayesian Classification System, p. 431–441. ISBN 1-55860-163-5. Available from Internet: <http://dl.acm.org/citation.cfm?id=170641.170679>.

CHERKASSKY, V.; MA, Y. Practical selection of {SVM} parameters and noise estimation for {SVM} regression. **Neural Networks**, v. 17, n. 1, p. 113 – 126, 2004. ISSN 0893-6080. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0893608003001692>.

CHEUNG, S. et al. Using model-based intrusion detection for scada networks. In: **Proceedings of the SCADA Security Scientific Symposium**. Miami Beach, Florida: [s.n.], 2007.

CHIKUNI, E.; DONDO, M. Investigating the security of electrical power systems scada. In: **AFRICON 2007**. [S.l.: s.n.], 2007. p. 1–7.

CLARKE, G. R.; REYNDERS, D.; WRIGHT, E. **Practical modern SCADA protocols: DNP3, 60870.5 and related systems**. [S.l.]: Newnes, 2004.

CORTES, C.; VAPNIK, V. Support-Vector Networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

DAINOTTI, A.; PESCAPE, A.; CLAFFY, K. C. Issues and future directions in traffic classification. **IEEE Network**, v. 26, n. 1, p. 35–40, January 2012. ISSN 0890-8044.

D'ANTONIO, S.; OLIVIERO, F.; SETOLA, R. High-speed intrusion detection in support of critical infrastructure protection. In: **Proceedings of the First International Conference on Critical Information Infrastructures Security**. Berlin, Heidelberg: Springer-Verlag, 2006. (CRITIS'06), p. 222–234. ISBN 3-540-69083-2, 978-3-540-69083-2. Available from Internet: <http://dx.doi.org/10.1007/11962977_18>.

DEAN, J.; GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. **Commun. ACM**, ACM, New York, NY, USA, v. 51, n. 1, p. 107–113, jan. 2008. ISSN 0001-0782.

DELL. **Dell Security Annual Threat Report**. [S.l.], 2015. Available from Internet: <https://software.dell.com/whitepaper/dell-network-security-threat-report-2014874708>.

DÉSIR, C. et al. One Class Random Forests. **Pattern Recognition**, v. 46, n. 12, p. 3490 – 3506, 2013. ISSN 0031-3203.

DONG, X. et al. Software-defined networking for smart grid resilience: Opportunities and challenges. In: **Proceedings of the 1st ACM Workshop on Cyber-Physical System Security**. New York, NY, USA: ACM, 2015. (CPSS '15), p. 61–68. ISBN 978-1-4503-3448-8. Available from Internet: <http://doi.acm.org/10.1145/2732198.2732203>.

DORSCH, N. et al. Software-defined networking for smart grid communications: Applications, challenges and advantages. In: **Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 422–427.

DUESSEL, P. et al. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. In: **Proceedings of the 4th International Conference on Critical Information Infrastructures Security**. Berlin, Heidelberg: Springer-Verlag, 2010. (CRITIS'09), p. 85–97. ISBN 3-642-14378-4, 978-3-642-14378-6. Available from Internet: <http://dl.acm.org/citation.cfm?id=1880551.1880559>.

EDMONDS, J.; PAPA, M.; SHENOI, S. Critical infrastructure protection. In: ____. Boston, MA: Springer US, 2008. chp. Security Analysis of Multilayer SCADA Protocols, p. 205–221. ISBN 978-0-387-75462-8. Available from Internet: <http://dx.doi.org/10.1007/978-0-387-75462-8_15>.

ERMAN, J.; ARLITT, M.; MAHANTI, A. Traffic classification using clustering algorithms. In: **Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data**. New York, NY, USA: ACM, 2006. (MineNet '06), p. 281–286. ISBN 1-59593-569-X. Available from Internet: <http://doi.acm.org/10.1145/1162678.1162679>.

ESTE, A.; GRINGOLI, F.; SALGARELLI, L. Support vector machines for tcp traffic classification. **Comput. Netw.**, Elsevier North-Holland, Inc., New York, NY, USA, v. 53, n. 14, p. 2476–2490, sep. 2009. ISSN 1389-1286. Available from Internet: <http://dx.doi.org/10.1016/j.comnet.2009.05.003>.

FARHANGI, H. The path of the smart grid. **Power and Energy Magazine, IEEE**, v. 8, n. 1, p. 18–28, January 2010. ISSN 1540-7977.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The Road to SDN. **Queue**, ACM, New York, NY, USA, v. 11, n. 12, p. 20:20–20:40, dec. 2013. ISSN 1542-7730.

FERNANDEZ, E. B.; LARRONDO-PETRIE, M. M. Designing secure scada systems using security patterns. In: **System Sciences (HICSS), 2010 43rd Hawaii International Conference on**. [S.l.: s.n.], 2010. p. 1–8. ISSN 1530-1605.

FOVINO, I. et al. Modbus/dnp3 state-based intrusion detection system. In: **Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on**. [S.l.: s.n.], 2010. p. 729–736. ISSN 1550-445X.

FRANK, M.; WOLFE, P. An algorithm for quadratic programming. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 3, n. 1-2, p. 95–110, 1956. ISSN 1931-9193. Available from Internet: <http://dx.doi.org/10.1002/nav.3800030109>.

FRIEDBERG, I.; MCLAUGHLIN, K.; SMITH, P. Towards a cyber-physical resilience framework for smart grids. In: ____. **Intelligent Mechanisms for Network Configuration and Security: 9th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2015, Ghent, Belgium, June 22-25, 2015. Proceedings**. Cham: Springer International Publishing, 2015. p. 140–144. ISBN 978-3-319-20034-7. Available from Internet: <http://dx.doi.org/10.1007/978-3-319-20034-7_15>.

GOODNEY, A. et al. Efficient pmu networking with software defined networks. In: **Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on**. [S.l.: s.n.], 2013. p. 378–383.

GYLLSTROM, D.; BRAGA, N.; KUROSE, J. Recovery from link failures in a smart grid communication network using openflow. In: **Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 254–259.

HADLEY, M.; HUSTON, K. Secure SCADA Communication Protocol Performance Test Results. **Pacific Northwest National Laboratory (August 2007)**, 2007.

HARES, S.; WHITE, R. Software-defined networks and the interface to the routing system (i2rs). **IEEE Internet Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 17, n. 4, p. 84–88, 2013. ISSN 1089-7801.

HOFFMANN, H. Kernel PCA for Novelty Detection. **Pattern Recognition**, v. 40, n. 3, p. 863 – 874, 2007. ISSN 0031-3203.

HUITSING, P. et al. Attack Taxonomies for the Modbus Protocols. **International Journal of Critical Infrastructure Protection**, v. 1, p. 37 – 44, 2008. ISSN 1874-5482.

IGURE, V. M.; LAUGHTER, S. A.; WILLIAMS, R. D. Security issues in scada networks. **Computers & Security**, v. 25, n. 7, p. 498 – 506, 2006. ISSN 0167-4048. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167404806000514>.

JAIN, A. K. Data clustering: 50 years beyond k-means. **Pattern Recognition Letters**, v. 31, n. 8, p. 651 – 666, 2010. ISSN 0167-8655. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)19th International Conference in Pattern Recognition (ICPR). Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167865509002323>.

JANSSENS, J. **Outlier selection and one-class classification**. Thesis (PhD) — Tilburg University, 2013.

KANG, D. J. et al. Analysis on cyber threats to scada systems. In: **2009 Transmission Distribution Conference Exposition: Asia and Pacific**. [S.l.: s.n.], 2009. p. 1–4. ISSN 2160-8636.

KHAN, S.; MADDEN, M. A Survey of Recent Trends in One Class Classification. In: COYLE, L.; FREYNE, J. (Ed.). **Artificial Intelligence and Cognitive Science**. [S.l.]: Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6206). p. 188–197. ISBN 978-3-642-17079-9.

KIM, Y. jin et al. Virtualized and self-configurable utility communications enabled by software-defined networks. In: **Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 416–421.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

LEE, H. H. et al. Advanced intelligent computing theories and applications. with aspects of theoretical and methodological issues: 4th international conference on intelligent computing, icic 2008 shanghai, china, september 15-18, 2008 proceedings. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. chp. Real-Time Communications on IEC 61850 Process Bus Based Distributed Sampled Measured Values Applications in Merging Unit, p. 1250–1257. ISBN 978-3-540-87442-3. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-87442-3_154>.

LENG, Q. et al. One-class classification with extreme learning machine. **Mathematical Problems in Engineering**, Hindawi Publishing Corporation, v. 2015, 2015.

LEWIS, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In: ____. **Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 4–15. ISBN 978-3-540-69781-7. Available from Internet: <http://dx.doi.org/10.1007/BFb0026666>.

LI, X. et al. Securing smart grid: cyber attacks, countermeasures, and challenges. **IEEE Communications Magazine**, v. 50, n. 8, p. 38–45, August 2012. ISSN 0163-6804.

LINDA, O.; VOLLMER, T.; MANIC, M. Neural network based intrusion detection system for critical infrastructures. In: **Neural Networks, 2009. IJCNN 2009. International Joint Conference on**. [S.l.: s.n.], 2009. p. 1827–1834. ISSN 1098-7576.

LU, X. et al. On network performance evaluation toward the smart grid: A case study of dnp3 over tcp/ip. In: **Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE**. [S.l.: s.n.], 2011. p. 1–6. ISSN 1930-529X.

MACKIEWICZ, R. E. Overview of iec 61850 and benefits. In: **2006 IEEE PES Power Systems Conference and Exposition**. [S.l.: s.n.], 2006. p. 623–630.

MAGLARAS, L.; JIANG, J. Intrusion detection in scada systems using machine learning techniques. In: **Science and Information Conference (SAI), 2014**. [S.l.: s.n.], 2014. p. 626–631.

MCCLANAHAN. The benefits of networked scada systems utilizing ip-enabled networks. In: **Rural Electric Power Conference, 2002. 2002 IEEE**. [S.l.: s.n.], 2002. p. C5–C7.

MCCLANAHAN. Scada and ip: is network convergence really here? **Industry Applications Magazine, IEEE**, v. 9, n. 2, p. 29–36, Mar 2003. ISSN 1077-2618.

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. Available from Internet: <http://doi.acm.org/10.1145/1355734.1355746>.

MOHAGHEGHI, S.; STOUPIS, J.; WANG, Z. Communication protocols and networks for power systems-current status and future trends. In: **Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES**. [S.l.: s.n.], 2009. p. 1–9.

MONSANTO, C. et al. Composing software defined networks. In: **Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)**. Lombard, IL: USENIX, 2013. p. 1–13. ISBN 978-1-931971-00-3. Available from Internet: <https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/monsanto>.

NADER, P.; HONEINE, P.; BEAUSEROY, P. Intrusion Detection in SCADA Systems using One-Class Classification. In: **Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European**. [S.l.: s.n.], 2013. p. 1–5.

NGUYEN, T. T. T.; ARMITAGE, G. A survey of techniques for internet traffic classification using machine learning. **IEEE Communications Surveys Tutorials**, v. 10, n. 4, p. 56–76, Fourth 2008. ISSN 1553-877X.

NIST. Roadmap for smart grid interoperability standards. **National Institute of Standards and Technology**, n. r31108r3, p. 239, September 2014. Available from Internet: <http://www.nist.gov/smartgrid/upload/NIST-SP-1108r3.pdf>.

OMAN, P.; PHILLIPS, M. Intrusion detection and event monitoring in scada networks. In: GOETZ, E.; SHENOI, S. (Ed.). **Critical Infrastructure Protection**. Springer US, 2008, (IFIP International Federation for Information Processing, v. 253). p. 161–173. ISBN 978-0-387-75461-1. Available from Internet: <http://dx.doi.org/10.1007/978-0-387-75462-8_12>.

PAOLUCCI, F. et al. A survey on the path computation element (pce) architecture. **Communications Surveys Tutorials, IEEE**, v. 15, n. 4, p. 1819–1841, Fourth 2013. ISSN 1553-877X.

PARTHASARATHY, S.; KUNDUR, D. Bloom filter based intrusion detection for smart grid scada. In: **Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on**. [S.l.: s.n.], 2012. p. 1–6. ISSN 0840-7789.

PFEIFFENBERGER, T. et al. Reliable and flexible communications for power systems: Fault-tolerant multicast with sdn/openflow. In: **New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on**. [S.l.: s.n.], 2015. p. 1–6.

PREMARATNE, U. et al. An intrusion detection system for iec61850 automated substations. **Power Delivery, IEEE Transactions on**, v. 25, n. 4, p. 2376–2383, Oct 2010. ISSN 0885-8977.

RINALDI, S. et al. Software defined networking applied to the heterogeneous infrastructure of smart grid. In: **Factory Communication Systems (WFCS), 2015 IEEE World Conference on**. [S.l.: s.n.], 2015. p. 1–4.

SAYEGH, N. et al. Scada intrusion detection system based on temporal behavior of frequent patterns. In: **Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE**. [S.l.: s.n.], 2014. p. 432–438.

SCHöLKOPF, B.; SMOLA, A. **Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond**. Cambridge, MA, USA: MIT Press, 2001. ISBN 0262194759.

SCIACCA, S. C.; BLOCK, W. R. Advanced scada concepts. **IEEE Computer Applications in Power**, v. 8, n. 1, p. 23–28, Jan 1995. ISSN 0895-0156.

SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. **Communications Magazine, IEEE**, v. 51, n. 7, p. 36–43, July 2013. ISSN 0163-6804.

SIDHU, T. S.; YIN, Y. Modelling and simulation for performance evaluation of iec61850-based substation communication systems. **IEEE Transactions on Power Delivery**, v. 22, n. 3, p. 1482–1489, July 2007. ISSN 0885-8977.

SILVA, A. et al. Identification and Selection of Flow Features for Accurate Traffic Classification in SDN. In: **Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on**. [S.l.: s.n.], 2015. To appear.

SILVA, E. Germano da et al. Capitalizing on sdn-based scada systems: An anti-eavesdropping case-study. In: **Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on**. [S.l.: s.n.], 2015. p. 165–173.

