MINISTRY OF EDUCATION
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

Computer Science and Information Technology

# PhD THESIS
## – ABSTRACT –

# Contributions to Improving the Performance in Software-Defined Networks

**PhD Student:**
**Sorin Buzura**

**PhD Supervisor:**
**Prof. Eng. Vasile-Teodor DĂDÂRLAT, PhD**

**Examination committee:**
Chair: Prof. Eng. **Rodica Potolea**, PhD - Technical University of Cluj-Napoca;
PhD Supervisor: Prof. Eng. **Vasile-Teodor Dădârlat**, PhD –
Technical University of Cluj-Napoca;
Members:
- Prof. Eng. **Mariana Mocanu**, PhD - Politehnica University of Bucharest;
- Prof. Eng. **Adrian Munteanu**, PhD - Vrije Universiteit Brussel, Belgium;
- Reader Eng. **Piroska Haller**, PhD - "George Emil Palade" University of Medicine, Pharmacy, Science and Technology of Târgu Mureș;
- Reader Eng. **Bogdan Iancu**, PhD - Technical University of Cluj-Napoca.

**– Cluj-Napoca –**
**2023**

# 1. Introduction

The progress in the field technological communications has encouraged the rapid development of large-scale computer networks. This is visible today in the form of the internet, which is used in most of our daily activities, whether professional or personal. The rapid development of the internet, due to global demand, was achieved through the deployment of multiple identical hardware devices (such as routers and switches), which all implement the same communication protocols (without the possibility of being easily customised). This architecture is known as the traditional network architecture, which is based on the homogeneity of the devices it contains. Although this architecture is easily scalable, the devices are not easily customisable, and this does not serve the current requirements where different applications and industries need custom management. This problem is also enhanced by the increasing heterogeneity of connected devices in modern networks. To address these issues, the Software Defined Networking (SDN) paradigm was introduced, which aims to define the operating policies of a network in a software configurable and dynamic way.

## 1.1. Context

The software-defined networking paradigm represents a new way of designing networks to keep up pace with the emerging innovations developments in the research field. This paradigm does not directly address the problems of traditional networks, such as routing, traffic engineering or security, but it gives the possibility to implement innovative solutions to address these problems [1]. Software-defined networks do not depend on the hardware over which they are built, and the complexity of designing such a network is moved to the software level. A software-defined network is divided into several planes, which are used depending on the deployment context of the network. The separate planes are the following: data plane, operational plane, control plane, management plane and applications plane. The data plane manages the packets transmitted over the network based on the rules received from the control plane. The main operations that devices in the data plane can perform on network packets are: forward, delete and change. The operational plane is theoretically separated, although in practical implementations its functions are part of the data plane. The main functions of this plane are to determine the status of devices (e.g. on or off) or the number of available ports on each device. The control plane is responsible to make decisions on network traffic by sending instructions to devices in the data plane containing instructions on what actions to perform on the data packets. The management plane is responsible for monitoring and configuring the operation of network devices, especially the devices in the control plane. The application plane contains the applications that use the network services which also define the deployment context of the network [2].

## 1.2. Motivation

The challenges proposed in the current thesis are about the improvement of Quality of Service (QoS) and Quality of Experience (QoE) of end users in software-defined networks. Regarding the proposed topic of services and performance improvement in programmable networks, performance metrics need to be defined in relation to each individual contribution. A scientific process is characterised by two features: measurability and repeatability. Thus, for

each test scenario the measured parameters must be defined and each simulation or experiment must have a deterministic behaviour. In traditional networks, QoS is characterised by measuring the volume of data transmitted between two network nodes in a given time interval. QoE is defined as the level of user satisfaction with network services [3]. This is not so easy to implement in software-defined networks, where architectural separation into the different planes together with the use of multiple protocols (e.g. OpenFlow, LISP - Location Identifier Separation Protocol or BGP - Border Gateway Protocol) allow different local optimizations, which are not easily measurable when transmitting data between two different terminal nodes. Thus, an important challenge that is addressed by this PhD thesis is to identify the measurement metrics for each simulation and each individual experiment that highlight the proposed improvements.

The research that was made in this thesis is significant because it provides a framework that addresses various theoretical and practical topics in the areas of traffic engineering, collaborative systems, artificial intelligence and security. Technical innovations are proposed in all of these theoretical areas of work and at all the architectural levels of software-defined networks. The above topics encompass most of the functions of a network. Their aggregate implementation in the same system provides a complete solution that can be used in software-defined networks. While most studies focus on a single topic where the same traditional metrics are used to measure performance, this thesis proposes new network topologies and new benchmarks for simulations and experiments. Having newly proposed topologies that address interdisciplinary topics, the measurement metrics also need to be adapted to represent different network operations that influence other functionalities within the network. One such example, which will be detailed in the following chapters, is in the context of security where a network attack consumes the available network bandwidth. This leads to a decrease in the bandwidth allocated to useful and relevant data traffic. The proposed metric to improve network performance is the detection time of an attack occurring in the network, which, the lower it is, the more it succeeds in increasing the amount of critical data traffic. So, it is not strictly measuring the transmission speed at the two ends, but it is measuring the duration of another operation that subsequently affects this speed of the relevant network traffic.

The complexity of this thesis is determined not only by the technical difficulties, but also by the chosen working methods, which, through incremental development, have led to results throughout the entire working period. Among the technical elements that demonstrate the increased complexity are the following: the interdisciplinary nature of the thesis that addresses different areas with applicability in the field of software-defined networking; the heterogeneous nature of the work carried out that uses different hardware technologies, different software technologies, but also different communication protocols between network devices; the development of solutions using information from all layers of the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol stack; the implementation of software solutions in different operating systems (Windows, Linux) and in different programming languages (C, C++, Windows Batch, Matlab, Python, Java, C#, Linux Bash, qmake); the automation of test scenarios through software programmability. Among the learning and working methods that have facilitated the development of the published solutions are the following methodologies: problem based learning - this methodology assumes the existence of a problem that has no solution and must be solved in some way; project based learning - this methodology assumes the existence of a project feature that must be brought to completion; challenge based learning - this methodology assumes the achievement of a goal for which the necessary knowledge does not yet exist, and which must first be identified and then learned.

## 1.3. Objectives

This subchapter presents the objectives of this thesis. The main objectives address the conceptual and theoretical aspects of the proposed topics. The secondary objectives address the implementation details that were necessary to fulfil the main objectives. A chapter from the personal contributions section corresponds to each main objective. Of the four personal contribution chapters, the first three address individual research innovations, while the last chapter incorporates the previous innovations into the same working environment.

### 1.3.1. Main Objectives

1.3.1.1. The main objective of the first personal contribution chapter is to propose performance improvement in software-defined networks by improving the QoS and QoE characteristics of the network by reducing redundant traffic at the data plane level in a wireless sensor network. (Chapter 3)

1.3.1.2. The main objective of the second personal contribution chapter is to propose performance improvement in software-defined networks by developing a distributed and collaborative control plane for reducing control traffic and maximizing critical data traffic in the network using various collaboration techniques. (Chapter 4)

1.3.1.3. The main objective of the third personal contribution chapter is to propose performance improvement in software-defined networks by addressing security risks that may negatively impact essential network data traffic. (Chapter 5)

1.3.1.4. The main objective of the fourth personal contribution chapter is to propose performance improvement in software-defined networks by creating a hybrid software and hardware development framework that allows concurrent testing of the policies defined in the previous objectives. (Chapter 6)

### 1.3.2. Secondary Objectives

The secondary objectives are grouped for each main objective represented by a separate personal contribution chapter:

1.3.2.1. Secondary objectives for traffic management in an SDWSN (Chapter 3):

- Study of different traffic management methods that can be integrated into an SDN
- Creation of a test network on a single device using the property of network interfaces to have multiple IP addresses configured on them
- Identification and usage of a representative data set for testing the implementation
- Definition of the performance metrics at the data plane layer in an SDWSN

1.3.2.2. Secondary objectives for the development of the distributed and collaborative control plane (Chapter 4):

- Implementation of the network communication between the devices in the distributed system (the distributed and collaborative control plane)
- Usage of the OpenVSwitch solution and the OpenFlow protocol to generate data sets during the testing phase

- Study and implementation of the fuzzing technique in the context of QoS in SDN
- Study and implementation of artificial intelligence (AI) techniques in SDN, more precisely the decentralised federated learning technique
- Definition of the performance metrics in the interaction between the control plane and the data plane of the SDN

1.3.2.3. Secondary objectives for addressing security risks in an SDN (Chapter 5):

- Study of different types of attacks that can be generated in the network
- Execution of network attacks using Kali Linux
- Usage of the OpenVSwitch software solution based on the OpenFlow protocol and its extension with functionality that will run at the data plane layer of the SDN
- Definition of metrics for measuring the performance in the data plane layer of the SDN

1.3.2.4. Secondary objectives for the development of a hybrid development framework (Chapter 6):

- Demonstration of the heterogeneous and interdisciplinary nature of the topic of software-defined networks
- Integration of different programming techniques and software architectures in the research work for rapid development and prototyping
- Application of solutions at all architectural layers of the SDN (application, control and data planes)
- Implementation of the framework using working environments used in research, such as Mininet and Mininet-WiFi
- Integrating multiple hardware devices into networks created in Mininet and Mininet-WiFi
- Development of methods using protocols from all layers of the TCP/IP protocol stack

## 1.4. Presentation of the Thesis Research

This PhD thesis is defended by the following published results: 3 articles in ISI indexed journals (all of them as first author), 4 papers presented at ISI Proceedings indexed conferences (3 of them as first author), 6 papers presented at IDB indexed international conferences (3 of them as first author).

# 2. Bibliographical Study Related to the Proposed Objectives

## 2.1. Traffic Engineering in Software-Defined Wireless Sensor Networks

User expectations are reflected by the increasing functionality of applications that use data provided by a wireless sensor network [4]. The academic community is making continuous efforts to provide opportunities to meet user requirements by offering dynamic programming solutions for such networks. The ultimate goal of these efforts is to allow the management of heterogeneous networks with different topologies as autonomously as possible [5], [6], [7]. Software-defined networking is now the main paradigm used for network programmability. It is important to note at this point that this paradigm does not impose a specific communication protocol between its planes, but OpenFlow is the most popular encountered protocol [8]. [9] proposes a routing algorithm in the context of SDWSN that defines packet transmission routes according to the energy level of each node in the network. [10] shows the activity patterns of a WSN while data is being transmitted or while the network is idle. There are studies in the literature that use different data caching methods in WSNs [11]. [12] uses a variant of data content caching. Another way to improve traffic quality is to use a routing algorithm that also uses load balancing techniques, or clustering [13]. [14] succeeds in formulating a scheduling mechanism that improves the energy efficiency of a SDWSN by a variable percentage between 20-40%. [15] proposes a data flow separation algorithm to minimize congestion with the aim of improving network energy efficiency. [16] shows a compromise solution that forwards data computed from a hashing operation. [17] presents ways to find similarities between data by processing the generated hashes. One of the most widely used techniques to reduce energy consumption in WSNs is grouping sensors in a cluster [18], [19]. The coordinating sensor is chosen based on the energy level of the sensors in the cluster, but different methods of predicting and probabilizing energy levels can also be chosen [20]. [21] uses machine learning technique to decide which sensor is the coordinating sensor of the cluster, thus reducing the transmitted traffic.

## 2.2. Collaborative Systems in the Software-Defined Networks Control Plane

Improvements in QoS are continuously being adjusted and will evolve as networks improve their operating parameters [22]. The OpenFlow protocol is used to standardize communication between the data plane and the control plane of a software-defined network. By removing some computational functions from devices in the data plane, they will increase the amount of data they can forward, resulting in improved network QoS characteristics. This is demonstrated in [23], which also argues that one of the most important parameters in the OpenFlow rules that can be used in the collaborative system is the OpenFlow rule lifetime, i.e. the time until the rule is deleted when it expires. [24] implements a framework in clusters of nodes of a data plane with the aim of providing improvements for QoS. [25] validates the use of the OpenFlow protocol in software-defined networks with the end goal to provide adaptive and dynamic QoS. [26] validates the use of a distributed control plane especially in the context of software-defined wireless sensor networks. [27] explains different design architectures for the control plane in software-defined networks. There are many techniques used to improve the QoS offered by computer networks: Random Early Detection (RED) algorithms, machine learning [28], [29], different traffic classification methods [30], [31], [32], etc. [33] presents

different ways in which learning-based techniques are integrated in distributed systems, such as in the context of the current PhD thesis. Recently defined by Google in 2017 [34], federated learning has since grown to be used in industry for various use cases in QoS related areas [35]. [36] presents different applications of using federated learning technique, but a general feature is that the process involves a centralized global model that is cloned and distributed to each participating node. [37] proposes a new solution in which the main idea is to communicate only the sign of each gradient update instead of the actual value. [38] uses ternary sparse compression to solve the non-IDD data problem. The algorithm is inspired by the top-k sparsification technique, communicating only a small fraction of the largest gradients in the model. [39] is a study that applies the federated learning technique to improve packet routing in a software-defined network. [40] uses federated learning to improve traffic monitoring systems based on the data they collect. [41] addresses the problem of data traffic classification by proposing a traffic classification protocol. In a switch implementing the OpenFlow protocol, whenever a packet does not match any flow record, packet transmission is stopped until the associated control device responds with the validity duration for that data flow [42]. [43] addresses the table miss problem and attempts to mitigate it by implementing functionality on OpenFlow switches not to delete flow records when their validity duration expires, but rather to mark them as inactive for a period of time in case the same rule is needed in the immediate future. [44] also addresses the problem of reducing the number of table misses, but in this case the paper uses buffering and cache-rolling for a series of packets belonging to the same stream, thus sending a single packet_in request for the series of packets belonging to the same data stream. For the system using the fuzzing technique in the current PhD thesis, simulations use datasets containing values similar to those presented in [45] to measure the performance of the system.

