information

Article

Assessment of SDN Controllers in Wireless Environment Using a Multi-Criteria Technique

Ioannis Koulouras, Ilias Bobotsaris, Spiridoula V. Margariti, Eleftherios Stergiou and Chrysostomos Stylios *

Department of Informatics and Telecommunication, University of Ioannina, 47150 Arta, Greece; i.koulouras@uoi.gr (I.K.); i.bobotsaris@uoi.gr (I.B.); smargar@uoi.gr (S.V.M.); ster@uoi.gr (E.S.)
* Correspondence: stylios@uoi.gr

Abstract: Software-defined network (SDN) technology can offer wireless networks the advantages of simplified control and network management. This SDN subdomain technology is called the software-defined wireless network (SDWN). In this study, the performance of four controllers in an SDWN environment is assessed, since the controller is the most significant component of the entire network. Using the Mininet-WiFi platform, the performance of each controller is evaluated in terms of throughput, latency, jitter, and packet loss. Moreover, a multi-criteria evaluation is introduced and applied to provide a fair comparison between SDWNs. This study provides an appropriate configuration of SDWNs that is useful for network engineering and can be used for SDWNs performance optimization.

Keywords: software-defined wireless network; evaluation methodology; multi-criteria evaluation; controller; performance evaluation; Mininet-WiFi

Citation: Koulouras, I.; Bobotsaris, I.; Margariti, S.V.; Stergiou, E.; Stylios, C. Assessment of SDN Controllers in Wireless Environment Using a Multi-Criteria Technique. *Information* **2023**, *14*, 476. https://doi.org/10.3390/info14090476

Academic Editor: Lin Chen

Received: 7 July 2023
Revised: 20 August 2023
Accepted: 25 August 2023
Published: 28 August 2023

1. Introduction

Software-defined networking (SDN) is a design approach that separates the data and control planes, providing flexibility to the forwarding process and intelligence to the routing process. Moreover, it unifies the control of all data plane elements implemented on the control plane through controller software [1–3]. Since the controller manages the entire SDN, it is a prominent factor that affects network performance. Thus, the SDN approach allows the establishment of new decision-making mechanisms through the controller, which collects all network-related data that are acquired.

Recently, the architecture and characteristics of SDNs have been transferred to wireless networks, resulting in the emergence of software-defined wireless networks (SDWNs). An SDWN can be considered as the next-generation network standard, capable of facilitating centralized network management. Due to the presence of heterogeneous devices, diverse data formats, and various protocols in wireless networks, different architectures have been introduced for SDWNs, each possessing distinctive capabilities [4]. It allows competing wireless network operators to engage in negotiations and collaboratively distribute accessible wireless networking resources [5,6], while enabling the operation and management of paths and traffic without the need for human oversight. The SDWN standard also establishes a centralized environment where the separation of hardware and software takes place, with a centralized controller overseeing the entire network. Utilizing batch flow rules, this approach facilitates efficient communication and enables quick and secure network operations. Additionally, the use of an open interface in this technology helps to mitigate vendor dependencies.

In general, a wireless channel can suffer from severe fluctuations and will never be completely stable due to multi-path fading, frequent path loss, and the capture effect, among other issues. As a result, the user's requests may fail to reach the desired destinations. If the requested content is not received, the user may re-initiate communication to retrieve it. Frequent request transmissions can cause network congestion, delays, and increased packet loss. To overcome the above-mentioned issues, the application of SDWN is proposed and considered. However, it poses several challenges, such as the choice of the appropriate controller determination, which is critical if designing cost-effective wireless solutions with efficient resource utilization, and good manageability is a priority.

SDWN technologies are currently not widely used or well understood; numerous SDN controllers exist, most of which are capable of being integrated into these networks. Each of these controllers has several features and different properties and supports various protocols for southbound and northbound interfaces that affect their performance. In this work, an extensive series of experiments was conducted on common SDN controllers (ONOS, Ryu, POX, and Open Daylight). This effort yielded results for four important performance factors for each controller and various populations of devices. Using an extension of the Mininet emulator, the Mininet-WiFi emulator, and exploiting its intrinsic characteristics, we tested the SDN controllers with the primary aim of obtaining a network performance that satisfies the device population within the quality constraints, and an extensive series of experiments was conducted.