SONG, I.-K. et al. Operation schemes of smart distribution networks with distributed energy resources for loss reduction and service restoration. **Smart Grid, IEEE Transactions on**, v. 4, n. 1, p. 367–374, March 2013. ISSN 1949-3053.

STOUFFER, K.; FALCO, J.; SCARFONE, K. Guide to Industrial Control Systems (ICS) Security. **NIST special publication**, Citeseer, p. 800–82, 2011.

STRAYER, W. T.; WEAVER, A. C. Performance measurement of data transfer services in map. **IEEE Network**, v. 2, n. 3, p. 75–81, May 1988. ISSN 0890-8044.

SWALES, A. Open modbus/tcp specification. **Schneider Electric**, v. 29, 1999.

TAX, D.; DUIN, R. Support Vector Domain Description. **Pattern Recognition Letters**, v. 20, n. 11–13, p. 1191 – 1199, 1999. ISSN 0167-8655.

TAX, D.; DUIN, R. Support Vector Data Description. **Machine Learning**, Kluwer Academic Publishers-Plenum Publishers, v. 54, n. 1, p. 45–66, 2004. ISSN 0885-6125.

TAX, D. M. **One-class classification**. [S.l.]: TU Delft, Delft University of Technology, 2001.

THOMAS, M. S.; MCDONALD, J. D. **Power system SCADA and smart grids**. [S.l.]: CRC Press, 2015.

THOMESSE, J.-P. Fieldbus technology in industrial automation. **Proceedings of the IEEE**, v. 93, n. 6, p. 1073–1101, June 2005. ISSN 0018-9219.

VALDES, A.; CHEUNG, S. Communication pattern anomaly detection in process control systems. In: **Technologies for Homeland Security, 2009. HST '09. IEEE Conference on**. [S.l.: s.n.], 2009. p. 22–29.

WANG, F. Y. et al. Protocol design and performance analysis for manufacturing message specification: A petri net approach. **IEEE Transactions on Industrial Electronics**, v. 41, n. 6, p. 641–653, Dec 1994. ISSN 0278-0046.

WANG, W.; XU, Y.; KHANNA, M. A Survey on the Communication Architectures in Smart Grid. **Computer Networks**, Elsevier North-Holland, Inc., New York, NY, USA, v. 55, n. 15, p. 3604–3629, oct. 2011. ISSN 1389-1286.

WHITLEY, D. A genetic algorithm tutorial. **Statistics and computing**, Springer, v. 4, n. 2, p. 65–85, 1994.

WICKBOLDT, J. et al. Software-Defined Networking: Management requirements and challenges. **Communications Magazine, IEEE**, v. 53, n. 1, p. 278–285, January 2015. ISSN 0163-6804.

YAN, Y. et al. A Survey on Smart Grid Communication Infrastructures: Motivations, requirements and challenges. **Communications Surveys Tutorials, IEEE**, v. 15, n. 1, p. 5–20, First 2013. ISSN 1553-877X.

YANG, Y. et al. Intrusion detection system for iec 60870-5-104 based scada networks. In: **Power and Energy Society General Meeting (PES), 2013 IEEE**. [S.l.: s.n.], 2013. p. 1–5. ISSN 1944-9925.

ZHANG, J. et al. Opportunities for software-defined networking in smart grid. In: **Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on**. [S.l.: s.n.], 2013. p. 1–5.

ZHU, B.; JOSEPH, A.; SASTRY, S. A taxonomy of cyber attacks on scada systems. In: **Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing**. Washington, DC, USA: IEEE Computer Society, 2011. (ITHINGSCPSCOM '11), p. 380–388. ISBN 978-0-7695-4580-6.

## AppendixA PUBLISHED PAPER – IM 2015

In this paper we present the benefits of adopting SDN on SCADA environments in relation to traditional supervisory systems of control and data acquisition. The work also presents a SDN-based mechanism os multipath routing that improve the confidentiality level of exchanged messages between power grid components. Our multipath routing avoids that an eavesdropper fully capture a communication between the control center and the power distribution substations. The approach installs few rules on the SCADA switches, for this we relied on static rules (that do not expire) and dynamic rules (that are frequently reinstalled and determinate the communication flow behavior between devices). We carried out an experimental evaluation comparing our approach (using different times of dynamic rules) and the POX standard behavior (single path routing). The experiments demonstrated that our approach, even installing few flow rules and without overloading the SDN controller and switches, was able to change the communication flows behavior of devices, improving the confidentiality level of messages exchanged on the SCADA network.

- **Title –**
  *Capitalizing on SDN-Based SCADA Systems: An Anti-Eavesdropping Case-Study*
- **Conference –**
  The 14th IFIP/IEEE International Symposium on Integrated Network Management (IM-2015)
- **Type –**
  Main track (full-paper)
- **Qualis –**
  B1
- **URL –**
  <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7140289>
- **Date –**
  May 11-15, 2015
- **Held at –**
  Ottawa, Canada
- **Digital Object Identifier (DOI) –**
  <http://dx.doi.org/10.1109/INM.2015.7140289>

# Capitalizing on SDN-Based SCADA Systems: An Anti-Eavesdropping Case-Study

Eduardo Germano da Silva, Luis Augusto Dias Knob, Juliano Araujo Wickboldt,
Luciano Paschoal Gaspary, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
Email: {eduardo.germano, luis.knob, jwickboldt, paschoal, granville, alberto}@inf.ufrgs.br

***Abstract*—Power grids are responsible for the transmission and distribution of electricity to end-users. These systems are undergoing a modernization process through the use of Information and Communication Technology (ICT), transforming the electric system into Smart Grids. In this context, Supervisory Control and Data Acquisition (SCADA) systems are responsible for the management and monitoring of substations and field devices. In this paper, we investigate the use of SDN as an approach to assist in the modernization of SCADA systems. We discuss its possible benefits, such as simplified management of power system resources. Moreover, SDN can facilitate the creation of new network applications that previously, with traditional networks, were more complex to be implemented. To illustrate the benefits of the use of SDN in SCADA, we designed a mechanism that aims to prevent a possible eavesdropper from fully capturing communication flows between SCADA components. The mechanism was implemented as an SDN-based application for SCADA systems that uses multipath routing, which relies on SDN features to frequently modify communication routes between SCADA devices. Further, we performed an experimental evaluation to verify the impact and performance of the mechanism in the SCADA network.**

## I. Introduction

Electric power grids are undergoing an intense modernization process through the use of Information and Communication Technology (ICT), transforming the electric system into Smart Grids [1]. Typically, power plants are complex environments, comprising thousands of devices that assist in the monitoring and control of resources, which rely on automated processes for the operation of the grid. In this context, Supervisory Control and Data Acquisition (SCADA) systems are widely distributed systems used in the management and monitoring of automated processes and components, *e.g.*, substations and field devices, in the electrical grid [2].

SCADA systems require technologies that facilitate resource management and allow the monitoring of the proper operation of communication networks [3]. In particular, Software-Defined Networking (SDN) is as a promising approach that can assist in the modernization of SCADA communication networks [4]. Some preliminary research efforts have advocated the use of SDN in SCADA [3], [5]. SDN offers an architecture that can facilitate the management and configuration of network devices. An SDN architecture can simplify network operation and optimize its performance compared to traditional management techniques, since network programmers are provided with a comprehensive view and direct control of the network, through a centralized controller device [6].

The purpose of this paper is twofold: *(1)* to investigate the advantages of using SDN in SCADA systems, and *(2)* to demonstrate a concrete case-study of an SDN application that can be used to increase privacy in SCADA. Initially, we discuss the possible benefits that can be achieved through the adoption of SDN into SCADA systems, such as simplified configuration of devices and better management of power system resources. Also, SDN characteristics can assist in the growth of the power system network infrastructure, facilitating the creation of new network applications that previously, with traditional architectures, were more complex to be implemented.

Furthermore, to illustrate the benefits of the use of SDN in SCADA, this paper also presents a case-study scenario describing a mechanism to enhance the privacy of information that is carried over SCADA networks. Our solution aims to prevent a possible eavesdropper in the network from fully capturing communication flows between SCADA components. To do this, we present an SDN-based network application for SCADA systems that uses *multipath routing*, which relies on SDN features to frequently modify communication routes between SCADA devices. This allows packet exchange between two end-devices in a SCADA network to be performed through more than one communication route. Further, evaluation results are presented, which measure the impact and performance of the implemented mechanism.

This paper is organized as follows: Section II presents some background about SCADA systems and SDN, and discusses the benefits of using SDN in SCADA systems. Section III describes a case-study scenario for the use of SDN in SCADA and the multipath routing strategy. Section IV presents the evaluation results and a performance analysis of our mechanism. Section V describes the related work. Finally, Section VI concludes the paper.

## II. SDN-Based SCADA Systems

Smart Grids are power distribution networks that depend on an increased level of automated monitoring and control, often exchanging data over IP-based communication protocols [1]. Compared to legacy power systems, Smart Grids rely on bidirectional and high-speed communication technologies to provide more flexible and accurate energy management [7]. Supervisory Control and Data Acquisition (SCADA) systems

978-3-901882-76-0 @2015 IFIP

are considered one of the main components of the power grid, and allow the control, management and acquisition of remote data from equipment and power substations. Due to their increasing complexity, SCADA systems demand techniques to simplify the management of system equipment, to ensure performance requirements, to automate their operation and to offer support for resilience functionality [3].

### A. SCADA Systems

SCADA systems are used in critical infrastructures such as power plants, water supplies, oil and gas facilities. In power plants, in specific, SCADA systems are used to control and monitor essential equipment for energy delivery. These systems comprise distributed components, which are often dispersed around thousands of kilometers and allow the continuous data acquisition that is critical to the functioning of the power grid [2]. These systems are organized in two main types of components: the control center, which includes the MTU (Master Terminal Unit), and substations geographically dispersed. The core of the SCADA system is the MTU. This component gathers information about the system operation and displays it to SCADA operators. Further, the MTU is capable of sending commands to substations to configure field devices in a remote way. Substations comprise a RTU (Remote Terminal Unit), which manages field devices such as sensors and actuators that are responsible for telemetry of automated processes and for the execution of commands sent by the MTU, and transmit data to the MTU. Figure 1 shows a typical SCADA architecture with the control center and its substations.

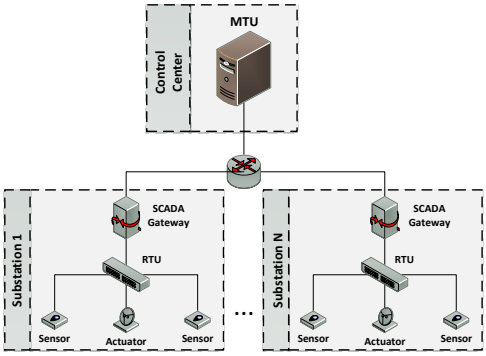



Fig. 1. Typical SCADA architecture.

Due to the increasing number of interconnected devices, sensors and actuators, and also the larger volume of information exchanged between components, SCADA systems are becoming more complex. In their majority, components of the SCADA system communicate through protocols originally developed for process automation, which have been ported to operate over the TCP/IP stack [8], *e.g.*, MODBUS TCP/IP [9], DNP3 over TCP/IP [10] and Ethernet/IP [11]. Further, modern SCADA systems are connected directly or indirectly to the Internet. Consequently, SCADA systems are susceptible to threats such as malwares and cyber-attacks. Therefore, a SCADA system must take into consideration aspects of system security, like timeliness, availability, integrity of data and components, and confidentiality [2]. Such systems require the ability to flexibly manage and configure a growing number of components and to monitor data flows across their communication networks, in order to prevent cyber-attacks, intrusions or malware from compromising the system operation, since the malfunctioning of the grid can result in major disasters. Thus, we aim to investigate the use of network management techniques in general, and SDN in particular, to assist in the management of SCADA communication networks.

### B. SDN and OpenFlow

Software-Defined Networking (SDN) is an emerging architecture for managing, monitoring and controlling switching devices and network traffic [4], [6]. SDN decouples the network control and the forwarding planes. This can simplify network management, offering to network programmers a comprehensive view of the network and the ability to control network devices from a centralized controller [12]. The SDN architecture consists of the following components: *(i) switches:* data forwarding devices that use a flow table to forward packets; *(ii) flow table:* a table that contains a list of flow entries and associated actions to be applied to the respective flows; *(iii) controller:* software component that manipulates and controls the flow tables of switches; and *(iv) secure channel:* communication channel that connects each switch to a controller and allows the controller to install flow rules. Figure 2 illustrates the SDN architecture and its components.

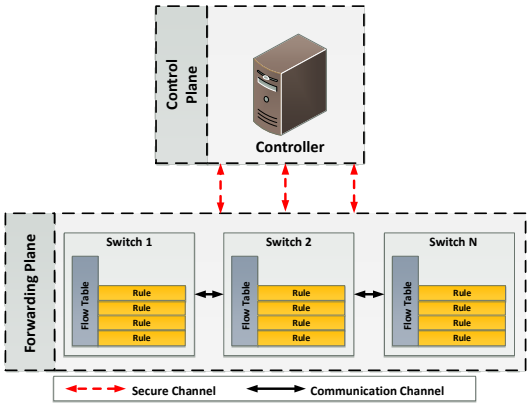



Fig. 2. SDN architecture.

To standardize the communication between the controller and the switches, the OpenFlow protocol has been proposed [13]. OpenFlow defines how applications running on the controller can program the flow table of each network switch. The communication between the controller device and the switches is performed over a secure channel, enabling the controller to manage and control all network switches, and to send and receive control messages to and from the switches.

### C. Discussion: Investigating the Benefits of SDN in SCADA

In this paper we advocate the use of SDN to assist in the management of SCADA systems. SDN can enable more flexible SCADA networks, since the addition of new policies and services requires changing the controller only [5]. Arguably, the use of SDN in SCADA will support more resilient systems, as solutions to mitigate attacks and other threats can be more easily implemented in the controller.