## 2.3. Security in Software-Defined Networks

Along with the positive opportunities introduced in the field of software-defined networking, security risks in such networks have also increased and, as a result, much research effort is directed towards detecting and mitigating network attacks [46]. Attacks based on the ARP spoofing techniques allow a malicious actor to assume the identity of another network device by generating ARP requests (via gratuitous ARP packets), but which appear to originate from the device intended for actual communication on the network [47]. It is important to note that the ARP spoofing attack is only the attack that creates the context for other attacks, such as MitM (Man in the Middle). Therefore, it is actually the attack that needs to be detected and mitigated, not the attacks that are developed afterwards [48]. [49] highlights three main categories of solutions against ARP spoofing attacks: solutions based on traffic patterns; solutions based on flowcharts; solutions based on analysing mappings between IP addresses and MAC addresses. [50] is an application that analyses data traffic and monitors all ARP requests flowing through the network, and when a request contains mappings that are inconsistent with the mappings already existing in the monitoring application's database, this data frame is blocked. [51] presents an improved network architecture that prevents the generation of ARP MitM attacks. [52] is a study that relies on mappings stored on other devices in the network. The mechanism of operation of the OpenFlow communication protocol runs in several steps according to [53], and the complexity and duration of this whole process is intended to be avoided. The flow graphs are based on the interception and interpretation of OpenFlow packets in the network. [54] dynamically generates flow graphs for OpenFlow traffic between the control plane and the data plane of a software-defined network. [55] does not operate in the context of software-

defined networks, but provides an approach for addressing ARP spoofing attacks by analysing network traffic using the JPCAP library for monitoring data traffic. A similar situation is found in [56], where a control device is extended with functionality that notes malicious MAC addresses in a blacklist, and this list is passed to switches via the OpenFlow protocol. [57] designs a multi-step algorithm to detect attacks in real time. [58] argues that computational efficiency is improved when the controller is not involved in decision making. [59] describes the implementation of methods to reject ARP spoofing attacks. The solution is implemented as an extension to the Open Floodlight control solution [60]. [61] states that it demonstrates a solution to the same problem, being implemented in the data plane, but it requires Ryu [62] to define the initial rules, which are passed to switches in the data plane. From a technical perspective, [63] proposes a detection algorithm also implemented in Bash, but running only on terminal devices, and the attack can only be detected if it is performed towards the current machine. [64] considers the impact of such solutions from a QoS point of view.

## 2.4. Frameworks for Simulations and Experiments

Mininet [65] is an emulator for Ethernet wired networks, but it has other extensions, such as Mininet-Wifi [66] and Mininet-Optical [67], which implement Layer 2 protocols from the ISO/OSI protocol stack, specific to wireless and optical transmission technologies respectively. An example for the utilization of the OpenFlow protocol is the study in [68], which explains the use of the POX SDN controller solution [69] for defining operating rules in the software-defined network. [70] presents an extension of the Cooja simulator under the name WSDP (Weather & Soil Data Provider). This extension is actually a plugin for the Cooja simulator [71] that is able to generate real-time geographic data for sensors created inside the simulator. [72] presents a complex approach to address congestion in networks. [73] presents a framework for simulating the route generation process based on different traffic categories in a 6LoWPAN wireless sensor network. [74] describes a solution on how to manage wireless sensor networks using the SDWSN paradigm where energy consumption is reduced by using control plane for network management. Another study addressing the dynamic routing problem is presented in [75] where a real-world environment is used to test different routing algorithms with the aim of increasing network lifetime. [76] uses the OMNeT++ simulator [77] to implement algorithms to improve the throughput of certain data flows in the event of congestion at the router level. The study presented in [78] explores different technologies used in software-defined network performance and stability experiments. Another study measuring the performance of POX and Ryu control solutions in networks built in Mininet can be found at [79]. Different topologies of linear networks, tree networks or data centre networks are used in the simulations. [80] uses a load balancing algorithm, which is based on monitoring the traffic in the used sensor network. The study presented in [81] uses Net2Plan [82], a network function management utility, together with the SDN-specific controller OpenDaylight, which is based on the Java programming language. [83] creates a framework to address energy efficiency and security in wireless sensor networks in the IoT domain. [84] uses Mininet-WiFi together with the Ryu controller to measure the performance of a multipath-based routing algorithm with the ultimate goal of improving QoS rules. For real-time programmatic processing of packets flowing through the network, PCAP (Packet CAPture) software solutions can be used, including the following: Npcap [85], SharpPcap [86], Pcap4J [87] or jNetPcap [88]. These solutions are usually used as a static or dynamic library by the main program. [89] presents a client-level QoS monitoring and management system using the SharpPcap library. [90] presents the challenges offered by a large-scale IoT environment. [91] presents a software solution that compares the execution time of some

operations performed at the network level. [92] addresses an emerging topic of unmanned aerial vehicles (drones), which benefit from autonomous control when they lose network connection. The study in [93] presents five large-scale SDN frameworks used in campus or backbone networks. The study in [94] addresses the problem of large-scale networks where multiple control devices are needed to serve the network requirements. [95] discusses various issues related to the placement of SDN control devices. [96] describes a software solution to simulate a data management system in a cloud computing context. [97] analyses asynchronous transmissions between nodes in a network with the ultimate goal of improving overall network performance and reducing overall energy consumption. [98] classifies different approaches for managing applications and work operations in large-scale systems. [99] discusses the issue of security in software-defined networks. [100] proposes the IoTSim-SDWAN solution, a simulator capable of modelling, simulating and evaluating new solutions in SD-WAN systems and datacenters built over SDN architecture. [101] proposes a mechanism for deploying IoT devices at the network edge using the methodology provided by SDN. [102] demonstrates the use of two architectures for control plane design, namely, a linear and a hierarchical architecture. According to [103], traditional methods used in the context of software updates include the following: manually updating each sensor; running a static operating system image connected statically at the node level; using dynamic operating systems. According to [104], for updating a WSN, three elements closely related to power level management in WSNs and related to the computational capacity of all devices must be considered: the execution environment at each sensor level; the network data transmission protocol; the algorithm used to optimize the software update process. In addition to these traditional elements, modern networks require several additional elements, according to [105], namely: non-intrusive remote updating (Over the Air); reducing the impact on WSN performance; limiting communication between devices and constraining communication to essential traffic only; rapid propagation of software updates in the network; implementing the ability to scale networks. [106] addresses this issue from a security perspective by using a blockchain system in the data transport mechanism. [107] also addresses this problem from a security point of view and implements a proprietary authentication key allocation and distribution system for the software update process. [108] discusses different industrial IoT standards such as Zigbee [109], WirelessHART [110], ISA.100 [111] and how they operate at different levels of the ISO/OSI protocol stack.

# 3. Improving the QoS Through Traffic Engineering

## 3.1. System Design

The research objective of this chapter is to implement a system in which a sufficiently knowledgeable control component can manage a wireless sensor network and determine whether or not the sensors should transmit data. In addition, the controller can also decide whether to transmit data proactively for each sensor, thus reducing the amount of data traffic in the data plane, which represents a WSN. This proposed system is shown in Figure 3.1.

In order for the controller to perform the proposed functions, it implements two components. First, it implements a basic learning component that learns the behaviour for each sensor, i.e. the data transmission frequency for each sensor, to determine a transmission pattern. Second, it implements a data transmission mechanism based on the previously determined interval patterns. It is important to note that the control component does not collect energy levels from sensors to improve energy efficiency, but relies solely on the data transmission frequency from each sensor.

The test environment shown in Figure 3.1 was built by implementing a control component and a generic, reusable sensor component in software. The control component and several sensor instances were deployed in different virtual machines on different computers in a wireless LAN. For initial tests, it is also possible to configure multiple network devices in the same machine using virtual network interfaces, but for final measurements, the wireless communication between network devices is required. This is mandatory to bring the environment closer to a real WSN.

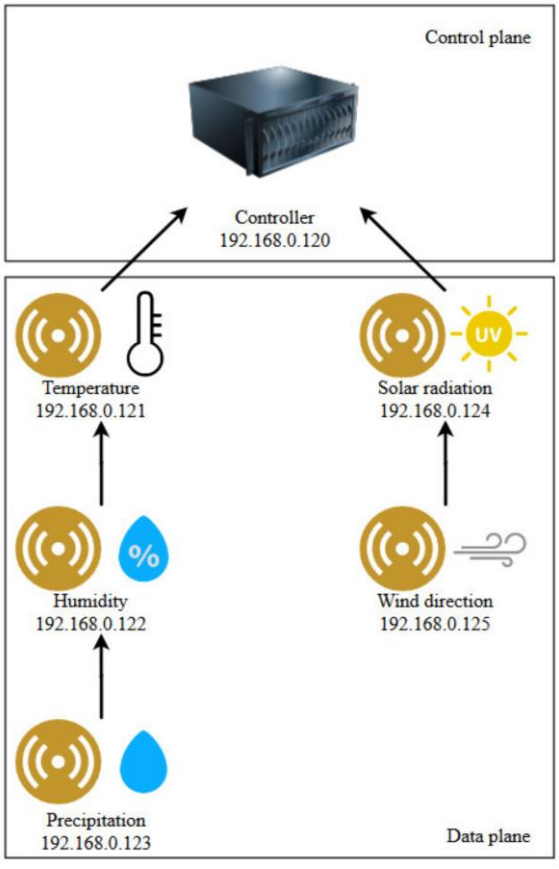


*Figure 3.1 Network topology used in the testing scenarios*

Each sensor implements functionality to generate sensor data and to act as a relay node according to the illustrated connections between network devices. Unlike using network emulators such as Mininet, where all network devices are run from a single computer, this approach has the advantage that there is a more realistic time interval between packet transmission and receipt. In addition, each sensor can be easily configured to generate random data or to replicate already stored data from a reference dataset at any interval. The network supports the IP protocol and static IP addresses have been configured on network devices according to the network topology. The sensor nodes were implemented with modules to generate sensor data, generate HELLO packets and routing capabilities to forward packets to a destination. Regarding the use of HELLO packets, these are not included in the mathematical model of the current proposal. This is based on the assumption that the use of HELLO messages is already a mechanism implemented by data transmission standards that are found at the data link layer of the TCP/IP protocol stack, checking whether wireless nodes are active or not. The wireless standard and literature guarantee that during the nonconstrained period, more information can be combined into a single packet, such as data and acknowledgements [112]. The minimum size of a TCP packet is 20 bytes [113], and when added to the size of the data payload, the results may have a different value depending on the type of packet transmitted, whether it represents sensor data or control data. Sensor data packets on network devices were generated from real sensor data retrieved from online data sources [114] over a four-week period. HELLO packets were designed to be generated at a much higher frequency than sensor data packets. The components were implemented in C++ using the Qt framework [115] for the facilities it brings to network communication and general software development.

## 3.2. Practical Results

The simulated context was the climate differences in various geographical regions. It should be noted here that the system works in other contexts as well, but the test network was chosen with the weather test subject because there are many already available data sources that can be used to perform the tests. Historical weather data was taken from the website of the data source referred to in [114] for different locations: Reykjavik (Iceland), Ouarzazate (Morocco), Budapest (Hungary). The reason why these three locations were chosen is due to the different climates they present. Reykjavik has a rainy oceanic climate, Ouarzazate has a desert climate, while Budapest has a temperate-continental climate. Calculation of the gains for the different network lifetime scenarios shows that the proposed system exhibits consistent behaviour in each scenario. In the long lifetime scenario, the total gain (i.e. percentage of optimised packets) is 25.1% of all packets. In the medium lifetime scenario, the total gain represents 25.71% of the total number of packets. And finally, in the short lifetime scenario, the gain represents 25.05% of all packets. This leads to the conclusion that the system saves about 25% of the generated network traffic. Most of the saved data values that are not transmitted are sensor defaults, which are generated when the sensors have nothing to report. The small difference between the scenarios depends solely on the nature of the sensor data in the selected dataset. The literature describes 5% as being critical for the QoS of a system [116].

## 3.3. Discussions and Conclusions

The system proves to be effective in eliminating duplicate traffic in the WSN layer by using a few simple algorithms that are implemented at both the controller and sensor levels.

They combine concepts such as learning (at the control device level), content awareness (at the sensor level) and caching. The system is efficient even in packet-loss environments, as demonstrated by previous test scenarios where transmission errors have occurred. The threshold at which the system becomes less efficient is when the packet transmission rate reaches 15%, which is considerably higher than the threshold defined in the literature for lower quality QoS in wireless data systems. The additional memory resources required by the system are negligible in the WSN layer. The controller needs more memory to cache data from each sensor in the topology; however, the controller does not belong to a resource-constrained environment and can be more easily updated to support deployment. In addition to reducing energy consumption in the WSN layer, this proposed system has an additional benefit of reducing congestion in sensor queues due to reduced network traffic, thus speeding up packet processing.

Considering the test scenarios, which use data from three real locations, it can be said that the system is more suitable for selective deployment in one location than in all locations. This is due to the real data values, which may be more stable in some parts of the planet. However, this is by no means a rule in all possible scenarios, as the test data may also be different depending on the selected time period. In any case, a definitive conclusion that can be drawn is that the system brings optimisations in any environment in which it is implemented. The system also succeeds in increasing network lifetime because fewer packets are transmitted by the WSN layer, it improves QoS by reducing traffic volume in the WSN layer, and it improves QoE by proactively transmitting cached data to the application plane instead of waiting for the same data to be received from the sensors. There are some limitations and situations where QoE is not achieved, but these situations are considered as not being critical situations.