There have been many studies that have evaluated networks in terms of several performance indicators, e.g., throughput, latency, or other metrics, by considering them as separate factors. Because the performance factors of SDNs are in general independent of each other and each of them has its own quantifiable field and effect on the network evaluation process, a multi-criteria evaluation mechanism is required.

Hence, in this paper, to go beyond the performance findings for independent factors—following the abovementioned multi-object methods—a multi-criteria methodology is presented and applied to specific SDWNs. Thus, the main contributions of the proposed work can be described as follows:

- The efficiency of SDWNs is assessed using extensive simulations in the Mininet-WiFi emulator, which provides values for four performance factors.
- A multi-criteria assessment approach is explained and applied to SDWNs.
- The extracted results reveal the best choice of controller (in terms of performance factors) for a given population of devices. This methodology could be a useful tool for SDWN administrators, network designers, and engineers.

The rest of this paper is organized as follows: Section 2 provides background knowledge concerning SDNs, OpenFlow for SDWNs, the Mininet emulator, and the Mininet-WiFi emulator, describes the implementation of performance evaluation experiments, and explains the multi-object evaluation mechanism. In Section 3, the performance of SDN controllers is compared, and the general performance factor results are illustrated and discussed. Finally, Section 4 concludes the paper and provides ideas for future work.

2. Materials and Methods

Conducting performance assessments across SDN controllers may be supported by simulation, emulation, and/or other tools once the environment and evaluation measures have been specified. In this section, we present the key studies on SDN controllers' performance evaluation, introduce the under-assessed controllers, describe the performance metrics, and describe the experimental environment.

2.1. Literature Review

Numerous studies have assessed network behavior by individually examining throughput, latency, and other performance metrics. Aljundi et al. [7] and Stergiou et al. [8] presented evaluations of multi-stage interconnection networks and multiple input, multiple output (MIMO) systems, respectively. They used typical simulation-based multi-objective methods for network performance evaluation as a time-efficient method able to be implemented given a sufficient quantity of values for each performance factor. However, in SDN networks, performance is related to the controller, which has the responsibility to reallocate tasks, manage nodes, and optimize the network's operations [9]. Due to the large number of available SDN controllers, the research community strives hard to find the suitable controller that meets the needs of each type of network, as, for example, in wireless networks.

Some works focus on comparing a small number of open SDN controllers using performance metrics and simple network topologies, while others use complex schemes to decide the best controller.

In the first category, Assegie and Nair [10] evaluated the performance of SDWN architectures using the Mininet-WiFi emulator to test the performance of TCP traffic bandwidth for various devices that form single or linear topologies. Chaurasia et al. [11] compared the performance and scalability of Mininet-WiFi and the IT-SDN framework, differentiated by their underlying southbound interfaces, for software-defined wireless networks. Although the results showed that IT-SDN outperformed Mininet-WiFi in the relative metrics performed, the community still uses Mininet-WiFi because of the accumulated experience with this emulator. Mamushiane et al. [12] evaluated the performance of popular open-source controllers (e.g., ONOS, Ryu, Floodlight, and OpenDayLight) in terms of latency, throughput, scalability, and other metrics using an OpenFlow benchmarking tool called Cbench. This approach is limited to examining a few performance parameters. Their study concluded that the OpenDayLight controller has rich functionality, the ONOS controller achieves the highest throughput, and Ryu exhibits the best latency. In a more recent study for the same controllers in Fat-tree topologies, similar behavior was observed regarding the performance of SDN controllers [13]. However, the performance results depend on the scale of the network. In [14], the authors also used the Mininet-WiFi emulator and conducted an experimental evaluation of four SDN controllers (ONOS, POX, Ryu, and Floodlight) in terms of jitter and throughput versus time in seconds and delay versus the number of packets. Nevertheless, the effect of the size of the network in terms of the number of devices is not visible; therefore, the results are of limited value. Additionally, each factor is calculated separately for each controller, and there is no single evaluation index.

Zhu et al. [9] presented an extended comparison between several SDN controllers on a wired network, using various benchmark tools such as CBench, PktBlaster, and OFNet. They consider WSNs specialized networks and point out the lack of controllers for wireless environments. Furthermore, they noticed that the high performance of a controller (e.g., high throughput of ODL and ONOS) comes at a price that translates into either higher latency or the consumption of more resources and depends on their architecture (e.g., multi-threaded controllers). Therefore, performance evaluation cannot focus on individual performance metrics.