TABLE I. BENEFITS OF SDN-BASED SCADA SYSTEMS FOR FCAPS MANAGEMENT.

| Property | Description |
|---|---|
| **Fault** | SDN enables the implementation of mechanisms for increasing the resilience of SCADA systems. The centralized view of the controller allows more efficient fault detection, isolation of affected components, and remediation of abnormal operation in the SCADA network. |
| **Configuration** | The OpenFlow protocol provides a standard API for the correct configuration of new devices added to the SCADA network and their communication protocols. This can reduce the configuration overhead of these components. |
| **Accounting** | The measurement capabilities of the controller provides the ability to collect metrics and statistics about the network traffic. This information can be used in dimensioning the capacity of the SCADA network, to plan the growth of the power grid, or to detect abuses in resource usage. |
| **Performance** | SDN can facilitate the use of QoS policies in SCADA systems, to perform load balancing between communication links and to optimize the operation of system components. |
| **Security** | The controller also permits the implementation of applications that can add more security to the SCADA system, *e.g.*, in terms of detecting malicious activity or protecting the information exchanged in the SCADA network. To illustrate this, Section III presents an anti-eavesdropping SDN-based application for SCADA. |

SCADA systems can benefit from the characteristics of SDN in several ways, such as:

- **Flexibility:** SDN enables more flexible systems [14], in which applications and protocols can be modified via a centralized controller. In SCADA systems, this will permit easily adding new field devices or upgrading existing applications in the SCADA network.
- **Centralized management:** the centralized control plane offers a global view of the network. Thus, an SDN-based SCADA control center will be able to manage not only field devices, but also monitor and control the network that interconnects system devices.
- **Standard API:** the OpenFlow protocol provides a standard API for controlling network switching devices. In SCADA networks, this standardization will permit a better integration of geographically dispersed equipment from different vendors.
- **Programmability:** via the controller it is possible to easily add new functionality to the network on demand. In SCADA, this will allow creating a range of customized services, *e.g.*, to control the reading frequency of field devices at a specific time of day.

Further, the characteristics of SDN can also enhance FCAPS (*fault*, *configuration*, *accounting*, *performance* and *security*) management in SCADA systems. Table I indicates some of the possible benefits of SDN-based SCADA systems for each FCAPS properties.

## III. ANTI-EAVESDROPPING IN SDN-BASED SCADA

This section presents our *multipath routing* strategy for SDN-based SCADA systems, and how it can be used to improve privacy in these systems. Firstly we present a case study scenario as a motivation for developing network applications that improve privacy in SCADA. Then we describe our multipath routing strategy to SCADA networks, using SDN.

### A. Case-Study Scenario

Consider a SCADA system responsible for controlling the electrical grid of a particular region, where a central control station monitors and manages multiple substations. The network topology of this SCADA system contains redundant communication routes, which allow, in case of a communication link breakdown, the exchange of messages between system components through an alternative path. In this paper we assume that the communication network connecting the SCADA components can be implemented using an SDN network. All components of the SCADA system, control center and substations, communicate through a high-speed wired SDN network, using a legacy communication protocol. The protocol adopted was ported to run over the TCP/IP stack and does not provide a secure communication between system devices, *i.e.*, communication is not encrypted, which allows a person without permission to eavesdrop the messages that travel in the SCADA network.

Eavesdropping is a network layer attack that consists in the interception of packets that travel over the network, with the intention of collecting confidential information. Unencrypted and weakly encrypted information exchange allow an individual attacker to intercept data transmitted over the network if he or she has access to the communication medium. In other words, an eavesdropper can obtain passwords, view the content of message exchanges and confidential information if the eavesdropper can access the local network.

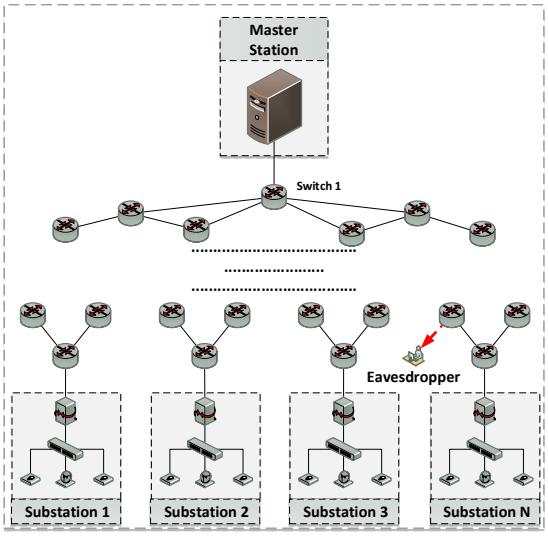



Fig. 3. Case study scenario.

SCADA systems, largely, use insecure and unencrypted communication networks [15]. In this context, through the placement of listening devices well positioned in the network, an eavesdropper can easily, for example, capture instructions forwarded from an MTU to sub-MTUs, RTUs, or even relevant information from sensors and actuators in the system [16]. Moreover, an eavesdropper can also collect the end-devices IP address and the access credentials of the SCADA system. If the IP address of the SCADA server is known by an attacker, it can be easily taken down or shutdown using a traditional

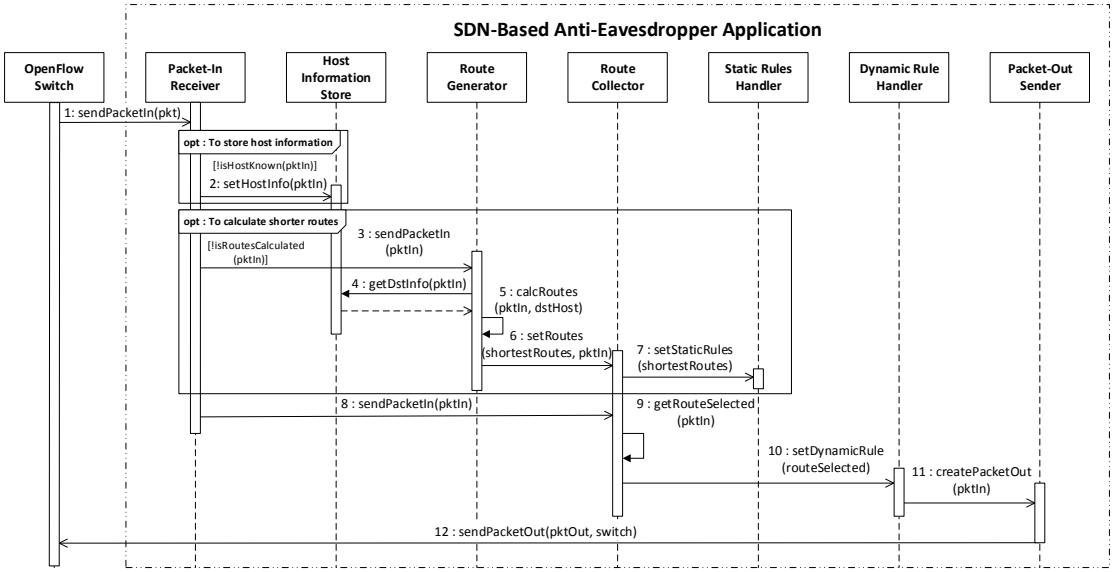



Fig. 4. Sequence diagram of the multipath routing algorithm.

Denial of Service (DoS) attack [17]. Finally, with the access credentials of a system, a person can control substations, and steal corporate data and delete system files [17]. Figure 3 gives an overview of the scenario presented in this case study.

### *B. SDN-Based Anti-Eavesdropping Approach*

Most routing algorithms used nowadays allow communication between devices through a single path for a long period of time [18]. In case a listening device is placed in this path, a large number of messages may be intercepted. This may facilitate message decryption if cryptography has been used. Furthermore, some attacks perform traffic analysis in communication patterns over encrypted connections, which decrease the effectiveness of cryptography techniques [19]. A communication network can be more efficient and robust if it has one or more extra paths for information flows, thus increasing resilience, security, fault tolerance and load balancing [20]. The technique of *multipath routing* was first proposed in the 1970's, and since then it has been used for different purposes in different types of networks [21].

In this paper, we present an SDN-based mechanism that can thwart eavesdropping attacks. Our mechanism uses the facilities provided by SDN to aid SCADA networks in the defense against unauthorized interception of flows by dispersing traffic across multiple paths. Thus, each route transmits only a portion of the packets exchanged during communication. The SDN controller knows the switches *a priori*, but identifies the end-hosts on demand. It also takes advantage of redundant network connectivity, allowing a source device to use multiple routes to communicate with a target device.

Considering the topology illustrated in Figure 3, and that the *master station* starts a continuous communication flow with a specific *substation N*, the proposed algorithm works as follows (each step below is depicted in the diagram in Figure 4). When the first data packet of a flow is received by the first switch (switch 1, in Figure 3), the switch will send a `Packet-In` message to the controller (step 1). If the *master station* is not known to the OpenFlow controller, information about this host (*master station*) will be stored, including its IP address, MAC address and the port number of the switch in which it is connected (step 2). Next, the algorithm calculates the $N$ shortest routes between the *master station* and the specific *substation*, if these routes have not been calculated yet (step 3). To calculate the $N$ shortest routes, information about the destination host is retrieved (step 4). Using the information retrieved from the source and destination hosts, Dijkstra's algorithm [22] is used to calculate the $N$ shortest routes (step 5), in $N$ stages. Considering $N = 2$, in the *first stage*, Dijkstra's algorithm identifies the shortest route between the two network devices, and subsequently all link costs have their weight increased by a tenfold factor. Immediately after, in the *second stage* (and with the link costs increased), Dijkstra's algorithm is executed again to return the second shortest route. Finally, also in the second stage, the link costs of the first route are reestablished to the original values. As explained later, the $N$ shortest routes will be used to deliver a communication flow using different paths and, for this reason, they are stored to be used afterwards (step 6).

Our strategy also relies on the use of *timers* specified by OpenFlow. Using the `Hard TimeOut` timer, which is represented in seconds, we define two types of rules to realize the multipath routing technique: *dynamic rules* and *static rules*. On the one hand, dynamic rules are defined with a low value for `Hard TimeOut`, allowing this kind of rule to expire often. On the other hand, *static rules* do not expire over time, thus they do not need to be reinstalled again on switches. Therefore, after storing the $N$ shortest routes between two hosts, the algorithm will immediately install the static rules on the switches that belong to the $N$ paths (step 7), except on the switches that splits the $N$ shortest routes chosen for communication (which were calculated above).

After installing the static rules, the algorithm retrieves information about the $N$ shortest routes (step 9). Route selection is performed via an internal flag, which allows the

alternation between routes. For example, considering only two paths ($N = 2$), if a flow is transmitted on the first route, when the dynamic rules expire and are reinstalled, the flow will be transmitted on the second route, and *vice versa*. To achieve this, the algorithm must install dynamic rules only on the switch that splits the $N$ routes (step 10 – and switch 1 in Figure 3). Dynamic rules expire according to the value of the `Hard TimeOut` timer. For example, if the timer is set to 5 seconds, dynamic rules will expire and will be reinstalled every 5 seconds. Finally, with the information from the `Packet-In` message, the algorithm generates a `Packet-Out` message (step 11) and sends it to the switch that initiated the interaction with the controller (step 12).

If the controller receives again a `Packet-In` message indicating that the *master station* wants to restart the communication with the same *substation*, the controller will install only dynamic rules on the switch that splits the $N$ routes. In this case, according to the diagram in Figure 4, after receiving a `Packet-In` message (step 1), the algorithm will only select the desired route (step 9) to install the dynamic rules on the corresponding switch (step 10), generate a `Packet-Out` message (step 11) and send it to the switch that requested the interaction (step 12). The pseudocode for the multipath routing strategy described above is illustrated in Algorithm 1. As discussed in the next section, this mechanism is able to prevent an eavesdropper from capturing entire communication flows between the master station and specific substations.

**Algorithm 1** SDN-Based Anti-Eavesdropper PseudoCode

```
1:  procedure MULTIPATH(pktIn, switch)
2:
3:      if (!isHostKnown(pktIn)) then
4:          setHostInfo(pktIn)
5:
6:      if (!isRoutesCalculated(pktIn)) then
7:          dstHost ← getDstInfo(pktIn)
8:          shortestRoutes ← calcRoutes(pktIn, dstHost)
9:          setRoutes(shortestRoutes, pktIn)
10:         setStaticRules(shortestRoutes)
11:
12:     routeSelected ← getRouteSelected(pktIn)
13:     setDynamicRule(routeSelected)
14:     pktOut ← createPacketOut(pktIn)
15:     sendPacketOut(pktOut, switch)
16:
17:     return None
```

## IV. Prototype and Experimental Results

In this section we outline the prototype implementation and present the experimental setup, including the topology as well as the description of each scenario used in the experiments. Then, we analyze the performance of the proposed solution.

### A. Prototype Overview

A prototype for the SDN-based anti-eavesdropping application was built using the POX OpenFlow controller. Figure 5 depicts the components that comprise this application. These include: `Packet-In Receiver`: component responsible for capturing `Packet-In` messages received by the OpenFlow controller; `Host Information Store`: upon receiving a `Packet-In` message, in case there is no information about a given element in the network, this component stores relevant information for that device; `Route Generator`: component responsible for calculating the $N$ shortest routes between two devices in the network; `Route Collector`: component that stores the routes calculated, and that selects a specific path for communication; `Static Rules Handler`: component that creates the static rules that will be installed in all switches along the $N$ shortest routes between two devices, except in the switch that splits these paths; `Dynamic Rule Handler`: component that defines the dynamic rules that will be installed in the switch that splits the communication routes between two devices; `Packet-Out Sender`: after completing the process of route definition, this component sends a `Packet-Out` message to the switch that sent the request to the controller.

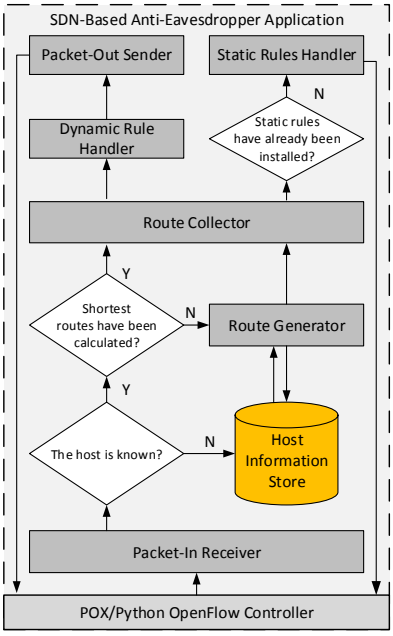



Fig. 5. Anti-eavesdropping application.