In conclusion, the main objective defined in subchapter 1.3.1.1. was achieved by improving network performance through traffic engineering and the desired effect of increasing network lifetime was successfully obtained. The secondary objectives defined in subchapter 1.3.2.1. were also fully achieved by the following aspects: traffic was handled by data caching and adaptive data broadcasting techniques; the system was implemented using socket communication between multiple IP addresses configured on the local interface of the computer used for the simulations; the actual dataset was taken from the Meteoblue website containing historical data accessible for validation of the solution; the metric for performance verification was chosen as the network lifetime.

## 3.4. Results Dissemination

The results obtained in this chapter were disseminated in the following publication:

[117] Buzura, S.; Iancu, B.; Dadarlat, V.; Peculea, A.; Cebuc, E. Optimizations for Energy Efficiency in Software-Defined Wireless Sensor Networks. Sensors 2020, 20, 4779. https://doi.org/10.3390/s20174779 IF: 3.576 Q1

# 4. Development of Collaborative Systems to Improve QoS Features in Software-Defined Networks

## 4.1. System Design

Two environments were defined and implemented in this stage. The first environment was based on simulating communication between network devices and simulating network traffic on the TCP, UDP and ICMP protocols. This approach was useful to validate the proposed algorithms before applying them in real data traffic context. The second environment was based on generating network traffic between multiple devices running on the same computer. This was achieved by using multiple local IP addresses (loopback addresses: 127.0.0.1 - 127.255.255.255 or multiple IP addresses configured on the same network interface) that allow processes in the operating system to use a specific Layer 3 address. This approach is useful for generating realistic test scenarios based on real network traffic. Realistic test scenarios are necessary to generate conclusive results. Three topologies of the proposed networks were considered. These are: a network with a non-distributed control plane; a network with a distributed and non-collaborative control plane; a network with a distributed and collaborative control plane. The first topology's purpose was more to validate the continuous implementation of the used algorithms. Also, the non-distributed and distributed noncollaborative scenarios also had the role of validating the proposed distributed and collaborative system improvements at the end.

### 4.1.2. The Design for a Distributed Fuzzing System

As previously mentioned, the goal of the system is to be self-adaptive and automatically adjust the traffic switching rules based on the number of missed rules. The number of failed rules is actually the number of packets for which the switch does not have an active switching rule and is forced to ask the controller for instructions on how to act on that packet. The fuzzing technique was chosen as the way for control devices to make decisions about changing packet switching rules, while the distributed method of using multiple control components allows multiple decisions to be made simultaneously, thus finding the optimal set of rules faster than when using a single control component. Communication between control devices is achieved by means of broadcast messages (transmitted throughout the control plane) through which the control devices share their configurations and statistics for each configuration, namely, a table with each timeout value to be used for each traffic type and the miss counter for each traffic type. The algorithm is implemented on the network control devices and assumes that the environment is a fully reactive SDWSN system, where the SDN component's data plane switches do not contain predefined rules at initialization for packet switching.

### 4.1.3. The Design for a Distributed Decentralized Federated Learning System

The system design begins with each control device waiting the receipt of a packet. When the packet is received, the controller checks the source of the packet if it was generated by another controller or by the directly connected switch from the data plane. If the source of the packet is another control device, this packet may only contain new training data to be stored for re-training. A control device starts its own training procedure when it shares data with other control devices. If the source of the packet is the attached switch, this packet can be

of two types. The first type of packet is an OpenFlow packet_in packet, and the second is a packet responding to the control device's request for training data from the switch. If the packet is of type OpenFlow packet_in, the controller increments the error counter for the traffic type queried by the communicator and responds to the switch with a packet of type packet_flow_mod containing the timeout value (to inform the switch how long it can redirect packets of that protocol type) that will be stored on the switch until it expires. The timeout that the controlling device responds with is calculated every three minutes by querying the local machine learning library. If the packet represents the response received for the training data request, this data will then be processed for the machine learning component.

## 4.2. Practical Results

In order to obtain the practical results, simulations were carried out in the topologies described in section 4.1. The three topologies were tested in the simulation scenario for the fuzzing technique. The topologies with multiple control devices were used for testing the federated learning scenario. The reason why all three topologies were only used in the fuzzing scenario is because the first topology was only useful in the development stages, but not for the final test. The final purpose of these practical results is to be able to observe the behaviour of the two techniques used in distributed systems.

### 4.2.1. Results Obtained Using the Fuzzing Technique

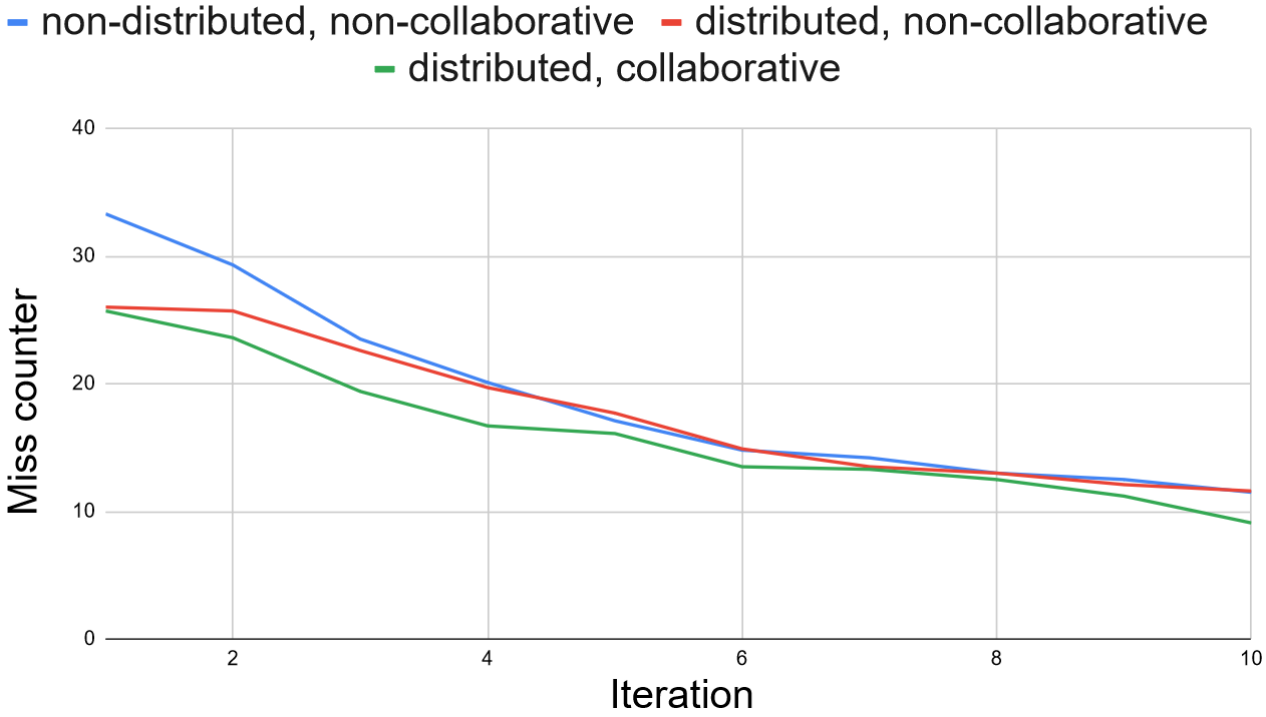


*Figure 4.1 Results obtained using the fuzzing technique*

In the first test scenario, the non-distributed and non-collaborative topology, the controller contains a single miss counter table which is used for each connected switch. After each iteration, the controller performs the fuzzing operation on the current timeout values with a random variation of maximum ten seconds. The second test scenario, the distributed and non-collaborative topology, implements the same algorithm on the control devices, with the difference that there is one switch connected to each control device, but the number of

sent packets is identical. The control devices implement the same algorithm, but since their number is multiplied by three, there will be more fuzzing and therefore there will be more chances to generate a more favourable rule set. The third test scenario, the distributed and collaborative topology, presents an extension of the previous scenario, in which the control devices share information about their current sets of switching rules and the number of misses generated by each rule for each traffic type. The main difference from the previous scenario is that there is less chance of generating temporary negative results, since control devices have the possibility to cancel their current rule set if another control device has shared a better performing rule set, thus avoiding the use of insufficiently good rule sets. Figure 4.1 shows the results obtained using the fuzzing technique.

### 4.2.2. Results Obtained Using the Federated Learning Technique

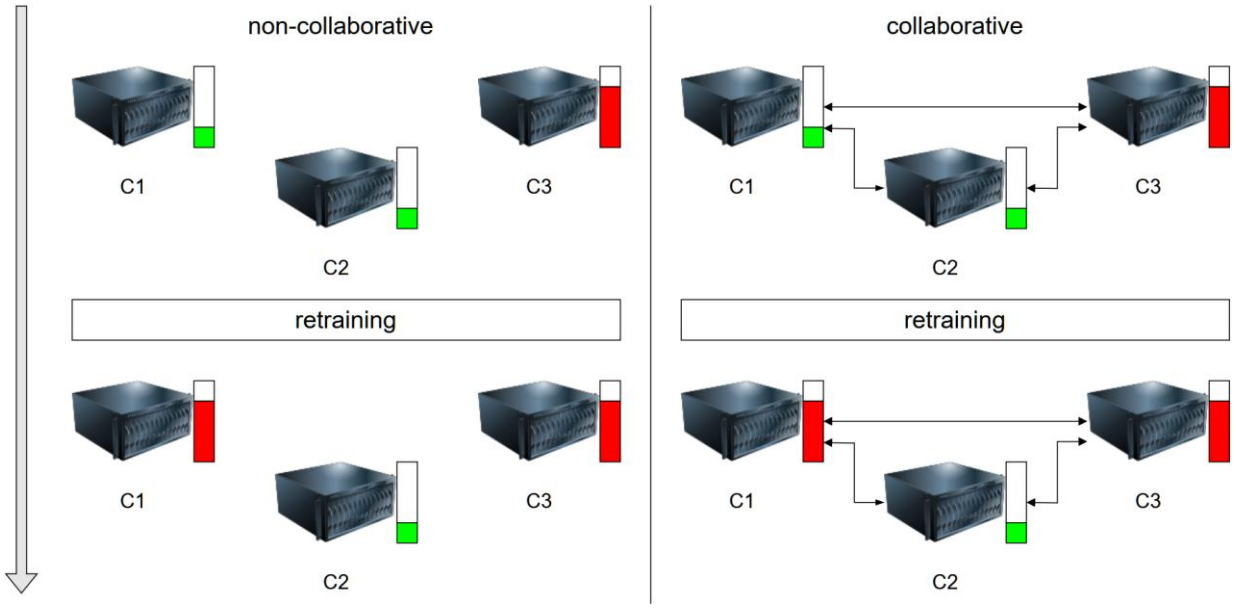


*Figure 4.2 Testing scenario for the decentralized federated learning technique*

Figure 4.2 shows the test scenario that was considered for the simulations using the decentralized federated learning technique. Two distributed systems with three control devices, one non-collaborative and the other collaborative, were considered. The figure indicates the congestion level encountered by each control device through the green or red loading bars, where the green color bar means low congestion and the red color bar means high congestion. It is important to explain that this is a simplified illustration, congestion is not actually encountered on the control device but on the switch, and this congestion is reflected on the control device by a higher number of OpenFlow packet_in packets received from the connected switch. So the green or red loading significance at the control device level represents the number of OpenFlow packet_in requests that are increasing as the attached switch receives network traffic for switching. The grey arrow indicates that there is a transition period between the states encountered in the simulation, and the three different states take place as follows: control devices encounter network traffic, C1 and C2 do not experience congestion, C3 experiences congestion; a retraining occurs after a certain period of time; in the third state, after retraining occurs on all control devices, C1 begins to experience congestion compared to the first state where this does not occur.

Figure 4.3 displays a graph view of the number of misses at 10-minute intervals in the simulation. The full simulation lasted a total of 2h and 30 minutes. The training took place in

each system approximately halfway through the run. The timing of training depends on the number of data sets collected for training, so this operation does not always occur at regular intervals. In the case of the non-training system, the C1, C2 and C3 control devices were re-trained at 1h 40m, 1h 50m and 1h 30m respectively. From the graph it can be seen that when switching to the third state, where congestion occurs on the C1 control device, an increase in the number of misses occurs. This happens due to the fact that more network traffic will inevitably generate more misses due to the lack of periods when there is no network traffic. It can be seen that in the case of the collaborative system there is always better performance by the fact that training data is shared before the congestion occurs. When entering the third state, the C1 controller will generate approximately the same number of misses as the C3 controller because the C1 controller has already managed to train based on the experience of the C3 controller. This is not the case for the non-collaborative system, in which the C1 controller will have at the beginning of the third state a number of misses approximately equal to the number of misses encountered from the C3 controller at the beginning of the first state.