Several researchers in the second category have worked in this direction. In [15], Khondoker et al. used an analytical method called "Analytical Hierarchy Process (AHP)" to evaluate controllers' performance in SDN networks. According to this method, multiple criteria are set from the beginning, and priorities are set among those criteria concerned with the controllers. Then, a dimensional comparison matrix is constructed that under mathematical processing results in a final vector. If the value of the final vector is not satisfactory, then the priorities are changed, and the calculations are repeated. Applying the AHP model to the five most popular SDN controllers revealed the Ryu controller as the winner. However, the AHP is a complex, subjective method, and its results are differentiated according to the given working context.

Although there are many studies on SDN controller performance analysis in terms of wired networks, studies on controller effectiveness in wireless networks are scarce. Thus, through this work, an assessment of SDN controllers in SDWN networks was attempted. Four popular open-source controllers were selected for comparison in terms of four performance metrics (throughput, latency, jitter, and packet loss). These controllers were selected for investigation essentially because of the availability of the source code, community interest, and compatibility with current Linux distributions. Furthermore, a fast, multi-objective method to evaluate controllers in SDN networks was analytically presented and applied.

2.2. SDN/SDWN Networks

Today, as traditional wireless networks expand in scale, traffic data forwarded by network devices (access points, switches, and routers) are becoming increasingly voluminous, making it challenging to obtain real-time information on the status of the entire network. The recently developed SDN technology [6] and its adoption in the framework of wireless networks offer a remedy for this problem.

SDNs and wireless network technology have been combined to satisfy the requirements for centralized control and to implement policies on the communication channel [11] related to modulation format, latency, power, etc. [16]. The issues of poor management effectiveness and challenging control that are associated with the dynamic nature of wireless networks seem to be resolved by SDWNs. The controller of a wireless network can obtain the changes in topological structure and state of the entire network thanks to SDWN technology and decide on the establishment of flow rules at the data layer [17].

By using a network controller, SDWNs can virtualize, integrate, and manage all their network resources in a uniform manner. They also employ a northbound interface to offer upper-layer applications on-demand network resource and service allocation (Figure 1).

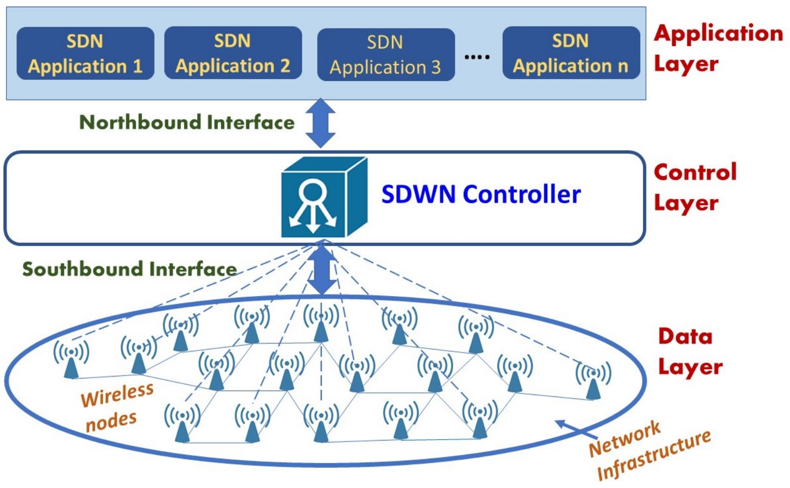


Figure 1. The general architecture of a software-defined wireless network (SDWN).

For network engineers to select the best controller in accordance with the requirements that must be met, it is important for them to analyze the performance of various types of controllers, research their features, and determine the significance of each feature's effect on the overall performance of the network. Although numerous studies have compared the performances of various types of controllers in wired networks, there have not been nearly as many of these studies for wireless networks [18].

In this paper, the performances of various controllers (POX, Ryu, Open Daylight, and ONOS) were investigated, and their performance factors (throughput, latency, packet loss, and jitter) were evaluated for different SDWN sizes. The Mininet-WiFi platform was used as a tool to emulate SDWNs and relevant experiments were carried out.