### B. Experimental Setup

The scenarios used in the performance analysis of our prototype consider a network topology based on studies of the power grid in countries like USA [23] and Italy [24]. Our network topology contains redundant communication paths, *i.e.*, different paths that lead to the same destination. The network topology consists of 10 switching devices and a number of hosts, which are responsible for simulating the behavior of SCADA system components. The topology was created using Mininet [25]. Mininet is a network emulator that enables the creation of virtual SDN/OpenFlow networks, including virtual hosts, switches, controllers, and links. The switches in the topology used in our experiments were numbered from 1 to 10. Furthermore, there is a *master station* directly connected to switch 1 and one power *substation* directly connected to each one of the nine remaining switch devices. Figure 6 illustrates the configuration of the network topology used in our experiments.

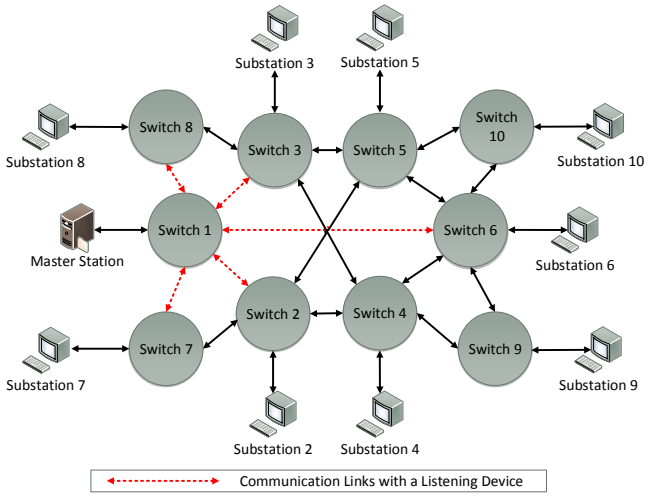



Fig. 6. Configuration of the network topology used in the experiments.

Our experiments consisted of all nine substations sending data simultaneously to the master station in the SCADA network. Each scenario runs for 600 seconds. The communication protocol chosen for message exchange was MODBUS TCP/IP [9]. The substations forward data packets (512 bytes) every 15 seconds, containing information from their respective sensors. This has been carefully chosen to simulate the behavior of a SCADA network, where the substations send periodic information to the master station. The speed of the communication links was set to 10 Gb/s. The initial value of all link costs was defined as 1, which is the default value. Finally, we introduced traffic listeners on 5 communication links that connect switch 1 (which is directly connected to the master station) to switches 2, 3, 6, 7 and 8. These listening devices simulate the behavior of an eavesdropper, and cover all possibilities of communication with the master station.

Further, we defined five scenarios (A, B, C, D and E) to evaluate the performance of our application. The first scenario (A) has an OpenFlow controller with POX default behavior, using the Spanning Tree algorithm [26] for unicast routing with only one communication path between devices. The remaining scenarios (B, C, D and E) use our multipath application, but flows are defined with different values of `Hard TimeOut` timer. This is used to determine how long a flow will follow a particular route before the dynamic rules expire. The value of `Hard TimeOut` in scenarios B, C, D and E is respectively 5, 10, 15 and 20 seconds. In these experiments, the scenarios that use multipath routing were configured to operate with two communication routes ($N = 2$).

### C. Evaluation Results

Firstly, we analyzed the routes chosen by the multipath strategy when two specific SCADA components communicate, namely the substation connected to switch 10 and the master station. Figure 7 presents the two best routes selected by the application during the experiments. In scenarios B, C, D and E, the first route selected was the one with the lowest cost, containing only 3 hops, which is presented as *First Route*. Further, in all scenarios, after increasing the cost of the links used in the first route, the second route chosen had 4 hops, presented as *Second Route* in Figure 7.

In order to observe the effects of choosing a given value

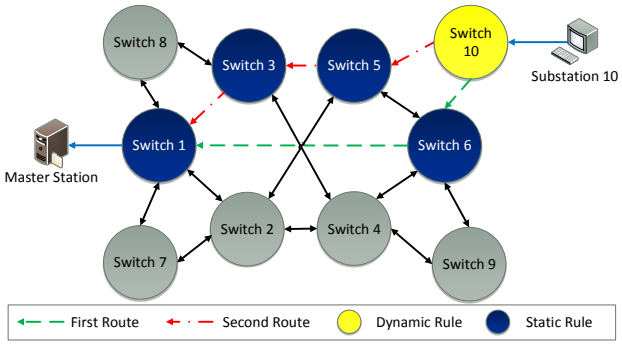



Fig. 7. Anti-eavesdropping communication between components.

for `Hard TimeOut` timer, and conduct a performance comparison between our solution and the default behavior of POX, we defined a set of *metrics*. Initially, we consider the *(i)* total number of flow rules installed at a given moment. We also measure the *(ii)* percentage of packet loss and the *(iii)* amount of `Packet-in` messages received by the controller in each scenario. Further we present the *(iv)* traffic rate in the secure channel. Furthermore, we analyzed the *(v)* amount of exposed communication among each substation and the master station in each scenario. The experiments for each scenario were performed 30 times with a confidence level of 95%.

We compared the number of rules installed in switches at a given time both using POX default behavior (scenario A) and a scenario using the proposed multipath strategy (scenario C)[1]. Figure 8 presents the number of rules necessary to accomplish the communication between *substation 10* and the *master station* in scenarios A and C. Note that POX default solution installs multiple rules simultaneously, reaching a peak of 36 rules after 20$s$). However, the multipath strategy maintains a stable number of rules, ranging between 7 and 8 rules. This is due to the lifetime of dynamic rules, which expire often. By analyzing the controller default behavior we noticed that it installs a rule for each type of flow between two devices, *e.g.*, one rule for ARP flows and another for TCP. This has impacted considerably the number of rules in scenario A.

We also analyzed the TCP packet loss in each scenario, which is depicted in Figure 9. The results indicate that the default solution presented lower packet loss, on average 0,5%, thus requiring fewer retransmissions. However, scenarios B, C, D and E presented slightly higher packet loss, respectively 3.1%, 2.7%, 1.3% and 1.1%. Despite that, the measured rate of retransmissions due to packet loss is still considered acceptable. We noticed that most packet retransmissions for scenarios using our multipath application occurred after switching between communication paths.

Further, we measured the amount of `Packet-In` messages received by the controller in each scenario. Compared to the default behavior, the multipath strategy obtained better results, sending less `Packet-ins` to the controller. These results are presented in Figure 10, where 134 `Packet-ins` were received by the controller in scenario A, 95 in scenario B, 97 in scenario C, 94 in scenario D, and 95 in scenario E. This indicates that the multipath application caused less

[1]Although we performed a similar analysis with the other multipath scenarios, these are not shown here due to space constraints.

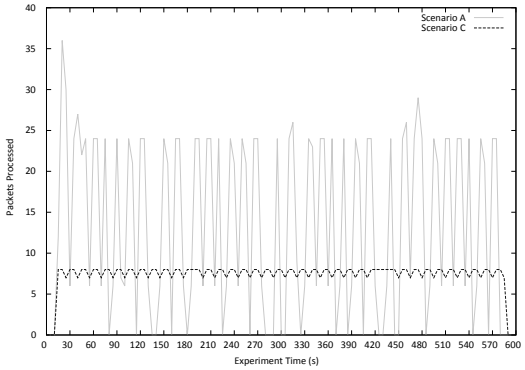



Fig. 8. Number of rules installed during the experiments.

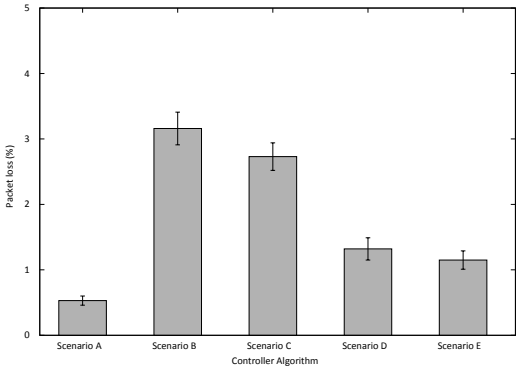



Fig. 9. Packet loss in each scenario.

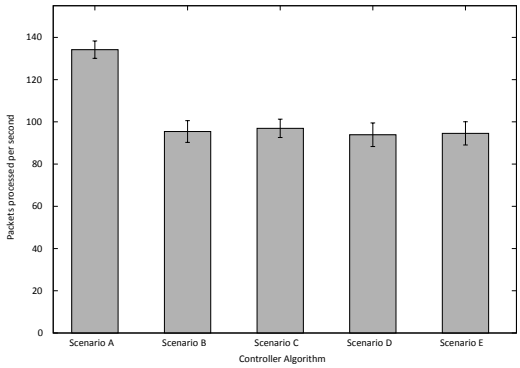



Fig. 10. Number of packets processed by the controller.

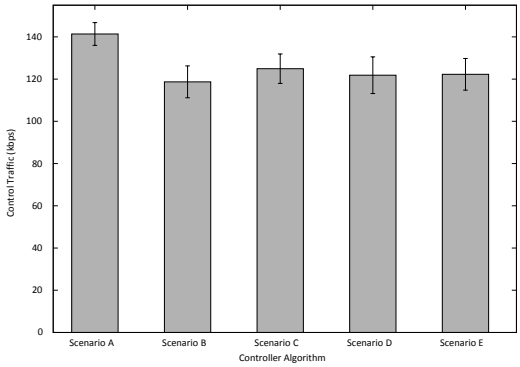



Fig. 11. Traffic generated in the secure communication channel.

processing overhead, compared to the default behavior of POX. However, no significant variation was observed if we consider the lifetime of dynamic rules.

Figure 11 depicts the amount of generated traffic in the secure channel in each scenario. With the POX default behavior, the communication rate between controller and switches was higher, generating up to 138 *kbps* in the secure communication channel. The scenarios using the multipath strategy, in general, generated less traffic between switches and the controller. In particular, the traffic generated in the secure communication channel was 115 *kbps* in scenario B, 122 *kbps* in scenario C, 119 *kbps* in scenario D, and 119 *kbps* in scenario E.

In order to evaluate the ability of preventing an eavesdropper from capturing communication flows, we instantiated listening devices in all five direct communication links to switch 1, which is directly connected to the master station. These listening devices aim to simulate the behavior of an eavesdropper, who positioned himself in privileged points of the SCADA network. We analyzed the amount of packets that were intercepted for each existing communication flow. The end result was similar for all scenarios that use the multipath application. However, using POX default solution, which relies on a single path to accomplish communication, all the information exchanged between the substation and the master station has been exposed. For example, in scenario A it was possible to capture all the information exchanged between substation 7 and the master station, by intercepting the data packets that arrived through switch 7. Instead, using the multipath strategy, an attacker positioned in the same point in the network could intercept only 25% of packets exchanged between substation 2 and the master station, and 75% of packets exchanged between substation 7 and the master station. This is because more than one route is configured to send TCP packets, however all TCP acks are received through one path, *i.e.*, the first shortest route. Table II details the amount of exposed communication in each scenario.

TABLE II. AMOUNT OF COMMUNICATION EXPOSED AMONG SUBSTATIONS AND THE MASTER STATION.

| **Communication Link / Communication Exposition** | **POX default behavior (Spanning Tree Algorithm)** | **Multipath solution and $N = 2$** |
|---|---|---|
| **Switch 2 to Switch 1** | Substation 2 = 100%<br>Substation 4 = 100% | Substation 2 = 75%<br>Substation 4 = 75%<br>Substation 5 = 25%<br>Substation 6 = 25%<br>Substation 7 = 25%<br>Substation 9 = 25% |
| **Switch 3 to Switch 1** | Substation 3 = 100%<br>Substation 5 = 100% | Substation 3 = 75%<br>Substation 4 = 25%<br>Substation 5 = 75%<br>Substation 8 = 25%<br>Substation 10 = 25% |
| **Switch 6 to Switch 1** | Substation 6 = 100%<br>Substation 9 = 100%<br>Substation 10 = 100% | Substation 6 = 75%<br>Substation 9 = 75%<br>Substation 10 = 75% |
| **Switch 7 to Switch 1** | Substation 7 = 100% | Substation 2 = 25%<br>Substation 7 = 75% |
| **Switch 8 to Switch 1** | Substation 8 = 100% | Substation 3 = 25%<br>Substation 8 = 75% |

### *D. Discussion*

With respect to performance, the proposed multipath application generates a lower workload to the controller when compared to the default behavior, which performs routing by a single path. The packet loss of the multipath strategy can be even lower by increasing the lifetime of dynamic rules. However, increasing the lifetime of dynamic rules allows an eavesdropper to intercept more communication.

Further, as discussed in the previous section, even if the eavesdropper is well positioned in a specific point of the network, it cannot intercept an entire communication between two devices. As shown in Table II, if the routing is done via two paths ($N = 2$), in the worst case, the eavesdropper will intercept no more than 75% of a communication flow. The maximum level of exposure can be minimized if the topology has more redundant paths. The exposure level of a communication which uses our scheme can be calculated as: $Exposure = (50 + 50/N)/100$, where $N$ is the number of paths. For example, if we choose 5 paths for routing ($N = 5$), the maximum level of exposure will be 60%.

## V. Related work

In this section we present research efforts that are related to our work. In Section V-A we review some work that use SDN in Smart Grids. Section V-B presents studies that aim to ensure grid connectivity with multipath routing. Finally, Section V-C presents research efforts based on network traffic analysis in SCADA systems.