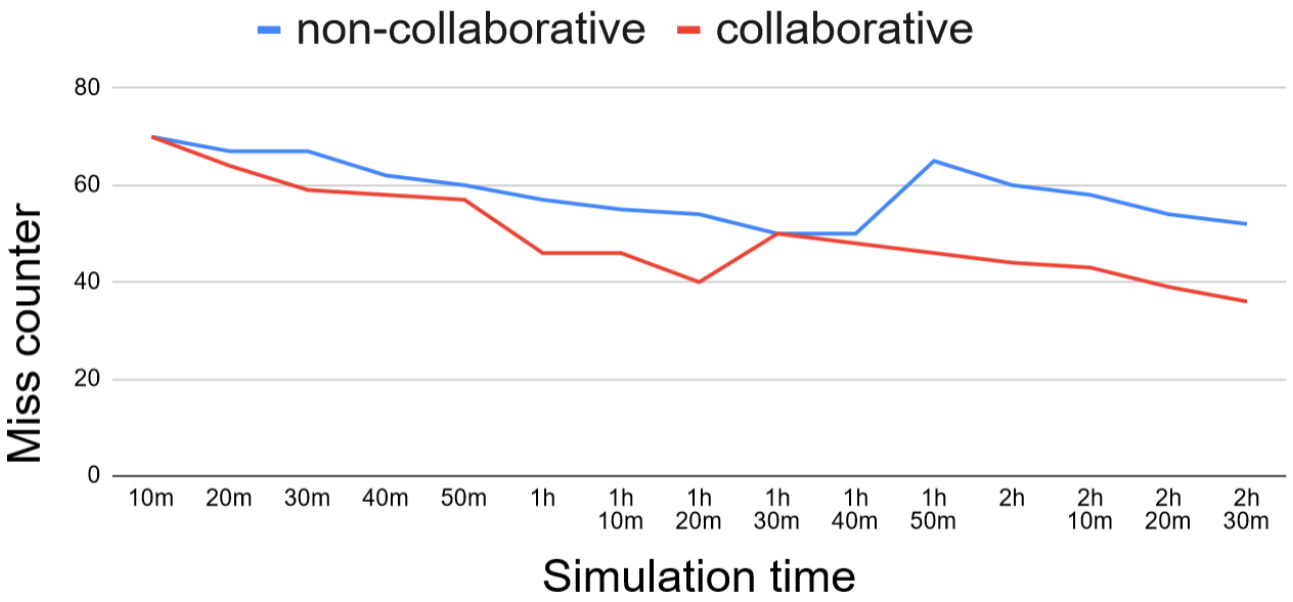


*Figure 4.3 Results obtained using the decentralized federated learning technique*

## 4.3. Discussions and Conclusions

The proposed fuzzing system shows an overall improvement in the average number of misses on switches in the data plane when using the distributed and collaborative system. The collaborative technique also generates the fastest decrease in the number of misses after the completion of the first iteration. If further explored, this mechanism based on distributed delegation of decision making can become a powerful solution in dynamic content environments (e.g. SDWSNs) to reduce manual network management effort. Automating the definition of QoS rules improves the user experience and, given current artificial intelligence technologies and capabilities, this area can be greatly improved. Continuing the proposal made in the fuzzing system, a system using decentralized federated learning technique was implemented which solves the same problem of decreasing the number of misses as different types of network traffic are encountered by switches in the data plane. The scenario simulated in the decentralized federated learning system is more complex and uses real data traffic, unlike the fuzzing scenario where network traffic was simulated. The simulations

demonstrate the effectiveness of the distributed and collaborative approach. It provides the best performance in terms of reducing OpenFlow communication packets between control devices and switches. In addition, self-adaptability is demonstrated in each scenario, but is shown to perform best in the collaborative scenarios.

In conclusion, the QoS in such a distributed and collaborative system is improved because if there are fewer interactions with OpenFlow packets, then the time spent waiting to make the decision via the OpenFlow protocol is replaced by actually switching data traffic in the network. Thus, the effective data rate is increased and decreases the time spent transferring control packets which also blocks essential data communication. This increases the data rate at the switch, so network performance is improved. Therefore, the primary objective proposed in 1.3.1.2 was achieved. Also, the complex implementation of the test system required different levels of implementation, but through the successful implementation of both the fuzzing system and the decentralized federated learning system, the secondary objectives proposed in 1.3.2.2 have also been achieved.

## 4.4. Results Dissemination

The results obtained in this chapter were disseminated in the following publication:

[118] S. Buzura, V. Dadarlat, B. Iancu, A. Peculea, E. Cebuc and R. Kovacs, "Self-adaptive Fuzzy QoS Algorithm for a Distributed Control Plane with Application in SDWSN", 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129922

# 5. Improving the QoS by Addressing Security Risks

## 5.1. System Design

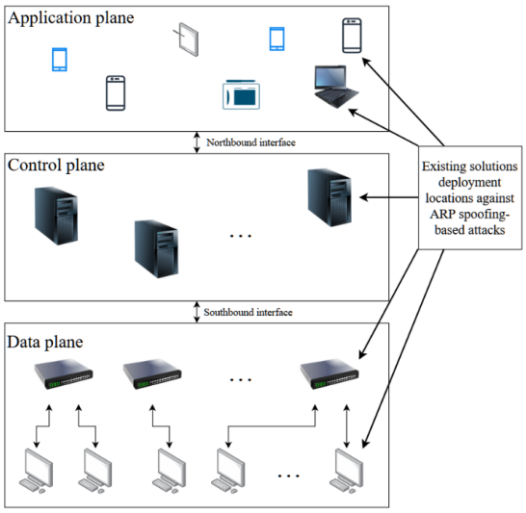


*Figure 5.1 Proposed implementation in SDN architecture. The execution of the solution on the data plane switches is shown*

Given the SDN architecture, it is important to note that the current solution operates in the data plane and is implemented as an extension to OpenVSwitch. Figure 5.1 graphically shows the location of the implementation relative to an SDN topology. This subchapter presents the system that was implemented to demonstrate the concepts and the proposed approach. Figure 5.2 shows the components of the solution behaving in a similar way to the Linux pipeline architecture, where information is passed on from one process to another via a pipe or a file. It is a sequential flow of operations, and each component generates output data that is considered input data for the next component. Using this approach, each component is easily replaced or enhanced, the only restriction being that the specific format constraints for input and output data must be maintained.

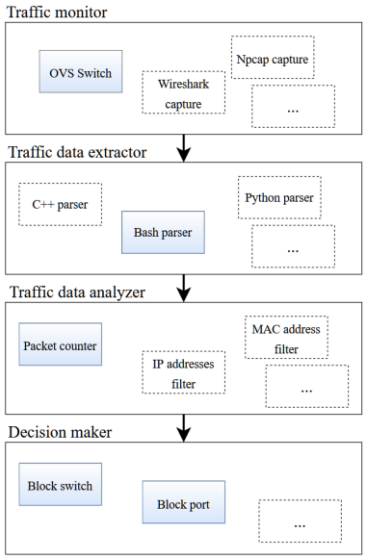


*Figure 5.2 The sequence of operations required to detect and mitigate attacks. The used components and possible implementation alternatives are highlighted*

Another important characteristic of this solution is that the sequence of operations is executed in a loop with a configurable time interval. The time interval parameter is stored in a configuration file which is read at run initialization. Thus, the solution is dynamic and provides parameterization at runtime, not at compile time. This is useful if the switch needs to be configured without recompiling any components. The configuration file also contains a parameter for the severity level passed to the decision maker component. The severity level can be used differently depending on the criticality level of the current network. Different decisions can be triggered in case of detecting an attack depending on the business domain where the network is used.

## 5.2. Practical Results

One important thing to point out is that the duration and mitigation time is actually the time it takes for the solution to execute the entire loop. Consequently, the detection and mitigation duration cannot be shorter than the execution of the solution. Therefore, the minimum detection and mitigation time of the attack is given by the execution time of the solution. As the experimental results demonstrate, the duration varies with the size of the network and the level of congestion. For low traffic scenarios, the average detection and mitigation time is approximately the same. However, as the number of network devices and traffic both increase, the computation time increases exponentially, as shown in Figure 5.3.

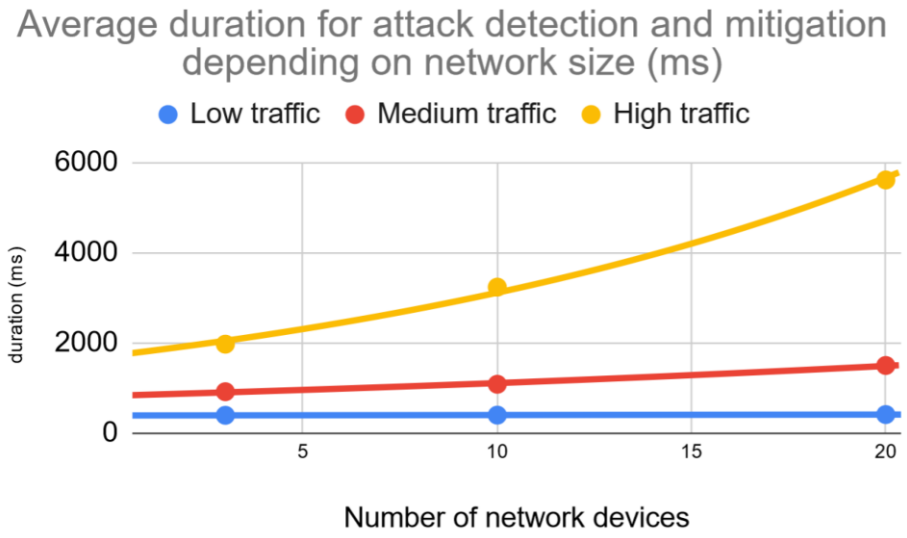


*Figure 5.3 Average time to detect and mitigate attacks in networks of different sizes with different levels of congestion. The figure shows the exponential increase in duration as the number of devices in the network increases, but also as the level of congestion increases*

This behaviour is caused by the fact that low traffic generates fewer OpenVSwitch flow entries, data that needs to be generated on the switch, which means that the next steps in the sequence do not receive as much input data for processing and do not generate significant additional processing time. As more traffic is generated, the required memory and computing hardware resources also increase. For medium traffic volume scenarios, an increase in detection and mitigation time is more noticeable as the network topology becomes more complex. Comparing the ten-device network with the three-device network, the processing time increases by 17.8%. Comparing the twenty-device network with the ten-device network, processing time increases by 38.3%. Therefore, an important finding that can be deduced is that, as more devices are in the network, the computational processing time increases

exponentially. In the case of high traffic volume scenarios, the increase in duration is again noticeable, as in the previous case where there is an average traffic volume. Comparing the ten-device network with the three-device network, the processing time increases by 63.9%. Comparing the network of twenty devices with the network of ten devices, the processing time increases by 73.3%. Again, a non-linear increase is visible, showing that the larger the network is, the slower the solution becomes.

As for the proposed distributed MitM attack, its detection and mitigation requires higher computational resources than the attack from a single device. The details presented show the case where the ARP spoofing is only shared between two devices; however, this can scale to multiple attackers being simultaneously coordinated. Since the detection must verify all available network devices in tuples of K arguments, the complexity of the solution increases to O(ˆK), n to the power of K, where n is the number of devices in the network and K is the number of coordinated attackers. Of course, more optimizations can be performed, such as marking some devices as trusted so that they are not inspected, thus reducing the complexity, but this is beyond the scope of the current study.

## 5.3. Discussions and Conclusions

In later stages of development, more experimentation can be done by replacing components in the plugin architecture and studying new components that have better performance. System performance can be extended by monitoring only certain devices based on their addresses, thus reducing the computation time, as fewer components will be inspected. Another idea for further development on the physical layer of networks is to perform tests in heterogeneous programmable networks where WiFi devices connect to a switch via wireless access points. The proposed solution already supports this configuration, as WiFi networks are processed identically to wired Ethernet networks (they have the same Layer 2 header), but further study could be done in terms of the latency of signal conversions in the physical layer (from radio waves to electrical signals), and an access point would allow multiple devices to be connected in a given location. This study would provide more information on the possibilities of implementing this solution in a real hardware environment. Another idea for further development is to extend this concept to a distributed system. A collaborative environment can be created in which different switches can share data on known MAC addresses of attackers, so that an attacker can be more easily found in different locations of the network. A collaborative environment can integrate artificial intelligence techniques, such as federated learning, to detect attacks based on more complex traffic patterns. Network traffic between Mininet devices was generated using the iPerf tool [119].

Further development of the current solution may overcome ideas of local complexity through software and attempts to increase the performance of a single network device. Therefore, future trials can be included in SDN architecture plans by using virtualization of network functions through network hypervisors. The study presented in [120] presents CoVisor, a network hypervisor that allows multiple control devices to coexist on the same device and allows sharing their best features for optimizing network functionality. Libera is also a network hypervisor that has the ability to create a dedicated network infrastructure for managing specific network traffic [121]. Such hypervisors as CoVisor and Libera could be used together to create a virtualized ecosystem that allows different policies and features to be applied on different network infrastructures. A more concrete use case related to the solution presented in this security article would be to implement different attack mitigation strategies in different network infrastructures. Simultaneous testing and performance measurement can take place in separately created network infrastructures, leading to a faster

method of deciding which is the best method for dealing with a specific attack. However, a disadvantage that hypervisors can have in the context of security is that functionality is moved to intermediate levels of the network, but attacks are always identified preferably as close as possible to the source of the attack. Thus, for critical use cases, it is better to operate at the edge of the network and not in the upper SDN layers after the information has already been passed through many intermediate devices. In such a scenario, the use of a network hypervisor is questionable and proper benchmarking and testing should be performed before implementing a network infrastructure solution.

Another concept that can be used together with virtualised and programmable ecosystems is that of application-specific programmable integrated circuits, in which functions at the software level of the application are moved to the hardware layer of the circuit. Thus, in theory, use cases that are implemented in the higher levels of the operating system can be moved to the hardware. Used together with the advantages offered by a network hypervisor, dynamic programmability can be configured in different network locations.

Software-defined networks have the advantage of being programmable and allow the development of many particular network use cases. In this regard, the current solution has focused on creating an extendable software architecture that can detect and mitigate attacks based on ARP spoofing techniques in Ethernet networks created in Mininet. The solution is built on the Linux pipeline architecture and a plugin system, in which a sequence of operations is mandatory. The sequence of operations contains different modules for data acquisition, relevant data extraction, data analysis and decision making. Each of the components is replaceable using a configuration file and, depending on the network use cases and desired severity levels, different actions can be taken upon attack detection. The fact that the components are replaceable allows the system to be easily extendable and configurable.