2.3. Mininet-WiFi

A free tool for modelling networks called Mininet [19] is suitable for emulating SDNs. Using Mininet makes it possible to easily establish a virtual network. An extension of Mininet, Mininet-WiFi, is a testing platform for SDWN. It retains the functionality of Mininet and additionally provides virtualized elements for emulating wireless networks, such as wireless nodes (e.g., access point, wireless stations) [20,21]. Via Mininet, the OpenFlow protocol can also be supported and used for communication between the data forwarding layer and the control layer. The Mininet-WiFi software can handle a variety of topologies for various SDWN scenarios [22].

The Mininet-WiFi aims to promote research on wireless networks and SDWNs by providing realistic simulations in a fully controlled environment [23,24]. Due to the availability of a variety of radio signal propagation types, Mininet-WiFi users can test and measure various performance factors and the signal propagation [22].

2.4. OpenFlow Protocol and SDN Controllers

2.4.1. OpenFlow Protocol

The OpenFlow protocol is mostly employed to connect OpenFlow network devices with the OpenFlow controller via the southbound interface of the SDN [25,26]. The OpenFlow establishes a physical secure channel to support Controller-to-Switch and vice versa communication using an exclusive transmission control protocol (TCP) link that acts as an interface for sending and receiving control messages. TLS-based asymmetrical encryption is used to protect this communication. In controlled environments (e.g., data centers) the use of security protocol (TLS) is optional.

2.4.2. SDN Controllers

An SDN controller is an application that provides flow control for better network management and application performance. Protocols are used by the SDN controller platform, which normally run on a server, to dictate packet transmission on devices. SDN/SDWN controllers minimize manual settings for individual network devices by directing traffic in accordance with forwarding policies set up by a network operator.

The fact that the centralized SDN controller is aware of all available network paths and can reroute packets based on traffic requirements is one of its main advantages. The controller may automatically change traffic flows and alert network administrators of heavy-loaded lines or links due to its insight into the network.

Enterprises can utilize multiple controllers to add redundancy. A setup with three controllers is a frequently used configuration for both open-source and commercial SDN/SDWN alternatives. This redundancy makes it possible for the network to function normally even if connectivity is lost or a controller becomes incapacitated.

Securing the controller is essential for any SDN/SDWN, since it serves as a single point of failure. Since the entire network is accessible to whoever acquires access to the controller, to guarantee that only authorized users have access, network administrators need to develop security and authentication procedures [27]. Below, four very common controllers that were used in the experiments that were carried out are described. These controllers are the ONOS, Ryu, POX, and Open Daylight (ODL) controllers.

- **ONOS controller:** The Open Network Operating System (ONOS) controller is a Java-based and open-source, frequently deployed under many research and testing contexts. The ONOS controller has a web interface that is accessible online. The user can quickly change the network architecture using this interface. The ONOS controller has rich features that lead to high performance, availability, and scalability. It is easy to deploy and comes with comprehensive documentation [28] while acting "as an extensible, modular, and distributed SDN controller" [29]. ONOS uses the Replicated Agreement for Fault-Tolerant (RAFT) mechanism for switching and mapping to its key controller. The central database of ONOS contains the data for this mechanism. The leader node initiates the process of scanning the network for updates and sends these changes to the nodes that are closest to it to maintain coherence throughout its cluster. The entire network architecture, controller roles (slave or master), switches, hosts, flows, pathways, and activities are all stored when considering a network that was created by ONOS. Additionally, the web-based graphical user interface is useful for configuring and viewing statistical data [30].
- **Ryu controller:** Ryu [31] is an open, Python-based SDN controller. All versions of OpenFlow and other southbound APIs are supported by Ryu. Ryu's multi-threading technique significantly improves performance, particularly under heavy loads [32]. Ryu has a relatively straightforward web-based Graphical User Interface (GUI) that

shows topology and flow data [33]. The Ryu controller's characteristics include seamless accessibility, the ability to control many processes, and the ability to monitor OpenFlow operations via a virtual local area network that relies on state transfer (REST). Nevertheless, according to many metrics, it is still regarded as having subpar performance compared to modern controllers.