### A. SDN in Smart Grids

Research efforts investigating the use of SDN in Smart Grid communication networks are still scarce. Cahn *et al*. [3] discuss how SDN can alleviate some of the current problems in Smart Grid communication networks. The authors present the design and development of a new architecture for communication with grid substations, allowing the network to be auto-configurable, secure and reliable against possible system misconfigurations, through the use of SDN. The SDN-based architecture was called Software-Defined Energy Communication Network (SDECN), and a prototype was developed using the Ryu OpenFlow controller and evaluated in a testbed with real IEDs (Intelligent Electronic Devices). Further, Goodney *et al*. [5] propose the use of SDN to control the communication between devices responsible for measuring electrical waves in the grid, known as PMUs (Phasor Measurement Units). The authors developed an SDN-based network application to facilitate the management of PMUs and provide support for essential features, such as multicast and multi-rate.

### B. Multipath Routing in Smart Grids

Differently from our proposal, which alternates the information flow between multiple paths, Hong *et al*. [27] investigate how to transmit duplicate information using multiple communication routes in Smart Grids. In particular, the authors present two multipath routing algorithms, specifically developed for Smart Grids. These algorithms aim to solve the min-max non-disrupting $k$-path computation problem ($M^2$NKPCP), in which two routing paths share switches and a possible failure of a specific equipment can disable an entire communication flow. The algorithms calculate totally disjoint routes, and they differ by the trade-off between running time and quality of the output. Also, Vaidya *et. al* [28] focus on other part within the Smart Grid, by using multipath routing more specific in the AMI (Advanced Metering Infrastructure). AMI is responsible for the automatic measurement, management and analysis of energy consumption and distribution to end-users. The study aims to mitigate the problems of security mechanisms in routing protocols in wireless ad hoc networks, through the adoption of multipath routing in wireless mesh AMI networks.

### C. Traffic Analysis in SCADA Systems

Barbosa *et al*. [29] investigate the main characteristics of network traffic in SCADA systems. The study looks into the similarity between SCADA traffic and SNMP traffic. The authors analyze nine different datasets, of which six are SNMP traces and three are SCADA traces. From the results, the study concludes that SCADA traffic and SNMP traffic are similar in the sense that devices generate information flows in a periodical fashion. Further, Cheung *et al*. [30] propose an IDS (Intrusion Detection System) based on behavioral models for SCADA networks. This IDS creates models that represent the expected network behavior of the devices that are connected to a SCADA system. The authors point out that SCADA systems have topologies that hardly change over time, and thus the behavior of the devices maintains a pattern. This facilitates the detection of possible attacks that may cause changes to the expected network behavior. Finally, Barbosa [31] presents an IDS that can detect data injection and DoS attacks. This IDS explores the traffic periodicity in SCADA systems.

## VI. Conclusion and Future Work

Power grids are responsible for the transmission and distribution of electricity to end-users. However, over the recent years, power grids are becoming more sophisticated, with the aim of increasing their safety, reliability, economical and energy efficiency, and reducing their environmental impact. To assist in the modernization process of electric power grids, we are investigating the use of SDN in SCADA systems. In this context, SDN-based SCADA systems can facilitate the design and development of Smart Grid network applications, by making them more robust and flexible. Also, we presented a concrete case-study of an SDN-based application for multipath routing to increase the privacy of the information that is carried over SCADA networks, and make it more difficult for an eavesdropper to capture communication flows between SCADA devices. The multipath routing mechanism is based on the use of dynamic and static flow rules. We acknowledge that the work presented in this paper has applicability beyond the prevention of eavesdropping. Although we chose to limit the scope of the paper to a single case study, other uses could include load balancing and resilient routing.

Further, we performed an experimental evaluation to verify the impact and performance of the mechanism in the SCADA network. We found that dynamic rules with a shorter lifetime make it more difficult for an eavesdropper to intercept the communication, but a longer lifetime may be advantageous for large-scale SCADA systems, because this reduces the management overhead in the controller. As future work, in order to avoid the min-max non-disrupting $k$-path computation problem ($M^2$NKPCP) [27], we intend to refine the algorithm and permit the selection of completely disjoint routes.

## Acknowledgement

This work is supported by ProSeG - Information Security, Protection and Resilience in Smart Grids, a research project funded by MCTI/CNPq/CT-ENERG # 33/2013.

## REFERENCES


[1] H. Farhangi, "The path of the smart grid," *Power and Energy Magazine, IEEE*, vol. 8, no. 1, pp. 18–28, January 2010.

[2] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST special publication*, pp. 800–82, 2011.

[3] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, "Software-defined energy communication networks: From substation automation to future smart grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 558–563.

[4] J. Wickboldt, W. De Jesus, P. Isolani, C. Bonato Both, J. Rochol, and L. Zambenedetti Granville, "Software-defined networking: management requirements and challenges," *Communications Magazine, IEEE*, vol. 53, no. 1, pp. 278–285, January 2015.

[5] A. Goodney, S. Kumar, A. Ravi, and Y. Cho, "Efficient pmu networking with software defined networks," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 378–383.

[6] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software-defined networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482629

[7] W. Wang and Z. Lu, "Survey cyber security in the smart grid: Survey and challenges," *Comput. Netw.*, vol. 57, no. 5, pp. 1344–1371, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.12.017

[8] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *Computers & Security*, vol. 25, no. 7, pp. 498 – 506, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404806000514

[9] A. Swales, "Open modbus/tcp specification," *Schneider Electric*, vol. 29, 1999.

[10] X. Lu, Z. Lu, W. Wang, and J. Ma, "On network performance evaluation toward the smart grid: A case study of dnp3 over tcp/ip," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011, pp. 1–6.

[11] P. Brooks, "Ethernet/ip-industrial protocol," in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, vol. 2, Oct 2001, pp. 505–514 vol.2.

[12] S. Shah, J. Faiz, M. Farooq, A. Shafi, and S. Mehdi, "An architectural evaluation of sdn controllers," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 3504–3508.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[14] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn," *Queue*, vol. 11, no. 12, pp. 20:20–20:40, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2559899.2560327

[15] Y. Mo, T.-H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber-physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, Jan 2012.

[16] E. Chikuni and M. Dondo, "Investigating the security of electrical power systems scada," in *AFRICON 2007*, Sept 2007, pp. 1–7.

[17] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, "Cyber security and privacy issues in smart grids," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 981–997, Fourth 2012.

[18] W. Lou and Y. Fang, "A multipath routing approach for secure data delivery," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 2. IEEE, 2001, pp. 1467–1473.

[19] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 10–29.

[20] J. He and J. Rexford, "Toward internet-wide multipath routing," *Network, IEEE*, vol. 22, no. 2, pp. 16–21, 2008.

[21] N. F. Maxemchuk, "Dispersity routing," in *Proceedings of ICC*, vol. 75. Citeseer, 1975, pp. 41–10.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] T. Overbye and J. Weber, "Visualizing the electric grid," *Spectrum, IEEE*, vol. 38, no. 2, pp. 52–58, Feb 2001.

[24] A. E. Motter, S. A. Myers, M. Anghel, and T. Nishikawa, "Spontaneous synchrony in power-grid networks," *Nature Physics*, vol. 9, no. 3, pp. 191–197, 2013.

[25] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[26] J. P. Russell, D. M. Goodman, C. D. Murton, C. T. W. Ramsden, and J. Shields, "Spanning tree algorithm," Apr. 16 2002, uS Patent 6,373,826.

[27] Y. Hong, D. Kim, D. Li, L. Guo, J. Son, and A. O. Tokuta, "Two new multi-path routing algorithms for fault-tolerant communications in smart grid," *Ad Hoc Networks*, vol. 22, no. 0, pp. 3 – 12, 2014, special Issue on Routing in Smart Grid Communication Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514001012

[28] B. Vaidya, D. Makrakis, and H. Mouftah, "Secure multipath routing for ami network in smart grid," in *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, Dec 2012, pp. 408–415.

[29] R. Barbosa, R. Sadre, and A. Pras, "A first look into scada network traffic," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 518–521.

[30] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA Security Scientific Symposium*, vol. 46, 2007, pp. 1–12.

[31] R. R. R. Barbosa, "Anomaly detection in scada systems: a network based approach," Ph.D. dissertation, University of Twente, Enschede, April 2014. [Online]. Available: http://doc.utwente.nl/90271/

## AppendixB PUBLISHED PAPER – COMPSAC 2016

This paper presents a NIDS for SCADA systems based on machine learning algorithms. Our solution relied on SDN for capturing information about communication flows of a SCADA network. Moreover, our detection mechanism uses OCC algorithms. These algorithms only use samples of the normal system behavior for detecting cyber-attacks against SCADA systems, *i.e.*, they do not need samples of specific attack classes. This characteristic of OCC algorithms is ideal for developing specific SCADA NIDSes, because these environments are automated and have a traffic profile without many variations, and with stable connection matrices. Furthermore, are still scarce the attack traces publicly available. Our prototype used OCC algorithms based on SVMs that can quickly classify large datasets. Finally, the NIDS obtained good results in our experimental evaluation, achieving approximately 98% of accuracy, proving to be efficient in the detection of attacks targeted to the power grid.

- **Title –**
  *A One-Class NIDS for SDN-Based SCADA Systems*
- **Conference –**
  The 40th IEEE Computer Society International Conference on Computers, Software & Applications (COMPSAC-2016)
- **Type –**
  Main track (full-paper)
- **Qualis –**
  A2
- **URL –**
  <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7552026>
- **Date –**
  June 10-14, 2016
- **Held at –**
  Atlanta, USA
- **Digital Object Identifier (DOI) –**
  <http://http://dx.doi.org/10.1109/COMPSAC.2016.32>

2016 IEEE 40th Annual Computer Software and Applications Conference

# A One-Class NIDS for SDN-Based SCADA Systems

Eduardo Germano da Silva*, Anderson Santos da Silva*, Juliano Araujo Wickboldt*,
Paul Smith†, Lisandro Zambenedetti Granville*, Alberto Schaeffer-Filho*

*Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
Email: {eduardo.germano, assilva, jwickboldt, granville, alberto}@inf.ufrgs.br
†Safety and Security Department, Austrian Institute of Technology, Vienna, Austria
Email: paul.smith@ait.ac.at

***Abstract*— Power systems are undergoing an intense process of modernization, and becoming highly dependent on networked systems used to monitor and manage system components. These so-called Smart Grids comprise energy generation, transmission, and distribution subsystems, which are monitored and managed by Supervisory Control and Data Acquisition (SCADA) systems. In this paper, we discuss the benefits of using Software-Defined Networking (SDN) to assist in the deployment of next generation SCADA systems. We also present a specific Network-Based Intrusion Detection System (NIDS) for SDN-based SCADA systems, which uses SDN to capture network information and is responsible for monitoring the communication between power grid components. Our approach relies on SDN to periodically gather statistics from network devices, which are then processed by One-Class Classification (OCC) algorithms. Given that attack traces in SCADA networks are scarce and not publicly disclosed by utility companies, the main advantage of using OCC algorithms is that they do not depend on known attack signatures to detect possible malicious traffic. Our results indicate that OCC algorithms achieve an approximate accuracy of 98% and can be effectively used to detect cyber-attacks targeted against SCADA systems.**

## I. Introduction

Electric power grids are undergoing a modernization process and evolving into the so-called Smart Grids [1][2], improving the generation, transmission, and distribution of electrical energy. Smart Grids allow a more resilient, secure, and reliable power supply for end-users, such as industries, schools, hospitals, and residences. A power grid is composed of thousands of electronic devices, such as transformers, relays, fuses, or disconnectors. An important component of a power grid is the Supervisory Control and Data Acquisition (SCADA) system [3], responsible for monitoring, controlling, and managing automated processes of the power grid, such as shutting down of an electrical substation, or monitoring the passing electric tension on a transmission line.

Just as power grids are becoming Smart Grids, SCADA systems are also evolving, for example, by using more secure communication protocols and field devices with more processing capacity. Recently, efforts to merge Software-Defined Networking (SDN) with SCADA systems have been carried out [4][5]. SDN is a promising network paradigm that can support the evolution of SCADA communication networks as well [4]. SDN introduces an architecture that simplifies network operations by relying on a logically centralized element often referred as controller [6]. That adds the flexibility required to quickly deploy and configure new field devices and to develop more complex network services in the context of SCADA networks [4].

Given the importance of power grid infrastructures, they may become target of malware infections and cyber-attacks. If these threats are not properly detected and handled they can cause outages in power supply, or even destroy substation equipment [7]. An Intrusion Detection System (IDS) [8] is then necessary to assist in detecting and mitigating such threats. In this paper, we rely on OpenFlow, currently the most important protocol for SDN implementation [9], to present a Network-based IDS (NIDS) [10] designed specifically for SDN-based SCADA systems. Our NIDS uses One-Class Classification (OCC) algorithms [11] that enable detecting abnormal traffic behavior from a homogeneous training set containing only the signature of traffic generated under normal network operation [11]. There are many aspects that make OCC algorithms ideal for anomaly detection in SCADA networks, which includes: (i) they can detect unknown types of attacks for which there are no signatures available, (ii) they do not require specific SCADA attack traces, which are scarce and often not publicly disclosed by utility companies, and (iii) this class of algorithm is suitable for the kind of traffic behavior and periodicity found in SCADA systems.

To demonstrate the benefits and accuracy of our proposed NIDS, we present an analysis comparing two machine learning algorithms of OCC: OCSVM [12] and SVDD [13]. This comparison shows the efficiency of our approach to detect cyber-attacks targeted at a power grid. In our experiments, we simulated an SDN-based SCADA system using a large-scale topology, with one main control center, four intermediate control centers, eight distribution substations, and hundreds of field devices. SDN enables the SCADA system to control and monitor field devices and the network that interconnects SCADA components. The SCADA system

0730-3157/16 $31.00 © 2016 IEEE
DOI 10.1109/COMPSAC.2016.32

uses Modbus/TCP protocol and, during the experiments, we simulated an attacker that starts a DoS attack targeted at one substation. The cyber-attack exploits a Modbus vulnerability that allows flooding unauthorized read requests to SCADA devices operating under this communication protocol [14].

The remainder of this paper is organized as follows. In Section II, we present background on SDN-based SCADA systems, as well as fundamental concepts related to One-Class Classification algorithms. In Section III, we describe the design of our proposed NIDS and details of the algorithms implemented. In Section IV, we present the evaluation results and a performance analysis of our approach. In Section V, we describe the related work and finally, in Section VI, we conclude this paper with final remarks and future work.