In conclusion, the main objective defined in 1.3.1.3. was achieved by improving network performance through detection and mitigation of security risks in software-defined networks and the desired effect of increasing the speed of essential data traffic has also been successfully achieved. The secondary objectives defined in 1.3.2.3. were also fully achieved through the following aspects: malicious traffic of ARP spoofing attacks was analysed in detail by performing MitM ARP spoofing attacks using the Ettercap utility in KaliLinux; the OpenVSwitch solution was extended with the ability to detect and mitigate ARP spoofing attacks; the metric for performance verification was chosen as the speed of data plane transfer of essential data.

## 5.4. Results Dissemination

The results obtained in this chapter were disseminated in the following publication:

[122] S. Buzura, M. Lehene, B. Iancu, V. Dadarlat "An Extendable Software Architecture for Mitigating ARP Spoofing-Based Attacks in SDN Data Plane Layer", Electronics 2022, 11, 1965. doi: https://doi.org/10.3390/electronics11131965 IF 2.69 Q3

# 6. Development of a Framework for Software-Defined Networking Simulations

## 6.1. The Framework in the Context of the Proposed Objectives

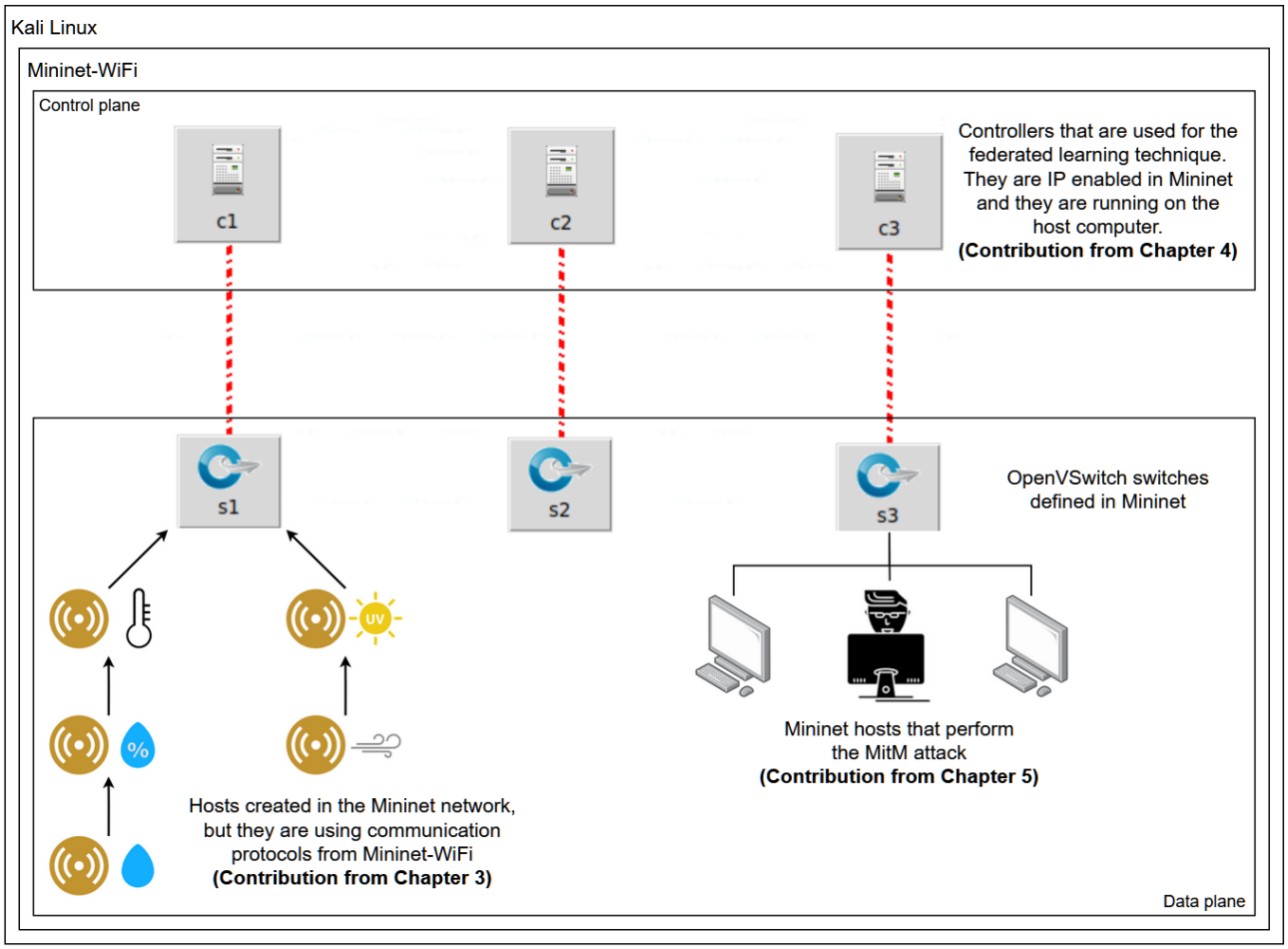


*Figure 6.1 The framework used in the context of the proposed objectives in this PhD thesis*

For the finality of this PhD thesis it is also necessary to incorporate the solutions of the main proposed objectives for the finality of the thesis into the same network. Thus, Figure 6.1 shows how all these solutions can be merged into a single test topology. The following components are identified: A Kali Linux operating system that is required to perform the MitM attack; the Mininet-WiFi emulator installed within the virtual machine that uses the Kali Linux operating system, it is important to remember that Mininet-WiFi contains both the wireless protocol extensions and the basic functionality provided by Mininet; the control devices are added to the Mininet network and are referenced by IP address, this means that it is not required to run the control solutions on the same operating system where the Mininet instance is running and in this case, they will be executed from the host machine using a Windows operating system where the Matlab exported machine learning library can be used; the wireless nodes that make up a sensor network are attached to switch s1, they can be coordinated to send or not send data depending on the cache-based optimization solution. Having said that, the framework is validated for use in simulation contexts with heterogeneous and interdisciplinary technologies and provides a complete solution in the

possibilities of studying the behavior of programmable networks addressing both QoS and security topics, offering numerous possibilities to study the proposed contributions to improve the performance of software-defined networks.

## 6.2. Testing Environment

At this point it is important to reiterate that the main objective of this chapter is to provide architectural and implementation details on how to build a hybrid environment with software and hardware simulation components. The experimental work is limited to the design for portability, as this context of use is the only experiment that can provide accurate and reusable measurement values for the literature. This is because the OpenFlow packets that are used in this experiment are transmitted over LAN and WAN network types that are commonly used in modern networks. Designs for scalability, parallel computations, and software update delivery depend heavily on the hardware resources of the Linux machine running Mininet and Mininet-WiFi, which are subjective from the perspective of the current work. In order to simulate the above design ideas, the required components are shown in Figure 6.2. The configuration consists of a PC running a Linux operating system, on which Mininet and Mininet-WiFi were installed, hardware components, which were attached directly to the PC via USB interfaces (Ubertooth One and Libelium Waspmote sensor) and an Ethernet interface (Toradex i.MX6 board). The advantage of the approach proposed above for duplicating traffic and simultaneously reading multiple data from multiple hardware devices is the possibility to extend it to large-scale simulations, where only one sensor of each type can be used. This drastically reduces the cost of simulation and implementation time.

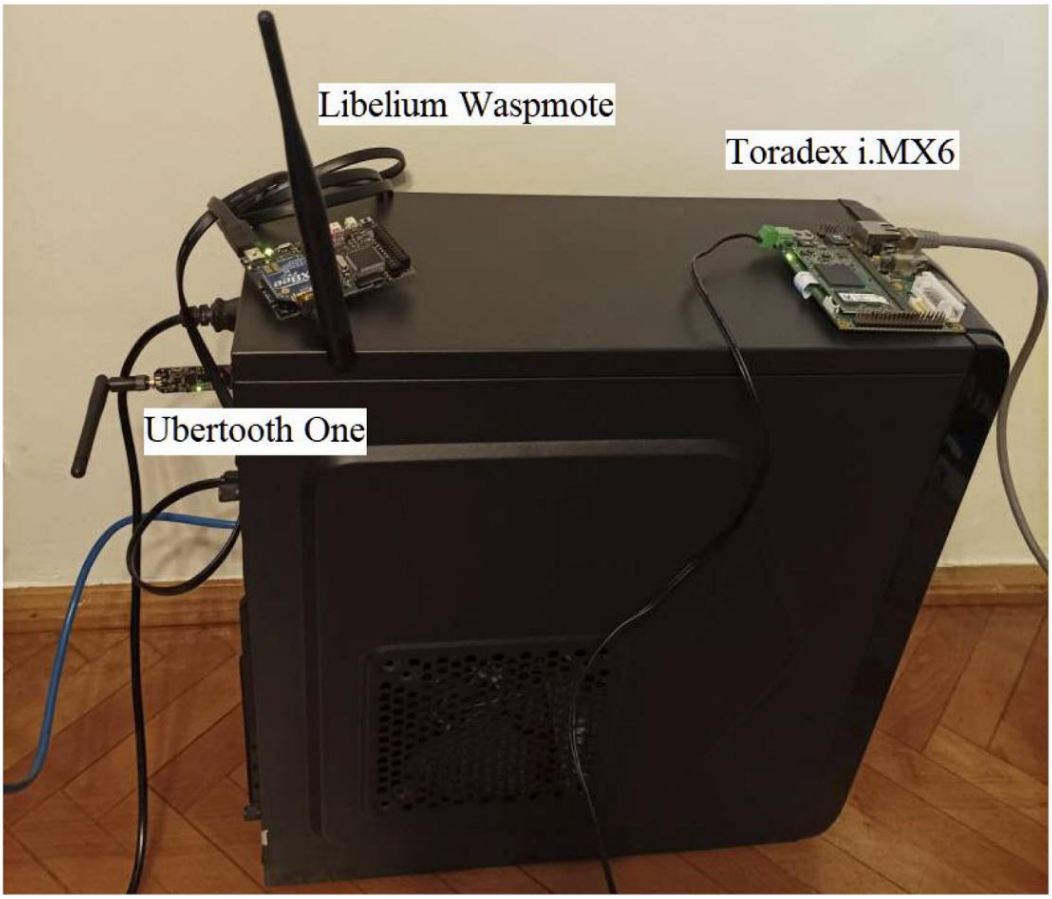


*Figure 6.2 Components used to create the hybrid software and hardware framework for the proposed simulations*

## 6.3. Discussions and Conclusions

The result of the research work that was in this chapter has created a hybrid software and hardware simulation framework for network testing. Four main design ideas were used, namely: design for scalability, design for parallel computation, design for portability, and design for software update on nodes of a wireless sensor network. These design ideas provide the possibility to test the system in different use cases. The use contexts presented in the paper are: data collection from sensors in a WSN, network security, comparative testing of different placement locations of SDN control devices, and the delivery of software updates in an SDN environment. The SDN test system that has been implemented contains contributions in the data plane and the control plane. The ultimate goal of using the SDN paradigm is to bring benefits to the application plane, which are visible from the improvements made in the lower SDN planes. To validate the proposed hardware software hybrid simulation approach, measurements were performed in the scenario designed for portability. The goal was to study the effect of positioning an SDN control device in different locations in the LAN containing the data plane, but also in a different location in the Internet. The results showed that the impact of data transfer between the data plane switch and the SDN control device is negligible when the SDN control device is located in the same LAN (regardless of the number of intermediate switches), but data transfer is significantly slower when the SDN control device is located in a different Layer 3 network.

The main novelty proposed to introduce real-time reading of sensor data is limited to a number of sensors equal to the number of interfaces available on the computer, thus deducing the exact number of hardware devices that can connect to Mininet. These interfaces include the following: USB, serial, Bluetooth, etc. The number of available interfaces can be increased somewhat by adding external adapters or USB hubs that allow more devices to be connected. However, this number remains limited, and reading data from multiple interfaces consumes other hardware resources (processor processing power and available memory) that are needed for experimental calculations. As far as measurements on OpenFlow packets are concerned, they were performed only on the OpenFlow packet_in, which has a size between approximately 600 and 1400 bytes. The OpenFlow packet packet_in contains the Ethernet frame for which the request is made, plus a few additional bytes that are specific to the OpenFlow protocol. With a reduced packet size, the transmission times will have approximately similar values. To create a complete evaluation, the entire OpenFlow set of packet types could be measured for data transmission. Interpreting the results presented, the variation in transfer time is small, and this is due to the fact that OpenFlow is an application-level protocol running over TCP. Establishing a connection takes up a considerable part of this transmission time with the three-step TCP handshake.

Considering the wide range of SDN domain applications and large-scale environments, creating a real test environment is a complex and costly task. To address this limitation, a hybrid software and hardware simulation framework was proposed. By integrating simulated data with real-time sensor data (from hardware devices) in a Mininet emulator, realistic simulations were generated and validated in different real use cases. In addition, new datasets can be easily generated for specific scenarios by saving the data captured from real-time sensors. The newly generated datasets and scenarios can be easily replicated in other research projects, thus contributing to the body of science.

In conclusion, the main objective defined in 1.3.1.4. was achieved by improving the performance of software-defined networks, allowing the creation of complete simulation scenarios that can concurrently test different functionalities using hardware devices. The sub-objectives defined in 1.3.2.4. were also fully achieved through the following aspects: the

heterogeneity of software-defined networks is highlighted by the hybrid framework combining different software solutions with different hardware devices; different programmability concepts in different programming languages have been used to create the framework; programmable networks have been addressed from all architectural planes; different networks created in Mininet and Mininet-WiFi have been extended with data read in real time from hardware devices; protocols from all levels of the TCP/IP protocol stack have been used in the development of the framework, thus providing a complete solution for the created framework.