- **POX controller:** The POX controller is also a Python-based open-source controller. It can run on Linux, Mac OS X, and Windows operating systems. POX does not have an official graphical user interface. Since it is written in Python, it performs rather poorly compared to other controllers building on compiled languages such as Java or C++. Some functions of the POX controller are inaccessible in certain systems. Even though POX runs on various platforms, most of its features are available on the Linux platform [34]. Moreover, POX does not operate in a distributed manner, does not support multi-threading, and only supports OpenFlow version 1.0 [35].
- **Open Daylight controller:** The Open Daylight (ODL) controller is a Java-based controller; it is a modular, open-platform SDN controller that allows networks of any size and scope to be controlled, customized, and automated [36]. It has extensive documentation and supports distributed controllers and multithreading [37,38]. ODL has eliminated supplier dependence and supports additional protocols beyond OpenFlow. ODL's major goal is to provide centralized management, so that an intelligent network can be programmed by utilizing API frameworks. ODL leverages various software tools, including Java interfaces, Maven, OSGi, and Karaf, in its design and configuration. OpenDaylight's modularity and flexibility enable end users to choose the features that are most important to them and to build controllers that are tailored to their specific requirements.

The main advantages and disadvantages of the aforementioned SDN controllers chosen for performance investigation are summarized in Table 1.

Table 1. The main advantages and disadvantages of most widely used SDN controllers.

SDN Controller	Advantages	Disadvantages
ONOS	Scalability and performance, open source and community support, modular architecture	Steep learning curve, limited vendor support, outstanding feature set
Ryu	Open source and extensible, ease of use and learning, well-defined APIs and provide customizability	Limited scalability for large networks, few advanced features, device compatibility
POX	Lightweight and simplicity, active community and support, provides full control over the network	Limited scalability and performance, lack of advanced features, easy learning curve
ODL	Rich feature set, strong community and vendor support, modular architecture	Complexity, steep learning curve, resource requirements, the release cycle can sometimes lead to version compatibility issues with third-party applications and plugins

2.4.3. Performance Metrics

The performance factors used in the current study are throughput, Round-Trip Time (RTT) packet delay, packet loss rate, and jitter. Based on simulated network traffic for a given period of time, we use the average values of the commonly utilized assessment metrics of instantaneous throughput, latency, and packet loss rate over different time periods to represent the network performance. The performance factors used here are defined as follows [39].

Throughput: Consider the data rate from a node (i) to a node (j). The instantaneous throughput in a time period Δt is denoted by $Th_{i,j}$. This magnitude expresses the amount of data transferred in an arbitrary time period. An SDWN has, in general, a few $i - j$ links. To represent the average throughput performance of the SDWN, we obtained instantaneous throughput measures based on the simulated 24-h network traffic. Thus, the average throughput of the network traffic of an SDWN can be expressed as follows:

$$Th_{avg} = \frac{\sum_i \sum_j Th_{ij}}{\Delta t} \tag{1}$$

RTT packet latency: RTT latency refers to the time that it takes for a packet to travel from its origin point to its destination and for a response to be received (response time). Let us consider a link $i - j$ and let us suppose that we measure an RTT packet latency $d_{i,j}$. This is instantaneous latency. Different links present their own RTT packet latencies. Instantaneous RTT latency measurements based on simulated 24-h network traffic were taken to calculate the SDWN's average RTT latency. Hence, the average RTT packet latency is expressed as follows:

$$D_{avg} = \frac{\sum_i \sum_j d_{ij}}{\Delta t} \tag{2}$$

The ping utility, which provides message packets that report failures and other information using the Internet Control Message Protocol (ICMP) protocol, was used to measure the RTT delay. The experiment was carried out specifically by running the ping command on the client side, sending an ICMP echo request to the server node, and then waiting for the echo reply packet to be received.

Packet loss: The packet loss is the term used to describe the number of data packets that are transported across a computer network (SDWN) but do not arrive at their destination. Let us consider a random link $i - j$ and let us suppose that we measure a packet loss equal to $ploss_{i,j}$ in a time period Δt. Each time this is done, different links will present different packet loss values. To calculate the SDWN's typical packet loss, measurements of the instantaneous packet loss based on simulated 24-h network traffic were made. The average packet loss is, therefore, stated as follows:

$$PacketLoss_{avg} = \frac{\sum_i \sum_j ploss_{ij}}{\Delta t} \tag{3}$$

Jitter: Jitter is defined as variation in the packet latency. Jitter does not always depend on the RTT latency itself. In other words, jitter refers to the inconsistency or variability in the arrival times of packets within a network. Jitter may be due to congestion, queuing delays, or varying network conditions. High jitter can lead to packet loss, increased latency, and degraded network performance. To perform jitter measurements, we should be sure that the network is capable of measuring and logging packet-level timing information. Additionally, it is necessary to create network traffic with various characteristics. Next, we need to collect packet-level information, such as inter-arrival times and packet timestamps. The inter-arrival times of packets are required for jitter calculations. The standard deviation of the inter-arrival periods can be used to calculate jitter. Measurements are obtained by repeatedly establishing User Datagram Protocol (UDP) connections in typical Mininet topologies and using the iperf utility or ping command. In the experiments that were conducted, the ONOS, Ryu, POX, and ODL controllers' packet delay variations were assessed.