## II. Background

In this section, we present the fundamental aspects about SCADA systems and the communication protocols used in this kind of cyber-physical system. In addition, we present how SDN can assist in the modernization process of SCADA systems and power grids. Furthermore, some concepts of OCC are discussed as well as their advantages and benefits in the detection of network anomalies in SCADA systems.

### A. SDN-Based SCADA Systems

SCADA systems are used in critical infrastructures such as power grids, water supplies, oil and gas facilities. In power grids, in specific, SCADA systems are highly distributed, used by power utilities to collect data, monitor, and control devices through power lines [3]. SCADA systems present a well-defined architecture with two main types of components: a single master unit, called Master Terminal Unit (MTU); and slave units, or Remote Terminal Units (RTUs) [7]. For large-scale SCADA systems that contain several RTUs, subMTUs are also employed to alleviate the workload on the primary MTU [3]. In the power distribution system, RTUs monitor energy distribution substations and the electrical voltage forwarded to final users [4]. These substations are usually composed of thousands of field devices, such as sensors, circuit breakers, actuators, relays, and transformers [5]. Thus SCADA systems have a large number of interconnected devices that transmit a considerable amount of information about the system actuation environment and the automated process.

To provide a more reliable data transmission mechanism, most of the communication protocols used by current SCADA systems were ported to execute over TCP/IP, *e.g.*, Modbus/TCP and DNP3 over TCP/IP [15]. In these cases, adaptations have been carried out without considering security aspects (*e.g.*, data encryption) allowing an attacker to intercept the communication and read the information being transmitted [4]. Another important issue is that, to allow remote and more flexible maintenance of their components, SCADA systems were typically connected to the corporate network of the organization responsible for the system, that consequently is directly or indirectly connected to the Internet. This tendency makes SCADA systems susceptible to common threats, such as malware and cyber-attacks [15].

There are many efforts to increase the security level of SCADA systems, *e.g.*, proposal of new communication protocols, improvement of existing protocols, new security and management standards, or even IDSes [15][16]. However, the incorporation of SDN into SCADA systems emerges as an attractive research area, since SDN can help in the evolution of SCADA communication networks, facilitating the development of network applications [4]. The adoption of SDN in SCADA will support more resilient systems, as solutions to mitigate attacks and other threats can be more easily implemented in the SDN controller. SCADA systems can benefit form the characteristics of SDN in several ways:

- **Flexibility:** SDN permits easily adding new field devices or upgrading existing network applications inside the SCADA system [5].
- **Centralized Management:** The MTU can not only manage field devices but also monitor and control the network that interconnects system devices [4].
- **Standard API:** The OpenFlow protocol provides a standard API that allows a better integration of geographically disperse network equipment from different vendors [5][17].
- **Programmability:** SDN allows creating a range of customized services, *e.g.*, to perform load balancing between communication links, to optimize the operation of system components, or even to identify and mitigate traffic anomalies [4].

### B. One-Class Classification

Anomaly detection techniques aim to detect unexpected behaviors, also called *anomalies* or *outliers*, in a dataset [18]. In communication networks, unexpected behaviors may represent the occurrence of malicious activities, creating a real necessity for sophisticated detection mechanisms to prevent the network from service degradation. Frequently, machine learning is suitable to this task because it offers a wide range of mechanisms that can be applied for traffic classification and for detecting intrusions in different contexts.

There are different classification techniques associated with machine learning. Supervised machine learning algorithms require a training step, *i.e.*, an initial step in which the classifier learns the profile of the target class. A classifier trained with positive and negative classes is called a binary classifier, while a classifier that is trained with several samples representing many classes is named a multiclass classifier. However, both of these types of classifiers require the profile of the target attack to operate, which may be difficult to be obtained when the profile required is a novel, unknown attack. To alleviate this restriction, OCC

algorithms have been designed such that they only need one data profile, which in the case of network can be the normal expected traffic behavior, a frequently available traffic information.

SCADA systems are environments that can take advantage of OCC algorithms. It is noteworthy that the same benefits of using machine learning in general networks can be achieved in SCADA systems as well. In addition, SCADA traffic flows are naturally periodic and their networks have stable connection matrices [7]. This characteristic encourages the use of OCC algorithms for detecting anomalous behaviors in SCADA networks. These algorithms do not rely on malicious signatures, but instead need only the expected traffic behavior, making the detection process faster and more accurate. Since an OCC-based NIDS does not require attack signatures to build a classifier model, it is well suited for intrusion detection in SCADA systems [19]. Further, SDN allows the periodic gathering of precise statistics in SCADA networks. These statistics can be used to create a model of the SCADA system normal and expected behavior. Thus, this behavioral model in combination with OCC algorithms, is used to build a resilience mechanism for detecting cyber-threats in SCADA networks.

## III. One-Class NIDS for SDN-Based SCADA Systems

In this section, we present a brief background on the OCC algorithms that we adopted in the proposed NIDS for SDN-based SCADA systems. Furthermore, we also introduce our strategy to detect intrusions in SCADA communication networks and detail the architecture and the respective components of our NIDS.

### A. OCC Algorithms

The OpenFlow protocol allows the integration of network devices from different vendors through its standard API [4]. This feature facilitates gathering network flow statistics via a logically centralized controller. These statistics about the SCADA network are essential for the proper functioning of our NIDS. Algorithms are used to analyze these statistics, finding anomalous behaviors in the SCADA network. Furthermore, we also consider fundamental that our approach must:

- Adapt to the scale and heterogeneity of existing SCADA systems. In other words, the proposed NIDS must be able to detect anomalous behaviors in SCADA networks of small power distribution companies and large-scale systems responsible for managing the power grid of entire countries, independently of the protocol used or the behavior of network devices;
- Manipulate constantly, promptly, and accurately large datasets. This requirement is important because the sampling period in SCADA systems in the power distribution sector ranges from 2 to 4 seconds [20]. In addition, a single substation may contain thousands of field devices [5] indirectly (via RTU) or directly connected to the SCADA MTU. Furthermore, a NIDS that has a fast anomaly detection process can avoid or minimize the incidence of outages in the power supply;
- Enable the detection of harmful network anomalies in SCADA systems. Our NIDS must detect anomalous behaviors known by operators as well as anomalies that exploit previously unknown vulnerabilities, and consequently unpatched (Zero-Day exploit). Furthermore, as SCADA attack traces are scarce and not publicly disclosed, the proposed NIDS must be able to detect such anomalies without using attack signatures.

To fulfill these requirements, we adopted OCC algorithms based on Support Vector Machine (SVM), such as One-Class Support Vector Machine (OCSVM) [12] and Support Vector Data Description (SVDD) [13]. SVMs are a set of supervised learning methods that analyze datasets and recognize patterns [21]. SVMs are among the most popular methods of supervised learning. SVM-based algorithms deal well with large datasets and in most cases perform better in comparison to other supervised learning methods [22]. A brief description of OCSVM and SVDD is presented below:

*1) One-Class Support Vector Machine (OCSVM):* It is a supervised machine learning algorithm presented by Schölkopf *et al.* [12]. OCSVM resembles the traditional two-class SVM, where the training set is composed of two groups, one positive and one negative. However, OCSVM, in its training phase, uses a homogeneous set of instances and is indicated for problems that involve anomaly detection. Thus, with data used to fit the algorithm, OCSVM learns a decision function and classifies new data as similar or distinct to the training set. As the traditional SVM for binary classification, OCSVM uses kernel methods for classifying validation samples. In particular, OCSVM can use traditional kernel methods (linear, polynomial, radial basis function - RBF, or sigmoid), or customized kernels defined by the user.

*2) Support Vector Data Description (SVDD):* Also known as Support Vector Domain Description, it was introduced by Tax and Duin [13][23] and is another type of SVM-based OCC algorithm. SVDD is a useful method for novelty detection and has been applied to a variety of applications that need to monitor the rise of novelties. This algorithm uses the training set to define a hypersphere with minimum radius, which is used for binary classify samples of a validation set. The hypersphere of SVDD is modeled to involve the majority of training samples. In the validation stage, new samples that are not inside the hypersphere area are classified as novelties, whereas samples that are inside the hypersphere are considered normal samples.

### B. Architecture Overview

In this paper, differently from previous approaches that present IDSes for traditional SCADA

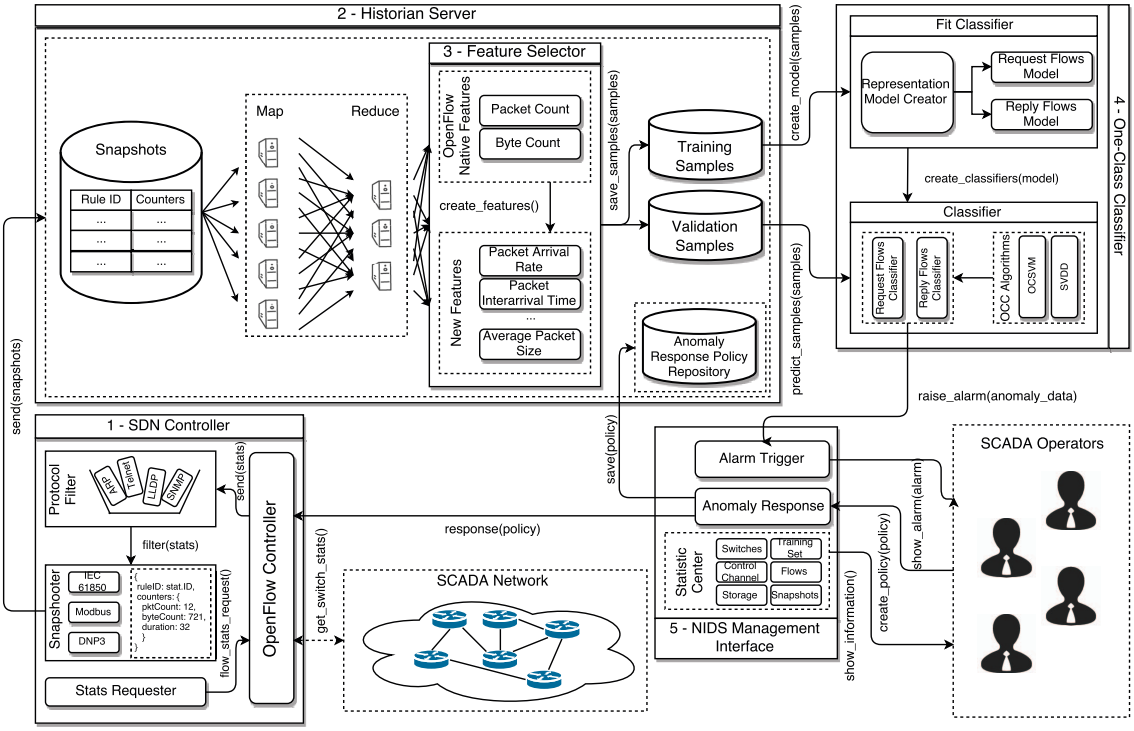



Figure 1. Architecture overview of the proposed NIDS for SDN-Based SCADA Systems.

systems [7][19][24][25], we present a NIDS designed and developed for SDN-based SCADA systems. Our approach takes advantage of the characteristics of SCADA networks to accomplish *novelty* detection in this environment. Considering that SCADA systems have predictable and periodic network traffic and that their networks present a stable connection matrix [7], any novelty can be considered a malicious behavior, such as the beginning of a cyber-attack, or even the misconfiguration of a particular field device. So, as our classifier knows the expected behavior of the SCADA network, any data that indicates novelty in relation to this model can be reported to the operator as a network anomaly. When the entry point of an anomaly in the SCADA network is determined, SDN allows routing rules to be updated in real time. This facilitates anomaly mitigation, enabling the redirection of malicious network traffic, or even dropping the intruder's packets.

The OpenFlow protocol installs rules in each network switch to route data packets. Each rule may have unique information about a communication flow, such as, MAC and IP address of source and destination devices, packet and byte counters, rule duration, and the switch in which the rule is installed. Although there are other approaches for collecting network information, we keep our solution based only on SDN. This allows the proposed NIDS to be extended to create more sophisticated solutions for resilience in SCADA networks. The proposed NIDS uses the OpenFlow protocol for periodically extracting statistics of the SCADA network. This strategy permits continuously gathering statistics on the same frequency of the SCADA sampling period. Our NIDS generates samples from these network statistics. Samples serve as basis to build the model that predicts the network behavior. Each sample has features that are defined by the SCADA operators *a priori*. In addition to the native features provided by OpenFlow (packet count, byte count, and rule duration), our approach also enables the use of other features extracted from native ones. The selection of an optimal set of features can increase classifier accuracy [26], decreasing the rate of false-positive alarms generated during network monitoring.

Our NIDS is composed of five components that intercommunicate to monitor the network and to report possible anomalous behaviors in SCADA systems. Our architecture has been designed to be scalable, *i.e.*, it must allow the detection of anomalies in SCADA systems responsible for monitoring small regions, as well as large-scale systems. Thus, the proposed NIDS must be also capable of promptly processing large amounts of data. For this reason, the architecture contemplates the adoption of mechanisms that enable the parallel and distributed processing of data. Figure 1 presents an overview of the proposed NIDS architecture. A brief description of its components is presented below:

*1)* ***SDN Controller****:* This component is responsible for monitoring and for applying routing strategies to SCADA network switches. It is composed of *Stats Requester*, *OpenFlow Controller*, *Protocol Filter*, and *Snapshooter*. *Stats*

*Requester* is a module responsible for creating request messages of flows statistics and for controlling the periodicity in which these messages are sent to network switches. It generates requests in accordance with the periodicity previously defined by the SCADA operator. As said before, OpenFlow is currently the most important protocol for SDN implementation. The *OpenFlow Controller* executes routing strategies, applies anomaly response policies on the network, and manages statistic requests forwarded to the switches and their respective replies. At this stage, *Protocol Filter* blocks statistics regarding non-specific SCADA protocols, allowing only previously defined types of packets (such as LLDP, ARP, and ICMP, and specific SCADA protocols, such as Modbus, DNP3, and IEC-61850) to be forwarded on the network and generate statistics for the NIDS. Finally, *Snapshooter* is responsible for structuring these statistics of SCADA protocols in snapshots and for forwarding this information to the Historian Server.