## 6.4. Results Dissemination

The work developed in this chapter was disseminated in the following publications:

[123] S. Buzura, A. Peculea, B. Iancu, E. Cebuc, V. Dadarlat, R. Kovacs "A Hybrid Software and Hardware SDN Simulation Testbed", Sensors 2023, 23, 490. https://doi.org/10.3390/s23010490 IF: 3.847 Q2

[124] S. Buzura, V. Lazar, B. Iancu, A. Peculea and V. Dadarlat, "Using Software-Defined Networking Technology for Delivering Software Updates to Wireless Sensor Networks," 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), Iasi, Romania, 2021, pp. 1-6, doi: 10.1109/RoEduNet54112.2021.9637720.

[125] S. Buzura, V. Dadarlat, A. Peculea, H. Bertrand and R. Chevalier, "Simulation Framework for 6LoWPAN Networks Using Mininet-WiFi," 2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2022, pp. 1-5, doi: 10.1109/AQTR55203.2022.9802017.

[126] S. Buzura, A. Suciu, E. Cebuc, B. Iancu and V. Dadarlat, "Development Framework for Simulating Routing Behavior in Software-Defined Wireless Networks," 2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet), Sovata, Romania, 2022, pp. 1-6, doi: 10.1109/RoEduNet57163.2022.9921101.

# 7. Conclusions

## 7.1. General Conclusions

This PhD thesis provides a development framework in the areas of traffic engineering, collaborative systems development, artificial intelligence and security, with applicability in software-defined networking. Personal contributions have been implemented at all levels of programmable networks, with the ultimate goal of improving the QoS provided by such systems. The resulting framework provides a complete solution for conducting research studies in software-defined networks using the aforementioned areas.

The field of computer networks is an interdisciplinary field where different notions of programmability, management and security often overlap in research. Because of this, a very good theoretical understanding of these areas is required, but relevant practical experience is also needed to develop solutions and conduct research studies in the field of computer networks. For this reason, the complexity of the field and consequently of the current thesis, does not only have specific technical features of local implementation of the proposed functionalities but also has a complex overview of integrating interdisciplinary concepts with the role of finding compatibility between them.

Regarding the manner in which the presents results of this PhD thesis were achieved, it is important to mention that some of the research topics were carried out over a long period of time, even over several years, until the results were successfully disseminated. The processes used were incremental with different collaborators (co-authors of the published articles) and the way of working was published in [127] at an educational conference. Continuing this educational theme, the laboratory activities for the Computer Networks subject (in the English teaching language) were also updated with topics studied and practically experienced during this PhD thesis. Thus, for the laboratory activities document referenced in [128], two completely new papers on socket programming and ARP spoofing attacks were added to prepare students both with general technical notions and for various research topics.

## 7.2. Originality and Innovative Contributions of the Thesis

Chapter 3 on traffic engineering proposed a contribution to performance improvement in SDWSNs. In this context, performance improvement refers to improving energy efficiency and increasing network lifetime. Energy efficiency in an SDWSN has been improved through the use of transmitted data content analysis techniques and adaptive broadcast data transmission. Energy efficiency is improved by reducing the number of packets transmitted in the software-defined network data plane. The content analysis technique was implemented at the level of individual programmable sensors in the data plane, where hardware resources are limited on network devices. In the final solution, adaptive data transmission is moved to the control plane and replaces data transmission in the data plane. The implemented system increases the network lifetime as the number of packets in the WSN layer is reduced, thus improving QoS by reducing traffic volume and avoiding congestion. QoE is improved by proactively forwarding data from the control plane to the application plane. Thus, the main objective proposed in 1.3.1.1 was achieved. The secondary objectives proposed in 1.3.2.1. were also achieved.

Chapter 4 with the topic of implementing a distributed and collaborative control scheme proposed a contribution to a more efficient definition of OpenFlow switching rules. In this context, performance improvement refers to reducing control traffic in the network, thus favouring essential data traffic and consequently increasing the throughput of essential data. The proposed problem was solved in two ways, using fuzzing technique and using decentralized federated learning technique. Both techniques have a positive result by reducing the control traffic in the network, but an advantage can be mentioned in favor of the federated learning technique, namely, that the generated rules better adjust to the actual traffic duration, and these rules do not remain active during traffic-free periods. Thus, the main objective proposed in 1.3.1.2. were achieved. The secondary objectives proposed in 1.3.2.2. were also achieved.

Chapter 5 concerning security proposes contributions to improve QoS in an SDN by removing malicious traffic from the network, thus increasing the relevant data traffic transmitted. Performance is thus improved by increasing the transmission speed of essential data. A solution for detecting and mitigating ARP spoofing attacks in the data plane of an SDN was implemented. The proposed software architecture is able to accommodate different operating environments (based on OpenVSwitch, npcap, etc.) and can integrate multiple software technologies for data processing. The implementation of the solution is done strictly in the data plane of an SDN without interaction with the control plane. The implemented solution was evaluated in networks of different sizes and with different levels of congestion, and the evaluation also shows the impact the solution has on the volume of data transmitted over the network in scenarios with and without the MitM attack running. Finally, a new distributed ARP spoofing attack is conceptualized and a behavioural analysis of the attacker's actions is performed. A solution for this new type of attack is also provided, thus offering a novelty with respect to the current state of knowledge. Thus, the main objective proposed in 1.3.1.3. was achieved. The secondary objectives proposed in 1.3.2.3. were achieved.

Chapter 6 concerning the creation of a framework that incorporates all the previous contributions proposes implementation software contributions for the simulation of software-defined networks. Various programming languages and socket programming techniques are used in Windows and Linux operating systems, but also in networks created using Mininet and Mininet-WiFi emulators. Different networking use cases are explored, such as: simulating large-scale networks; updating software on production-installed sensors without having wired access to them; simulating routing techniques in SDWSN; simulations for testing different algorithms in parallel on the same data; test scenarios for portability. Tests are carried out in networks with different constraints, such as: IPv4 wired networks; IPv6/6LowPan unwired networks; networks using services provided by the IPFS distributed system; networks extending the OpenFlow protocol and software solutions using this protocol, the Pox solution. Finally, a hybrid simulation environment is proposed using networks built in Mininet and Mininet-WiFi, which are extended with real-time data received from sensors physically connected to the host computer through different interfaces (USB, serial). All these contributions demonstrate the flexibility and heterogeneity of the SDN paradigm. The developed software simulation environments can be easily extended in the future to further studies in the field of programmable networks. These contributions also provide more implementation details, which are often missing in scientific works in the literature, but are necessary for the replication of studies and their validation. Thus, the main objective proposed in 1.3.1.4. was achieved. The secondary objectives proposed in 1.3.2.4. were also achieved.

In conclusion, the proposed objectives were achieved through the performed research work.

## 7.3. Further Development

Software-defined networks have gained popularity especially in datacenter or campus networks. A newer context is the use of the SDN paradigm in WANs, creating the SD-WAN topology. A possible further development is the creation of a framework based on the Mininet-Optical emulator which, in addition to the features of an SDN, also includes a simulator of the physical processes of data transmission over optical fibre. Different studies can be done with different parameters both at the upper levels of the TCP/IP protocol stack (network, Internet and application levels) and at the lower level of access to the environment. Thus, different test topologies can be created to serve different research studies in different application domains.

Another possible further development is in the area of security, namely checking the susceptibility of devices in an SDN to various attacks, such as the Nethammer attack [129], which is actually the Rowhammer attack caused by a high volume of data traffic received from the network. This attack primarily involves a software solution that reads data received from the network interface into the same memory area (identified by a pointer). And secondly, the hardware device running this software solution must be using a type of RAM memory that is susceptible to this attack (a memory that does not completely isolate electromagnetic interference between transistors). Further study would involve investigating different SDN solutions and performing stress and penetration tests on them to cause the attack to occur. SDN solutions that provide access to source code can be further analysed and understood for how they read incoming data from the network. If such vulnerabilities are found, solutions can be proposed for remediation, which will lead to improvements in the quality offered by existing solutions. For a better understanding of these vulnerabilities the practical applications in [130] and [131] can be studied, they detail the processes that take place at the hardware level.

# REFERENCES

[1] L. Peterson, C. Cascone, B. O'Connor, T. Vachuska, and B. Davie, Software-Defined Networks: A Systems Approach. Available at: https://sdn.systemsapproach.org/index.html (accessed on 01.03.2020)

[2] SDN standard specified in RFC 7426 „Software-Defined Networking (SDN): Layers and Architecture Terminology". Available at: https://datatracker.ietf.org/doc/html/rfc7426 (accessed on 01.03.2020)

[3] Nam, H.; Kim, K.-H.; Kim, J.Y.; Schulzrinne, H. Towards QoE-aware video streaming using SDN. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014

[4] Ezdiani, S.; Acharyya, I.S.; Sivakumar, S.; Al-Anbuky, A. Wireless Sensor Network Softwarization: Towards WSN Adaptive QoS. IEEE Internet Things J. 2017, 4, 1517–1527

[5] Benzekki, K.; El Fergougui, A.; Elbelrhiti Elalaoui, A. Software-defined networking (SDN): A survey. Secur. Commun. Netw. 2016, 9, 5803–5833

[6] Misra, S.; Bera, S.; Achuthananda, M.P.; Pal, S.K.; Obaidat, M.S. Situation-Aware Protocol Switching in Software-Defined Wireless Sensor Network Systems. IEEE Syst. J. 2017, 12, 1–8

[7] Kadel, R.; Ahmed, K.; Nepal, A. Adaptive error control code implementation framework for software defined wireless sensor network (SDWSN). In Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, Australia, 22–24 November 2017

[8] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. Comput. Commun. Rev. 2008, 38, 69–74

[9] Xiang, W.; Wang, N.; Zhou, Y. An Energy-Efficient Routing Algorithm for Software-Defined Wireless Sensor Networks. IEEE Sens. J. 2016, 16, 7393–7400

[10] Alam, M.M.; Berder, O.; Menard, D.; Sentieys, O. Traffic-aware adaptive wake-up-interval for preamble sampling MAC protocols of WSN. In Proceedings of the Third International Workshop on Cross Layer Design, Rennes, France, 30 November–1 December 2011

[11] Liu, D.; Chen, B.; Yang, C.; Molisch, A.F. Caching at the wireless edge: Design aspects, challenges, and future directions. IEEE Commun. Mag. 2016, 54, 22–28

[12] Lei, F.; Cai, J.; Dai, Q.; Zhao, H. Deep Learning Based Proactive Caching for Effective WSN-Enabled Vision Applications. Complexity 2019, 1–12

[13] Hu, W.; Qiu, Z.; Wang, H.; Yan, L. A Real-time scheduling algorithm for on-demand wireless XML data broadcasting. J. Netw. Comput. Appl. 2016, 68, 151–163

[14] Wei, Y.; Ma, X.; Yang, N.; Chen, Y. Energy-Saving Traffic Scheduling in Hybrid Software Defined Wireless Rechargeable Sensor Networks. Sensors 2017, 17, 2126

[15] Li, G.; Guo, S.; Yang, Y.; Yang, Y. Traffic Load Minimization in Software Defined Wireless Sensor Networks. IEEE Internet Things J. 2018, 5, 1370–1378

[16] Modares, H.; Salleh, R.; Moravejosharieh, A. Overview of Security Issues in Wireless Sensor Networks. In Proceedings of the Third International Conference on Computational Intelligence, Modelling & Simulation, Langkawi, Malaysia, 20–22 September 2011

[17] Sadowski, C.; Levin, G. Simhash: Hash-Based Similarity Detection; Technical Report, Google; University of California: Santa Cruz, CA, USA, 2007

[18] Wang, Z.; Zhang, M.; Gao, X.; Wang, W.; Li, X. A clustering WSN routing protocol based on node energy and multipath. Clust. Comput. 2017, 22

[19] Abbasi, A.A.; Younis, M. A survey on clustering algorithms for wireless sensor networks. Comput. Commun. 2007, 30, 2826–2841

[20] Xu, L.; Collier, R.; O'Hare, G. A Survey of Clustering Techniques in WSNs and Consideration of the Challenges of Applying Such to 5G IoT Scenarios. IEEE Internet Things J. 2017, 4, 1229–1249

[21] Townsend, L. Wireless Sensor Network Clustering with Machine Learning. Ph.D. Thesis, Nova Southeastern University, Fort Lauderdale, FL, USA, 2018

[22] M. Karakus, and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey", Journal of Network and Computer Applications. 80, 2016, 10.1016/j.jnca.2016.12.019

[23] Adam Zarek, OpenFlow Timeouts Demystified, University of Toronto, Toronto, Ontario, Canada, 2012

[24] V. M. Vishnu, P. Manjunath, "SeC-SDWSN: Secure cluster-based SDWSN environment for QoS guaranteed routing in three-tier architecture", International Journal of Communication Systems, e4020, 2019, doi:10.1002/dac.4020

[25] Manzanares-Lopez, P.; Malgosa-Sanahuja, J.; Muñoz-Gea, J.P. A Software-Defined Networking Framework to Provide Dynamic QoS Management in IEEE 802.11 Networks. Sensors 2018, 18, 2247

[26] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined Wireless Sensor Networks," 2016 IEEE International Symposium on Consumer Electronics (ISCE), Sao Paulo, 2016, pp. 85-86.doi: 10.1109/ISCE.2016.7797384

[27] Oktian, Y. E., Lee, S., Lee, H., & Lam, J. (2017). Distributed SDN controller system: A survey on design choice. Computer Networks, 121, 100–111. doi:10.1016/j.comnet.2017.04.038

[28] S. T. V. Pasca, S. S. P. Kodali and K. Kataoka, “AMPS: Application aware multipath flow routing using machine learning in SDN”, Twenty-third National Conference on Communications (NCC), Chennai, 2017, pp. 1-6. doi: 10.1109/NCC.2017.8077095