*2)* ***Historian Server****:* This is a component that is typically present in the Control Center of several traditional SCADA systems [14]. This server stores logs about SCADA devices. We extended this component to store network snapshots collected by the SDN Controller. It needs to be able to store and process datasets generated by large-scale SCADA systems, and Distributed and parallel processing can help in reducing possible bottlenecks. Thus, in order to allow large-scale data processing, the proposed NIDS relies on the distributed nature of the MapReduce (MR) programming model [27] to implement a scalable version of the Historian Server. MR is used for: (*i*) finding flow rules stored in the Historian Server through the rules header (mapping process); and (*ii*) reducing these headers in keys (reduction process). These keys are fundamental for finding counters of a rule stored in the server. In other words, MR is used to find active rules in the network and catalog their counters. The stored *snapshots* are converted into *Training Samples* (used to fit the classifier model), and *Validation Samples* (analyzed by the NIDS to monitor the SCADA network). To increase its classification accuracy, we split samples in two classes: request and reply samples. Request samples have the MTU as source device, whilst reply samples have an RTU as source device. Further, the Historian Server also has the *Anomaly Response Policy Repository*, which is a component that contains strategies for anomaly mitigation predefined by SCADA operators. Thus, our NIDS can act proactively on the network without direct intervention of an operator, for example: redirecting an anomalous flow to a HoneyPot device; reducing the priority of a harmful flow to the system; or dropping malicious network packets.

*3)* ***Feature Selector****:* OpenFlow only provides native flow features, namely packet count, byte count, and rule duration. These features may not be adequate to describe the nature of a specific traffic profile. Using appropriate features to describe traffic behavior may increase the accuracy of our NIDS. Thus, the *Feature Selector* component [26] analyses the stored samples and offers an extensive set of features extracted from OpenFlow native counters, such as packet inter-arrival-time, packets per second, mean packet length, and so forth. Besides, this component uses feature selection techniques, such as Principal Component Analysis (PCA) and Genetic Algorithm (GA), to determine the optimal set of features for traffic classification. We strategically placed this component inside the Historian Server in order for it to have easy access to the stored traffic samples. Note that, however, we do not incorporate this component in our prototype and preliminary experiments, and this task is proposed as future work.

*4)* ***One-Class Classifier****:* This is the central component in the proposed architecture as it analyzes samples to find anomalous behaviors in the SCADA network. At first, *Fit Classifier* is responsible for the training step of the proposed NIDS. This training step is defined by the SCADA operator and occurs before the validation stage. The SCADA operator also can define if the training stage will occur from trace files, or from the normal functioning of the network. In addition, it is possible to define the periodicity in which our NIDS will be trained again. This component receives training samples for generating classification models: one specific for request flows, and another specialized in reply flows. These classification models are forwarded to the *Classifier* component, which in turn generates two classifiers: Request Classifier and Reply Classifier. Our approach allows SCADA operators to choose and combine OCC algorithms available for traffic classification. In addition, if any validation sample indicates an inconsistent state in the SCADA system, the Classifier will send information about the unexpected behavior to the NIDS Management Interface.

*5)* ***NIDS Management Interface****:* This component provides a management interface for SCADA operators to interface with our NIDS. NIDS Management Interface receives the details of anomalies detected by the One-Class Classifier and generates alarms to the operators. Alarms generated by the *Alarm Trigger* component contain information about where and when the network anomaly occurred in the power system. The *Anomaly Response* component allows the operator to define response policies to the anomalous behavior detected. These policies are stored in the Historian Server, in Anomaly Response Policy Repository, to be reused after. In addition, these policies are directly forwarded to the SDN Controller that applies these actions over the SCADA network. Management policies provided by SCADA operators can be used to redirect a DDoS attack to a HoneyPot device, and block or limit the anomalous traffic through an OpenFlow rule. Finally, *Statistic Center* presents information about the SCADA network and the Historian Server, such as the traffic on the OpenFlow control channel, communication flows disposed in the network, amount of snapshots stored in the database, and even the size of the training set.

## IV. Prototype and Experimental Results

In this section, we describe the proof-of-concept prototype that we implemented, the experimental setup used for the evaluation of the OCC algorithms, as well as the test scenarios used to simulate a realistic SCADA environment. We also discuss the experimental results that validate our prototype and verify the accuracy of the classification techniques.

### A. Prototype and Experimental Setup

We chose the POX SDN/OpenFlow Controller[1], which uses OpenFlow version 1.0 to manage and monitor the SCADA communication network. We implemented the Historian Server on a NoSQL database, more specifically the Apache Cassandra Server[2]. This database promotes design scalability and allows distributing system tasks to multiple clusters, decreasing possible processing bottlenecks. Finally, we used the LIBSVM[3] library that offers a simple and efficient implementation of the necessary machine learning algorithms, OCSVM and SVDD, for our prototype. It is important to note that both algorithms were used with their default parameters. Moreover, we used the RBF Kernel for the OCSVM algorithm.

We defined an evaluation scenario that simulates the SDN-based SCADA system of a particular power grid company. This company controls a hydro-power plant, transmission lines, and eight distribution substations. This power grid is responsible for supplying a particular region. To control and monitor this power grid, the company has a SCADA system based on a large-scale topology, with one MTU, four subMTUs, eight RTUs (each RTU contains 750 field devices directly connected), and eight hundred independent field devices. The independent field devices were simulated to control the voltage in transmission lines and each one was simulated independently. In addition, each independent field device and each RTU were subordinated to one subMTU. The MTU requests data from its subMTUs, that consequently request data from their subordinated devices (RTUs or field devices). We used the TCP version of the most common SCADA protocol, Modbus [28], for communicating devices of our SCADA system. To do that, we used the PyModbus library[4] and its Modbus `READ COILS` function. This function allows a MTU to read values of RTUs registers. Figure 2 depicts the topology that we used in our evaluation scenario. To simulate this environment, we used Mininet[5]. The topology, number of substations, number of field devices and protocols used are based on documents reporting actual SCADA systems deployed in the US [3].

Our experiments consisted of the MTU and subMTUs periodically requesting information from the subordinated

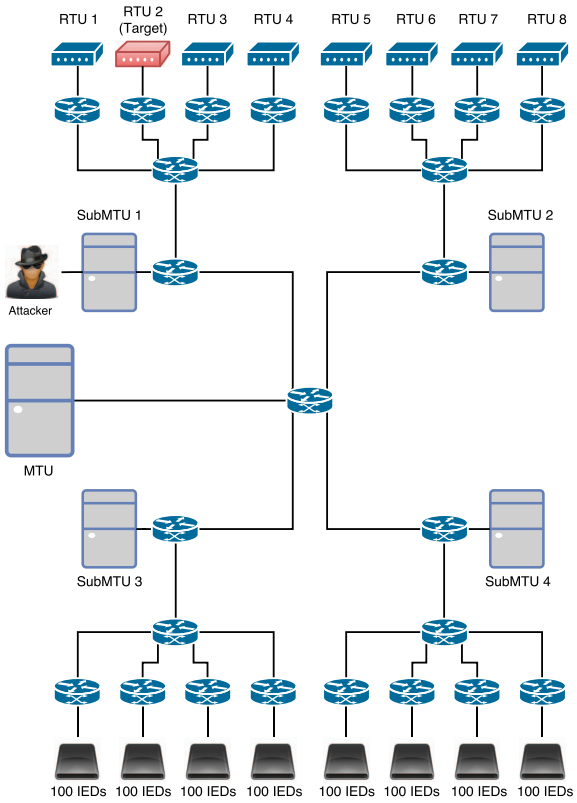



Figure 2. Configuration of the network topology used in our experiments.

devices, every 2 seconds. To respect the time requirements of traditional SCADA systems for power grids, we set the periodicity of network statistics gathering to 4 seconds. Furthermore, our NIDS was configured to collect information from the Historian Server every 4 seconds too. Further, the NIDS was responsible for analyzing the network information stored in the Historian Server and for generating alerts to system operators when a possible anomalous behavior in the SCADA communication network was detected.

During the experiments, we simulated the launch of a DoS attack targeted at one of the substations. We assume that this DoS could be launched by a disgruntled employee of the power distribution company who has remote access to one workstation of subMTU #1. The attacker intends to disrupt the communication between a particular substation (RTU #2) and the rest of the power system to force a hard-reset in the SCADA system and to cause financial losses to the company. The DoS attack is based on a vulnerability of Modbus/TCP named Unauthorized Read Request[6]. This vulnerability allows an attacker with IP connectivity to the RTU to send unlimited data requests and consequently cause a buffer-overflow in the SCADA slave device [29]. Currently, buffer-overflow is a major threat in modern and legacy SCADA systems [30]. We collected 24 hours of training

[1] http://www.noxrepo.org/pox
[2] http://cassandra.apache.org/
[3] https://www.csie.ntu.edu.tw/∼cjlin/libsvm/
[4] https://code.google.com/p/pymodbus/
[5] http://www.mininet.org/

[6] http://www.symantec.com/security_response/attacksignatures

samples to build a classifier model with the daily behavior of the evaluation scenario. Each experiment was conducted during 20 minutes and they contained two types of traffic: 10 minutes represented the expected system functioning; and 10 bursts of 1 minute indicated the occurrence of the DoS attack on the SCADA network.

### *B. Evaluation Results*

We evaluated the quality of the proposed NIDS for SDN-based SCADA systems using the two available SVM-based OCC algorithms for traffic classification, OCSVM and SVDD. With the assistance of a confusion matrix, calculated from the results of our experiments, we analyzed specific metrics for supervised machine learning algorithms, such as true positive rate (TPR), true negative rate (TNR), positive predictive value (PPV), negative predictive value (NPV), false positive rate (FPR), false discovery rate (FDR), false negative rate (FNR), and accuracy (ACC) [31]. In absolute numbers, our experiments generated a total of 29,709 samples which were classified by the proposed NIDS. Of the total of samples generated, 15,579 (52.439%) represented the normal functioning of our SCADA network, whilst the 14,130 (47.561%) remaining samples evidenced the DoS propagation in the power grid. Each repetition of our experiment generated on average 990.3 validation samples, which results in 519.3 positive samples and 471 negative samples on average for each repetition. It is important to note that the experiments were performed 30 times, in order to achieve a confidence level of 95%.

The confusion matrices presented in Figure 3 describe the traffic classification produced by the proposed NIDS. With both algorithms, OCSVM and SVDD, we can observe that our NIDS obtained significant results if we consider the validation samples classified as false-positive (FP) and true-negative (TN). The confusion matrices show that, for all repetitions of the experiments, our NIDS detected correctly and instantly the validation samples that indicated the propagation of the DoS attack on the SCADA network. In other words, 100% of the validation samples that presented the evidence of DoS attack (TN bars in Figure 3) were classified as anomalous behavior, being reported immediately to the SCADA operators. However, if we only analyze the expected behavior, we can see that OCSVM obtained slightly better performance if compared to the SVDD algorithm. OCSVM classified correctly, that is, classified as true-positive (TP), approximately 98.435% of the validation samples, whilst using SVDD this classification was 95.718%. Figure 4 presents a time series of the traffic classification of both OCSVM (Figure 4(a)) and SVDD (Figure 4(b)), respectively. As Figure 4(b) indicates, in the experiment using SVDD, the last three validation samples were classified as FP. This indicates that the RBF Kernel used by OCSVM fits better to the simulated traffic in our SCADA system in relation to the SVDD hypersphere. Note that figures 4(a) and 4(b), for better visualization of the traffic classification, present only part of an experiment, which contains 40 validation samples where 24 samples are the normal expected traffic, and 16 are the traffic samples evidencing the DoS attack.

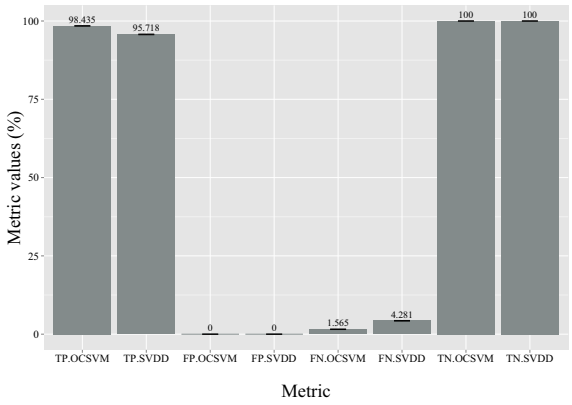


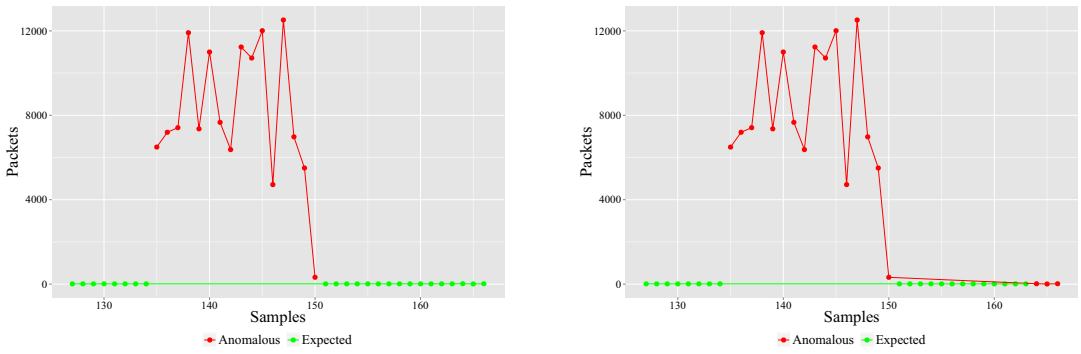

Figure 3. Confusion matrices generated through of the experiments.

As both algorithms obtained similar FP and TN rates, these results also have influence on metrics for evaluating the accuracy of our NIDS. As can be seen in Figures 5(a) and 5(b), both algorithms achieved 100% of TNR and PPV. Consequently, OCSVM and SVDD also achieved 0% of FPR and FDR. These results show that the One-Class NIDS accurately detected and reported the DoS attack in our evaluation scenario. Analyzing the remaining metrics, we can see that SVDD presented a higher FNR (4.281%) if compared to OCSVM (1.565%). OCSVM presented slightly higher TPR (98.435% against 95.718%). Furthermore, OCSVM also presented higher rate of NPV (98.31% against 95.501%). Ultimately, OCSVM obtained slightly better accuracy (99.18%) than SVDD (97.755%). Table I presents an overview of the obtained results in our experiments.

Table I
OVERVIEW OF RESULTS.

| Metrics / Algorithm | OCSVM | SVDD |
|---|---|---|
| **TPR** | $\approx 98.435\%$ | $\approx 95.718\%$ |
| **TNR** | $100\%$ | $100\%$ |
| **PPV** | $100\%$ | $100\%$ |
| **NPV** | $\approx 98.31\%$ | $\approx 95.501\%$ |
| **FPR** | $0\%$ | $0\%$ |
| **FDR** | $0\%$ | $0\%$ |
| **FNR** | $\approx 1.565\%$ | $\approx 4.281\%$ |
| **ACC** | $\approx 99.18\%$ | $\approx 97.755\%$ |

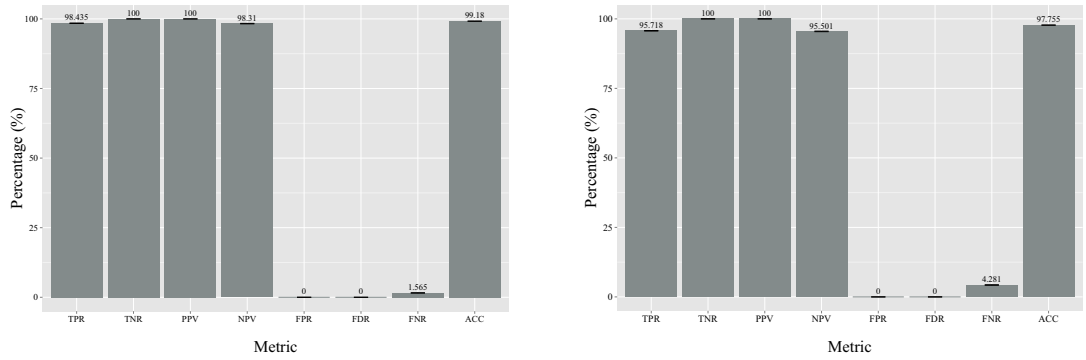
Finally, analyzing the final results, we can state that: novelty detection can be used in the detection of network anomalies in SCADA systems; and in our evaluation scenario, our approach was able to fully detect the DoS attack. Although the proposed NIDS has classified validation samples as FP, it behaved well in relation to the attack detection. This can be proved by analyzing the results obtained on metrics directly based on TN samples or FP samples, such as TNR, PPV, FPR, and FDR. Our experiments also showed




(a) Traffic classification using OCSVM.

(b) Traffic classification using SVDD.

Figure 4. Traffic classification of our One-Class NIDS for SDN-Based SCADA systems.




(a) Metrics using OCSVM.

(b) Metrics using SVDD.

Figure 5. Machine learning metrics obtained from our experiments.

that OCSVM presented slightly better accuracy than SVDD. This means that, given our assumption about the scarcity of public available attack traces in SCADA, it is possible to detect attacks targeted at the power system by using a classifier model that only represents the expected behavior of the SCADA network.

## V. Related Work

In this section, we review research efforts that are related to our work. In Section V-A, we describe research efforts that employ SDN in Smart Grids and SCADA systems. In Section V-B, we present studies that aim to detect possible intrusions in SCADA systems. Finally, in Section V-C, we discuss our contribution to this research topic.

### A. SDN in Smart Grids and SCADA systems

Research efforts investigating the use of SDN in SCADA systems are still scarce. In a previous work, we discussed the potential benefits that SDN can bring to the electrical grid and SCADA systems [4]. The aforementioned paper presents a multipath approach for SDN-based SCADA systems in which communication of SCADA devices is performed by more than one route in order to prevent possible eavesdroppers from fully capturing messages exchanged between SCADA devices. Cahn *et al.* [5] discussed how SDN can alleviate some of the current problems in Smart Grid communication networks. The authors presented the design and development of a new architecture for communication with grid substations, allowing the network to be auto-configurable, secure, and reliable against possible system misconfigurations. Kim *et al.* [32] introduced an SDN-based architecture that simplifies the development of network applications for Smart Grids, enabling self-configuration, security, and scalability. Dorsch *et al.* [33] presented and analyzed advantages and challenges of applying SDN in distribution and transmission networks of Smart Grids. Moreover, that work also introduced algorithms for fast recovery and load management, tested in IEC-61850 traffic. Gyllstrom *et al.* [34] also developed and evaluated algorithms for SDN-based Smart Grid networks. That paper analyzed the performance of algorithms for fast recovery facing link failures. Finally, Goodney *et al.* [17] proposed the use of SDN to control the communication between devices responsible for measuring electrical waves in the grid, known as Phasor Measurement Units (PMUs).

### B. IDS for SCADA Systems

In the literature, there are several research efforts that proposed IDSes for SCADA systems. Almalawi *et al.* [25]

proposed an unsupervised SCADA data-driven anomaly detection approach intended to be used as a passive SCADA IDS. This IDS has two main steps: (*i*) the identification of consistent and inconsistent states from unlabeled SCADA data traffic generated by system sensors and actuators using the density factor for the $k$-nearest neighbors of the observation; and (*ii*) the extraction of proximity-based detection rules for normal and anomalous behavior using statistically determined micro-clusters. Barbosa [7] investigated the main traffic characteristics in SCADA networks and presented a NIDS capable of detecting data injection and DoS attacks. This NIDS specifically explores the periodicity of traffic generated in SCADA systems. Finally, Cheung *et al.* [24] used the network behavior of SCADA components to create behavioral models for traffic analysis and anomaly detection.

### C. Discussion

In this paper, we advocate that the use of SDN will enhance the scalability and improve the flexibility of SCADA systems, and will also facilitate the creation of SCADA resilience mechanisms. We investigated IDSes currently available for SCADA systems and the network behavior of SCADA devices to contribute to this research topic. Unfortunately, there are few IDSes that exploit the network behavior of SCADA, and the systems that use these aspects do not employ the characteristics of SDN to collect more accurate information of the network. We proposed a NIDS that creates signatures of the expected network functioning, *e.g.*, it generates behavior models of the correct functioning of SCADA devices. Our approach periodically verifies the SCADA network, through SDN, in order to find anomalous behaviors that differ from the expected SCADA behavior. This verification is made by OCC-based machine learning algorithms. This choice was based on the characteristics of OCC algorithms, on the requirements of periodicity and connection matrix stability of SCADA systems, and on the paucity of SCADA attack signatures publicly available to research.

## VI. Conclusion and Future Work

Modern power systems comprise energy generation, transmission, and distribution subsystems that are monitored and managed by large-scale SCADA systems. Because of its importance, any threat to SCADA system operation may result in heavy economical losses or even put lives in danger. Therefore, in order to promote the modernization process of power grids, we investigate the development of a new generation of SCADA systems, named SDN-based SCADA systems. By relying on the global view of SDN-based SCADA systems and on their ability to gather switch statistics, we presented a specific NIDS for this kind of environment. Our NIDS uses OCC machine learning algorithms that, with a unique inlier homogeneous training set, can detect anomalous behaviors in SCADA networks, such as unauthorized system or network activity. We presented experimental results in a realistic SCADA environment that validate our prototype and verify the accuracy of the classification techniques, applied to the detection of a DoS attack based on a real vulnerability of the Modbus/TCP protocol. Our analysis was based on a comparison of two OCC algorithms, OCSVM and SVDD, which deal well with large datasets and have a fast classification process if compared to other machine learning techniques.

As future work, we intend to carry out a performance analysis of our NIDS using other OCC algorithms, such as, Kernel Principal Component Analysis (KPCA) [35] and One-Class Random Forests (OCRF) [36]. We also intend to combine classifiers to improve the accuracy of our proposal. In addition, we intend to incorporate the Feature Selector component in our solution to minimize possible false alerts. Finally, we plan to implement a user-friendly interface, for defining alternatives to mitigate the anomalous behaviors without compromising the functioning of SCADA devices.

## Acknowledgement

This work is supported by ProSeG - Information Security, Protection and Resilience in Smart Grids, a research project funded by MCTI/CNPq/CT-ENERG # 33/2013.

## References

[1] W. Wang, Y. Xu, and M. Khanna, "A Survey on the Communication Architectures in Smart Grid," *Computer Networks*, vol. 55, no. 15, pp. 3604–3629, Oct. 2011.

[2] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A Survey on Smart Grid Communication Infrastructures: Motivations, requirements and challenges," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 1, pp. 5–20, First 2013.

[3] K. Stouffer, J. Falco, and K. Scarfone, "Guide to Industrial Control Systems (ICS) Security," *NIST special publication*, pp. 800–82, 2011.

[4] E. Silva, L. Knob, J. Wickboldt, L. Gaspary, L. Granville, and A. Schaeffer-Filho, "Capitalizing on SDN-Based SCADA Systems: An anti-eavesdropping case-study," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 165–173.

[5] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, "Software-Defined Energy Communication Networks: From substation automation to future smart grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 558–563.

[6] J. Wickboldt, W. Jesus, P. Isolani, C. Both, J. Rochol, and L. Granville, "Software-Defined Networking: Management requirements and challenges," *Communications Magazine, IEEE*, vol. 53, no. 1, pp. 278–285, January 2015.

[7] R. Barbosa, "Anomaly Detection in SCADA Systems: A network based approach," Ph.D. dissertation, University of Twente, Enschede, April 2014. [Online]. Available: http://doc.utwente.nl/90271/

[8] H. Liao, C. Lin, Y. Lin, and K. Tung, "Intrusion Detection System: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16 – 24, 2013.

[9] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN," *Queue*, vol. 11, no. 12, pp. 20:20–20:40, Dec. 2013.

[10] J. Davis and A. Clark, "Data Preprocessing for Anomaly based Network Intrusion Detection: A review," *Computers & Security*, vol. 30, no. 67, pp. 353 – 375, 2011.

[11] S. Khan and M. Madden, "A Survey of Recent Trends in One Class Classification," in *Artificial Intelligence and Cognitive Science*, ser. Lecture Notes in Computer Science, L. Coyle and J. Freyne, Eds. Springer Berlin Heidelberg, 2010, vol. 6206, pp. 188–197.

[12] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[13] D. Tax and R. Duin, "Support Vector Data Description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[14] B. Zhu, A. Joseph, and S. Sastry, "A Taxonomy of Cyber Attacks on SCADA Systems," in *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, ser. ITHINGSCPSCOM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 380–388.

[15] V. Igure, S. Laughter, and R. Williams, "Security Issues in SCADA Networks," *Computers & Security*, vol. 25, no. 7, pp. 498 – 506, 2006.

[16] L. Jing, Y. Xiao, S. Li, W. Liang, and C. Chen, "Cyber Security and Privacy Issues in Smart Grids," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 981–997, Fourth 2012.

[17] A. Goodney, S. Kumar, A. Ravi, and Y. Cho, "Efficient PMU Networking with Software Defined Networks," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 378–383.

[18] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[19] P. Nader, P. Honeine, and P. Beauseroy, "Intrusion Detection in SCADA Systems using One-Class Classification," in *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, Sept 2013, pp. 1–5.

[20] M. Hadley and K. Huston, "Secure SCADA Communication Protocol Performance Test Results," *Pacific Northwest National Laboratory (August 2007)*, 2007.

[21] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] R. Caruana and A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 161–168.

[23] D. Tax and R. Duin, "Support Vector Domain Description," *Pattern Recognition Letters*, vol. 20, no. 1113, pp. 1191 – 1199, 1999.

[24] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using Model-Based Intrusion Detection for SCADA Networks," in *Proceedings of the SCADA Security Scientific Symposium*, vol. 46, 2007, pp. 1–12.

[25] A. Almalawi, X. Yu, Z. Tari, A. Fahad, and I. Khalil, "An Unsupervised Anomaly-Based Detection Approach for Integrity Attacks on SCADA Systems," *Computers & Security*, vol. 46, no. 0, pp. 94 – 110, 2014.

[26] A. Silva, C. Machado, R. Bisol, L. Granville, and A. Schaeffer-Filho, "Identification and Selection of Flow Features for Accurate Traffic Classification in SDN," in *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, Sept 2015, pp. 134–141.

[27] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[28] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack Taxonomies for the Modbus Protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37 – 44, 2008.

[29] A. Wermann, M. Bortolozzo, E. Silva, A. Schaeffer-Filho, L. Gaspary, and A. Barcellos, "ASTORIA: A framework for attack simulation and evaluation in smart grids." in *Network Operations and Management Symposium (NOMS), 2016 IFIP/IEEE*, April 2016, to appear.

[30] D. Incorporated, "Dell Security Annual Threat Report," Dell Incorporated, Tech. Rep., 2015. [Online]. Available: https://software.dell.com/whitepaper/dell-network-security-threat-report-2014874708

[31] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[32] Y. Kim, K. He, M. Thottan, and J. Deshpande, "Virtualized and Self-Configurable Utility Communications Enabled by Software-Defined Networks," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 416–421.

[33] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, and C. Wietfeld, "Software-Defined Networking for Smart Grid Communications: Applications, challenges and advantages," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 422–427.

[34] D. Gyllstrom, N. Braga, and J. Kurose, "Recovery from Link Failures in a Smart Grid Communication Network using OpenFlow," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 254–259.

[35] H. Hoffmann, "Kernel PCA for Novelty Detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863 – 874, 2007.

[36] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, "One Class Random Forests," *Pattern Recognition*, vol. 46, no. 12, pp. 3490 – 3506, 2013.