[29] J. Liu and Q. Xu, "Machine Learning in Software Defined Network," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 1114-1120. doi: 10.1109/ITNEC.2019.8729331

[30] R. Thupae, B. Isong, N. Gasela and A. M. Abu-Mahfouz, "Machine Learning Techniques for Traffic Identification and Classification in SDWSN: A Survey," IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, 2018, pp. 4645-4650. doi: 10.1109/IECON.2018.8591178

[31] B. Letswamotse, K. Modieginyane and R. Malekian, “SDN Based QoS Provision in WSN Technologies”, arXiv 2017, arXiv:abs/1702.08164

[32] P. Wang, S. Lin and M. Luo, "cs," 2016 IEEE International Conference on Services Computing (SCC), San Francisco, CA, 2016, pp. 760-765. doi: 10.1109/SCC.2016.133

[33] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A survey on distributed machine learning,” arXiv preprint arXiv:1912.09789, 2019

[34] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data” 2016, arXiv:1602.05629. Available at: https://arxiv.org/abs/1602.05629 (accessed on 01.06.2020)

[35] Zhang Y., Zhang X., Li X. (2021) Towards Efficient and Privacy-Preserving Service QoS Prediction with Federated Learning. In: Gao H., Wang X., Iqbal M., Yin Y., Yin J., Gu N. (eds) Collaborative Computing: Networking, Applications and Worksharing. CollaborateCom 2020. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 350. Springer, Cham

[36] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. ACM TIST, 10(2):12:1–12:19, 2019

[37] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In International Conference on Machine Learning, pages 560–569, 2018

[38] F. Sattler, S. Wiedemann, K. -R. Müller and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 9, pp. 3400-3413, Sept. 2020, doi: 10.1109/TNNLS.2019.2944481

[39] A. Sacco, F. Esposito and G. Marchetto, "A Federated Learning Approach to Routing in Challenged SDN-Enabled Edge Networks," 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 2020, pp. 150-154, doi:10.1109/NetSoft48620.2020.9165506

[40] Xu, C.; Mao, Y. An Improved Traffic Congestion Monitoring System Based on Federated Learning. Information 2020, 11, 365

[41] Mun, H.; Lee, Y. Internet Traffic Classification with Federated Learning. Electronics 2021, 10, 27

[42] OpenFlow protocol. Specifications available at: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed on 01.06.2020)

[43] E. Kim, S. Lee, Y. Choi, M. Shin and H. Kim, "A flow entry management scheme for reducing controller overhead," 16th International Conference on Advanced Communication Technology, Pyeongchang, 2014, pp. 754-757. doi: 10.1109/ICACT.2014.6779063

[44] A. V. Atli, M. S. Uluderya, S. Tatlicioglu, B. Gorkemli and A. M. Balci, "Protecting SDN controller with per-flow buffering inside OpenFlow switches," 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, 2017, pp. 1-5, doi: 10.1109/BlackSeaCom.2017.8277662

[45] H. I. Kobo, G. P. Hancke and A. M. Abu-Mahfouz, "Towards a distributed control system for software defined Wireless Sensor Networks," IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, 2017, pp. 6125-6130. Doi: 10.1109/IECON.2017.8217064

[46] Huang, S.; Griffioen, J. Network Hypervisors: Managing the Emerging SDN Chaos. In Proceedings of the 2013 22nd International Conference on Computer Communication and Networks (ICCCN), Nassau, Bahamas, 30 July–2 August 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–7

[47] Chica, J.C.; Imbachi, J.C.; Botero Vega, J.F. Security in SDN: A Comprehensive Survey. J. Netw. Comput. Appl. 2020, 159, 102595

[48] Mallik, A.; Ahsan, A.; Shahadat, M.M.Z.; Tsou, J.-C. Man-In-The-Middle-Attack: Understanding in Simple Words. Int. J. Data Netw. Sci. 2019, 3, 77–92

[49] Shah, Z.; Cosgrove, S. Mitigating ARP Cache Poisoning Attack in Software-Defined Networking (SDN): A Survey. Electronics 2019, 8, 1095

[50] Nehra, A.; Tripathi, M.; Gaur, M.S. FICUR: Employing SDN Programmability to Secure ARP. In Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017; pp. 1–8

[51] Furukawa, M.; Kuroda, K.; Ogawa, T.; Miyaho, N. Highly secure communication service architecture using SDN switch. In Proceedings of the 2015 10th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), Colombo, Sri Lanka, 22–25 September 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–3

[52] Solomon, N. Mitigating Layer 2 Attacks: Re-Thinking the Division of Labor. Master's Thesis, School of Computer Science, The Interdisciplinary Center, Reichman University, Herzliya, Israel, 2015

[53] Al-Somaidai, M.B. Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation. Am. J. Softw. Eng. Appl. 2014, 3, 74

[54] Dhawan, M.; Poddar, R.; Mahajan, K.; Mann, V. SPHINX: Detecting Security Attacks in Software-Defined Networks. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 8–11 February 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 8–11

[55] Hareesh, I.; Prasanna, S.; Vijayalakshmi, M.; Shalinie, S.M. Anomaly detection system based on analysis of packet header and payload histograms. In Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, India, 3–5 June 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 412–416

[56] Girdler, T.; Vassilakis, V.G. Implementing an Intrusion Detection and Prevention System Using Software-Defined Networking: Defending against ARP Spoofing Attacks and Blacklisted MAC Addresses. Comput. Electr. Eng. 2021, 90, 106990

[57] Munther, M.N.; Hashim, F.; Abdul Latiff, N.A.; Alezabi, K.A.; Liew, J.T. Scalable and Secure SDN Based Ethernet Architecture by Suppressing Broadcast Traffic. Egypt. Inform. J. 2021, 23, 113–126

[58] Shaghaghi, A.; Kaafar, M.A.; Buyya, R.; Jha, S. Software-defined network (SDN) data plane security: Issues, solutions, and future directions. In Handbook of Computer Networks and Cyber Security; Gupta, B., Perez, G., Agrawal, D., Gupta, D., Eds.; Springer: Cham, Switzerland, 2020; pp. 341–387.

[59] Rangisetti, A.K.; Dwivedi, R.; Singh, P. Denial of ARP Spoofing in SDN and NFV Enabled Cloud-Fog-Edge Platforms. Clust. Comput. 2021, 24, 3147–3172

[60] Open Floodlight SDN controller. Available at: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview (accessed on 17.05.2021)

[61] Lin, T.-Y.; Wu, J.-P.; Hung, P.-H.; Shao, C.-H.; Wang, Y.-T.; Cai, Y.-Z.; Tsai, M.-H. Mitigating SYN flooding Attack and ARP Spoofing in SDN Data Plane. In Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea, 22–25 September 2020; pp. 114–119

[62] Ryu SDN controller. Available at: https://ryu-sdn.org/ (accessed on 17.05.2021)

[63] Amin, A.A.M.M.; Mahamud, M.S. An Alternative Approach of Mitigating ARP Based Man-in-the-Middle Attack Using Client Site Bash Script. In Proceedings of the 2019 6th International Conference on Electrical and Electronics Engineering (ICEEE), Istanbul, Turkey, 16–17 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 112–115

[64] Yang, G.; Shin, C.; Yoo, Y.; Yoo, C. A Case for SDN-based Network Virtualization. In Proceedings of the 29th International Symposium on Modeling, Analysis, and Simulation of

Computer and Telecommunication Systems (MASCOTS), Houston, TX, USA, 3–5 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8

[65] Mininet network emulator. Available at: http://mininet.org (accessed on 01.03.2020)

[66] Fontes, R.R.; Afzal, S.; Brito, S.H.B.; Santos, M.A.S.; Rothenberg, C.E. Mininet-WiFi: Emulating Software-defined Wireless Networks. In Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 9–13 November 2015; pp. 384–389

[67] Mininet-Optical emulator. Avilable at: https://mininet-optical.org/ (accessed on 11.06.2022)

[68] R. Nagarathna and S. M. Shalinie, "SLAMHHA: A supervised learning approach to mitigate host location hijacking attack on SDN controllers," 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), 2017, pp. 1-7, doi: 10.1109/ICSCN.2017.8085680

[69] POX SDN controller. Available at: https://noxrepo.github.io/pox-doc/html (accessed on 17.05.2021)

[70] A. Bumb, B. Iancu and E. Cebuc, "Extending Cooja simulator with real weather and soil data," 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2018, pp. 1-5, doi: 10.1109/ROEDUNET.2018.8514130

[71] Cooja network simulator. Available at: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja (accessed on 17.05.2021)

[72] H. A. A. Al-Kashoash, M. Hafeez and A. H. Kemp, "Congestion Control for 6LoWPAN Networks: A Game Theoretic Framework," in IEEE Internet of Things Journal, vol. 4, no. 3, pp. 760-771, June 2017, doi: 10.1109/JIOT.2017.2666269

[73] A. HAKA, R. VASILEV, V. ALEKSIEVA and H. VALCHANOV, "Simulation Framework for Building of 6LoWPAN Network," 2019 16th Conference on Electrical Machines, Drives and Power Systems (ELMA), 2019, pp. 1-5, doi: 10.1109/ELMA.2019.8771633

[74] F. F. J. Lasso, K. Clarke and A. Nirmalathas, "A software-defined networking framework for IoT based on 6LoWPAN," 2018 Wireless Telecommunications Symposium (WTS), 2018, pp. 1-7, doi: 10.1109/WTS.2018.8363938

[75] Younus, M.U.; Islam, S.u.; Kim, S.W. Proposition and Real-Time Implementation of an Energy-Aware Routing Protocol for a Soft-ware Defined Wireless Sensor Network. Sensors 2019, 19, 2739

[76] S. Buzura, V. Dadarlat, A. Peculea, B. Iancu and E. Cebuc, "Simulations framework for network congestion avoidance algorithms using the OMNeT++ IDE," 2013 11th RoEduNet International Conference, 2013, pp. 1-8, doi: 10.1109/RoEduNet.2013.6511758

[77] OMNeT++ network simulator. Available at: https://omnetpp.org/ (accessed on 17.05.2021)

[78] Lunagariya, D.; Goswami, B. A Comparative Performance Analysis of Stellar SDN Controllers using Emulators. In Proceedings of the 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 19–20 February 2021; pp. 1–9

[79] Kazi, N.M.; Suralkar, S.R.; Bhadade, U.S. Evaluating the Performance of POX and RYU SDN Controllers Using Mininet. In Communications in Computer and Information Science Book Series (CCIS, Volume 1483); Venugopal, K.R., Shenoy, P.D., Buyya, R., Patnaik, L.M., Iyengar, S.S., Eds.; Springer: Cham, Switzerland, 2021; Volume 1483, pp. 181–191

[80] Cui, X.; Huang, X.; Ma, Y.; Meng, Q. A Load Balancing Routing Mechanism Based on SDWSN in Smart City. Electronics 2019, 8, 273

[81] J. Izquierdo-Zaragoza, A. Fernandez-Gambin, J. Pedreno-Manresa and P. Pavon-Marino, "Leveraging Net2Plan planning tool for network orchestration in OpenDaylight," 2014 International Conference on Smart Communications in Network Technologies (SaCoNeT), 2014, pp. 1-6, doi: 10.1109/SaCoNeT.2014.6867770

[82] Net2Plan tool. Available at: http://www.net2plan.com/ (accessed on 06.03.2022)

[83] Haseeb, K.; Ud Din, I.; Almogren, A.; Islam, N. An Energy Efficient and Secure IoT-Based WSN Framework: An Application to Smart Agriculture. Sensors 2020, 20, 2081

[84] M. S. Hossen, M. H. Rahman, M. Al-Mustanjid, M. A. Shakil Nobin and M. A. Habib, "Enhancing Quality of Service in SDN based on Multi-path Routing Optimization with DFS," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-5, doi: 10.1109/STI47673.2019.9068057

[85] Npcap software library for the C and C++ programming languages. Available at: https://npcap.com/ (accessed on 06.03.2022)

[86] SharpPcap software library for the C# programming language. Available at: https://www.nuget.org/packages/SharpPcap (accessed on 06.03.2022)

[87] Pcap4J software library for the Java programming language. Available at: https://www.pcap4j.org/ (accessed on 06.03.2022)

[88] jNetPcap software library for the Java programming language. Available at: https://www.jnetpcap.com/ (accessed on 06.03.2022)

[89] S. B. Bose and S. S. Sujatha, "Fuzzy Logic based QoS Management and Monitoring System for Cloud Computing", 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 2020, pp. 920-924, doi: 10.1109/ICISS49785.2020.9315874

[90] Bosmans, S.; Mercelis, S.; Denil, J.; Hellinckx, P. Testing IoT Systems Using a Hybrid Simulation Based Testing Approach. Computing 2018, 101, 857–872

[91] Ulbricht, M.; Acevedo, J.; Krdoyan, S.; Fitzek, F.H.P. Emulation vs. Reality: Hardware/Software Co-Design in Emulated and Real Time-sensitive Networks. In Proceedings of the European Wireless 2021 26th European Wireless Conference, Verona, Italy, 10–12 November 2021; pp. 1–7

[92] Tang, J.; Chen, X.; Zhu, X.; Zhu, F. Dynamic Reallocation Model of Multiple Unmanned Aerial Vehicle Tasks in Emergent Adjustment Scenarios. IEEE Trans. Aerosp. Electron. Syst. 2022, 1–43

[93] Huang, T.; Yu, F.R.; Zhang, C.; Liu, J.; Zhang, J.; Liu, Y. A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges. IEEE Commun. Surv. Tutor. 2017, 19, 891–917

[94] Zhao, Z.; Wu, B. Scalable SDN Architecture with Distributed Placement of Controllers for WAN. Concurr. Comput. Pract. Exp. 2017, 29, e4030

[95] Das, T.; Sridharan, V.; Gurusamy, M. A Survey on Controller Placement in SDN. IEEE Commun. Surv. Tutor. 2020, 22, 472–503

[96] Alwasel, K.; Calheiros, R.N.; Garg, S.; Buyya, R.; Pathan, M.; Georgakopoulos, D.; Ranjan, R. BigDataSDNSim: A Simulator for Analyzing Big Data Applications in Software-Defined Cloud Data Centers. Softw. Pract. Exp. 2020, 51, 893–920

[97] Alomari, A.; Subramaniam, S.K.; Samian, N.; Latip, R.; Zukarnain, Z. Resource Management in SDN-Based Cloud and SDN-Based Fog Computing: Taxonomy Study. Symmetry 2021, 13, 734

[98] Gonzalez, N.M.; Carvalho, T.C.M.D.B.; Miers, C.C. Cloud Resource Management: Towards Efficient Execution of Large-Scale Scientific Applications and Workflows on Complex Infrastructures. J. Cloud Comput. 2017, 6, 1–20

[99] Hegazy, A.; El-Aasser, M. Network Security Challenges and Countermeasures in SDN Environments. In Proceedings of the 2021 Eighth International Conference on Software Defined Systems (SDS), Gandia, Spain, 6–9 December 2021; pp. 1–8

[100] Alwasel, K.; Jha, D.N.; Hernandez, E.; Puthal, D.; Barika, M.; Varghese, B.; Garg, S.K.; James, P.; Zomaya, A.; Morgan, G.; et al. IoTSim-SDWAN: A Simulation Framework for Interconnecting Distributed Datacenters over Software-Defined Wide Area Network (SD-WAN). J. Parallel Distrib. Comput. 2020, 143, 17–35

[101] Uddin, M.; Mukherjee, S.; Chang, H.; Lakshman, T.V. SDN-Based Service Automation for IoT. In Proceedings of the 2017 IEEE 25th International Conference on Network Protocols (ICNP), Toronto, ON, Canada, 10–13 October 2017; pp. 1–10

[102] H. I. Kobo A. M. Abu-Mahfouz and G. P. Hancke "Fragmentation-Based Distributed Control System for Software-Defined Wireless Sensor Networks" IEEE Transactions on Industrial Informatics vol. 15 no. 2 pp. 901-910 Feb. 2019

[103] W. Munawar M. H. Alizai O. Landsiedel and K. Wehrle "Dynamic TinyOS: Modular and Transparent Incremental Code-Updates for Sensor Networks" 2010 IEEE International Conference on Communications pp. 1-6 2010

[104] C. Han R. Kumar R. Shea and M. Srivastava "Sensor network software update management: a survey" International Journal of Network Management vol. 15 no. 4 pp. 283-294 July 2005

[105] S. Brown and C. Sreenan "Software Updating in Wireless Sensor Networks: A Survey and Lacunae“ Journal of Sensor and Actuator Networks vol. 2 no. 4 pp. 717-760 Nov. 2013

[106] J. Lee "Patch Transporter: Incentivized Decentralized Software Patch System for WSN and IoT Environments" Sensors vol. 18 no. 574 2018

[107] S. Agrawal R. Roman M.L. Das A. Mathuria and J. Lopez "A Novel Key Update Protocol in Mobile Sensor Networks" Notes in Computer Science vol. 7671 2012

[108] P. Radmand A. Talevski S. Petersen and S. Carlsen "Comparison of industrial WSN standards" 4th IEEE International Conference on Digital Ecosystems and Technologies pp. 632-637 2010]

[109] Zigbee protocol. Specifications available at: https://web.archive.org/web/20130627172453/http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx (accessed on 17.05.2021)

[110] WirelessHART technology. Specifications available at: https://www.fieldcommgroup.org/sites/default/files/imce_files/technology/documents/WirelessHART_system_eng_guide.pdf (accessed on 17.05.2021)

[111] ISA.100 standard. Specifications available at: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa100 (accessed on 17.05.2021)

[112] Gast, M.S. 802.11 Wireless Networks: The Definitive Guide; O’Reilly: Chicago, IL, USA, 2002; pp. 218–221

[113] TCP protocol specified in RFC 793. Available at: https://tools.ietf.org/html/rfc793 (accessed on 01.06.2020)

[114] Meteoblue historical data. Available at: https://www.meteoblue.com/en/historyplus (accessed on 01.06.2020)

[115] Qt framework over the C++ programming language. Available at: https://www.qt.io/ (accessed on 15.01.2020)

[116] Mansfield, K.C.; Antonakos, J.L. Computer Networking from LANs to WANs: Hardware, Software, and Security; Cengage Learning: Boston, MA, USA, 2010; p. 501

[117] S. Buzura, B. Iancu, V. Dadarlat, A. Peculea, E. Cebuc "Optimizations for Energy Efficiency in Software-Defined Wireless Sensor Networks", Sensors 2020, 20, 4779. doi: https://doi.org/10.3390/s20174779

[118] S. Buzura, V. Dadarlat, B. Iancu, A. Peculea, E. Cebuc and R. Kovacs, "Self-adaptive Fuzzy QoS Algorithm for a Distributed Control Plane with Application in SDWSN," 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129922

[119] iPerf tool. Available at: https://iperf.fr/ (accessed on 17.05.2021)

[120] Jin, X.; Gossels, J.; Rexford, J.; Walker, D. CoVisor: A compositional hypervisor for software-defined networks. In Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15), Oakland, CA, USA, 4–6 May 2015; USENIX-The Advanced Computing Systems Association: Berkley, CA, USA, 2015; pp. 87–101

[121] Yang, G.; Yu, B.; Jin, H.; Yoo, C. Libera for Programmable Network Virtualization. IEEE Commun. Mag. 2020, 58, 38–44

[122] S. Buzura, M. Lehene, B. Iancu, V. Dadarlat "An Extendable Software Architecture for Mitigating ARP Spoofing-Based Attacks in SDN Data Plane Layer", Electronics 2022, 11, 1965. doi: https://doi.org/10.3390/electronics11131965

[123] S. Buzura, A. Peculea, B. Iancu, E. Cebuc, V. Dadarlat, R. Kovacs "A Hybrid Software and Hardware SDN Simulation Testbed", Sensors 2023, 23, 490. https://doi.org/10.3390/s23010490

[124] S. Buzura, V. Lazar, B. Iancu, A. Peculea and V. Dadarlat, "Using Software-Defined Networking Technology for Delivering Software Updates to Wireless Sensor Networks," 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), Iasi, Romania, 2021, pp. 1-6, doi: 10.1109/RoEduNet54112.2021.9637720

[125] S. Buzura, V. Dadarlat, A. Peculea, H. Bertrand and R. Chevalier, "Simulation Framework for 6LoWPAN Networks Using Mininet-WiFi," 2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2022, pp. 1-5, doi: 10.1109/AQTR55203.2022.9802017.

[126] S. Buzura, A. Suciu, E. Cebuc, B. Iancu and V. Dadarlat, "Development Framework for Simulating Routing Behavior in Software-Defined Wireless Networks," 2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet), Sovata, Romania, 2022, pp. 1-6, doi: 10.1109/RoEduNet57163.2022.9921101.

[127] S. Buzura, B. Iancu and V. Dadarlat, "Creating Educational and Research Tools for QoS-Focused Software-Defined Networking Projects", 2022 IEEE Global Engineering Education Conference (EDUCON), 2022, pp. 1179-1182, doi: 10.1109/EDUCON52537.2022.9766749.

[128] A. Peculea, B. Iancu, S. Buzura, V. Ratiu, coordonatori: V. Dadarlat, E. Cebuc, Computer networks. Practical activities, Ed. U.T. PRESS, 978-606-737-633-3, 2023

[129] M. Lipp et al., "Nethammer: Inducing Rowhammer Faults through Network Requests," 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 2020, pp. 710-719, doi: 10.1109/EuroSPW51379.2020.00102.

[130] A. Peculea, B. Iancu, V. Dadarlat, S. Buzura, Circuite analogice și numerice. Aplicatii practice, Ed. U.T. PRESS, 978-606-737-458-2, 2020

[131] A. Peculea, B. Iancu, V. Dadarlat, S. Buzura, Analog and Digital Circuits. Practical Applications, Ed. U.T. PRESS, 978-606-737-459-9, 2020

# LIST OF PUBLICATIONS

## Articles published in ISI indexed journals:

**S. Buzura**, B. Iancu, V. Dadarlat, A. Peculea, E. Cebuc "Optimizations for Energy Efficiency in Software-Defined Wireless Sensor Networks", Sensors 2020, 20, 4779. doi: https://doi.org/10.3390/s20174779 **IF: 3.576 Q1**

**S. Buzura**, A. Peculea, B. Iancu, E. Cebuc, V. Dadarlat, R. Kovacs "A Hybrid Software and Hardware SDN Simulation Testbed", Sensors 2023, 23, 490. https://doi.org/10.3390/s23010490 **IF: 3.847 Q2**

**S. Buzura**, M. Lehene, B. Iancu, V. Dadarlat "An Extendable Software Architecture for Mitigating ARP Spoofing-Based Attacks in SDN Data Plane Layer", Electronics 2022, 11, 1965. doi: https://doi.org/10.3390/electronics11131965 **IF: 2.69 Q3**

## Articles presented at ISI Proceedings indexed conferences:

**S. Buzura**, V. Dadarlat, A. Peculea, H. Bertrand and R. Chevalier, "Simulation Framework for 6LoWPAN Networks Using Mininet-WiFi", 2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 2022, pp. 1-5, doi: 10.1109/AQTR55203.2022.9802017.

**S. Buzura**, B. Iancu and V. Dadarlat, "Creating Educational and Research Tools for QoS-Focused Software-Defined Networking Projects", 2022 IEEE Global Engineering Education Conference (EDUCON), 2022, pp. 1179-1182, doi: 10.1109/EDUCON52537.2022.9766749.

**S. Buzura**, V. Dadarlat, B. Iancu, A. Peculea, E. Cebuc and R. Kovacs, "Self-adaptive Fuzzy QoS Algorithm for a Distributed Control Plane with Application in SDWSN", 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129922.

R. A. Kovacs, B. Iancu, V. T. Dadarlat, E. Cebuc and **S. Buzura**, "Extending K-cover genetic algorithm for efficient energy consumption in WSNs", 2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2019, pp. 1-6, doi: 10.1109/ROEDUNET.2019.8909581.

## Articles presented at conferences indexed in international databases:

**S. Buzura**, A. Suciu, E. Cebuc, B. Iancu and V. Dadarlat, "Development Framework for Simulating Routing Behavior in Software-Defined Wireless Networks", 2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet), 2022, pp. 1-6, doi: 10.1109/RoEduNet57163.2022.9921101.

Sauchea, **S. Buzura**, A. Peculea, E. Cebuc and V. Dadarlat, "Open Source Network Management System Based on SharpPcap for QoS and Security Policies", 2022 21st

RoEduNet Conference: Networking in Education and Research (RoEduNet), 2022, pp. 1-5, doi: 10.1109/RoEduNet57163.2022.9921027.

R. Kovacs, B. Iancu, V. Dadarlat, **S. Buzura**, A. Peculea and E. Cebuc, "A collaborative game theory approach for determining the feasibility of a shared AS blockchain infrastructure", 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2021, pp. 1-6, doi: 10.1109/RoEduNet54112.2021.9637711.

**S. Buzura**, V. Lazar, B. Iancu, A. Peculea and V. Dadarlat, "Using Software-Defined Networking Technology for Delivering Software Updates to Wireless Sensor Networks", 2021 20th RoEduNet Conference: - Networking in Education and Research (RoEduNet), 2021, pp. 1-6, doi: 10.1109/RoEduNet54112.2021.9637720.

V. Lazar, **S. Buzura**, B. Iancu and V. Dadarlat, "Anomaly Detection in Software Defined Wireless Sensor Networks Using Recurrent Neural Networks", 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP), 2021, pp. 19-24, doi: 10.1109/ICCP53602.2021.9733669.

**S. Buzura**, B. Iancu, V. Dadarlat, "A System Proposal for Collecting Medical Data from Heterogeneous Sensors Using Software-Defined Networks", 7th International Conference on Advancements of Medicine and Health Care through Technology, Springer International Publishing, MediTech, 2020, pp. 282-289, doi: https://doi.org/10.1007/978-3-030-93564-1_32.

## Article presented at an ISI Proceedings indexed conference before the PhD stage:

**S. Buzura,** V. Dadarlat, A. Peculea, B. Iancu and E. Cebuc, "Simulations framework for network congestion avoidance algorithms using the OMNeT++ IDE", 2013 11th RoEduNet International Conference, 2013, pp. 1-8, doi: 10.1109/RoEduNet.2013.6511758.

## Laboratory activity practical guides published with the research group CNP (Computer Networks and Protocols)

A. Peculea, B. Iancu, **S. Buzura**, V. Rațiu, coordonatori: V. Dadarlat, E. Cebuc, Computer networks. Practical activities, Ed. U.T. PRESS, 978-606-737-633-3, 2023. Available at: https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/633-3.pdf (accessed on: 19.05.2023)

A. Peculea, B. Iancu, V. Dadarlat, **S. Buzura**, Circuite analogice și numerice. Aplicatii practice, Ed. U.T. PRESS, 978-606-737-458-2, 2020. Available at: https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/458-2.pdf (accessed on: 19.05.2023)

A. Peculea, B. Iancu, V. Dadarlat, **S. Buzura**, Analog and Digital Circuits. Practical Applications, Ed. U.T. PRESS, 978-606-737-459-9, 2020. Available at: https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/459-9.pdf (accessed on: 19.05.2023)