2.5. General Performance Indicator for Multi-Criteria Evaluation of SDWN Networks

In general, performance evaluation factors can be classified into two categories: those that make the network behave better when they are maximized (e.g., throughput) and those that make the network behave better when they are minimized (e.g., latency, cost, etc.).

Let us portray the factors of the first group (maximized) as follows: $m^{max} = m_1^{max}, m_2^{max},$

$\ldots, m_p^{max}$, where p depicts the number of maximized factors. In the same way, let us denote the factors of the second group (minimized) as follows: $m^{min} = m_1^{min}, m_2^{min}, \ldots, m_r^{min}$, where r is the number of minimized factors. The main idea is to find a general indicator that incorporates all the tendencies of the individual coefficients and can suggest to us in a simple way how much better one network is compared to another.

Let us consider two networks a and b having r number of minimized factors; we can say that if $\sum_{i=1}^{r}(m_{a,i}^{min})^2 \leq \sum_{i=1}^{r}(m_{b,i}^{min})^2$, the a-network is better than the b-network. The above sums represent the Euclidean distance of the minimized performance factors. Each dimension portrays the vector space of an individual performance factor. To make the appearance of this formula more compact, it can be called the *general performance indicator* (*GPI*): $GPI = \sqrt{\sum_{i=1}^{r}(m_i^{min})^2}$. In this case, for two networks, a-network and b-network, if $GPI_a \leq GPI_b$, then a-network is more powerful than b-network.

To embed the factors of the minimized group (maximized), instead of a minimized factor m_i^{min}, the formula $(\frac{m_i^{min}}{max(m_i^{min})})$ is considered. When the factor m_i^{min} is maximized, the corresponding fraction is minimized. The term $max(m_i^{min})$ is the normalization value, and the resulting values ranges between 0 and 1. Applying this formula to all minimized factors a common basis is established. Thus, different performance minimized factors that have different measurement scales can expressed in a common area. Hence, the *GPI* for the minimized factors can be defined as follows: $GPI = \sqrt{\sum_{i=1}^{r}(\frac{m_i^{min}}{max(m_i^{min})})^2}$.

In the same way, the *GPI* for the maximized factors is defined as follows: the arbitrary maximized factor m_j^{max} is normalized using the formula $\left(1 - \frac{(m_j^{max})}{max(m_j^{max})}\right)$. The values of this fraction ranges between 0 and 1. While the factor m_j^{max} is a maximized factor the corresponding fraction is a minimized element. Thus, the the *GPI* for the p maximized factors can be expressed as follows: $GPI = \sqrt{\sum_{j=1}^{p}\left(1 - \frac{m_j^{max}}{max(m_j^{max})}\right)^2}$.

To combine both groups of factors, the *GPI* is defined by the following equation:

$$GPI = \sqrt{\sum_{i=1}^{r}\left(\frac{m_i^{min}}{max\left(m_i^{min}\right)}\right)^2 + \sum_{j=1}^{p}\left(1 - \frac{m_j^{max}}{max\left(m_j^{max}\right)}\right)^2} \tag{4}$$

In the immediately preceding formulas, all coefficients are considered equal. Due to the nature of certain problems, however, often, the applications' requirements dictates that we consider certain factors to be more important than others. Very often in multi-criteria problems, the determination of the weight of each parameter is both a decision problem and a design issue. Thus, in such cases, each factor should be multiplied by a special weight that characterizes that factor. Therefore, the final formula can be written as follows:

$$GPI(w_i, w_j) = \sqrt{\frac{\sum_{i=1}^{r} w_i \left(\frac{m_i^{min}}{max(m_i^{min})}\right)^2 + \sum_{j=1}^{p} w_j \left(1 - \frac{m_j^{max}}{max(m_j^{max})}\right)^2}{\sum_{i=1}^{r} w_i + \sum_{j=1}^{p} w_j}} \tag{5}$$

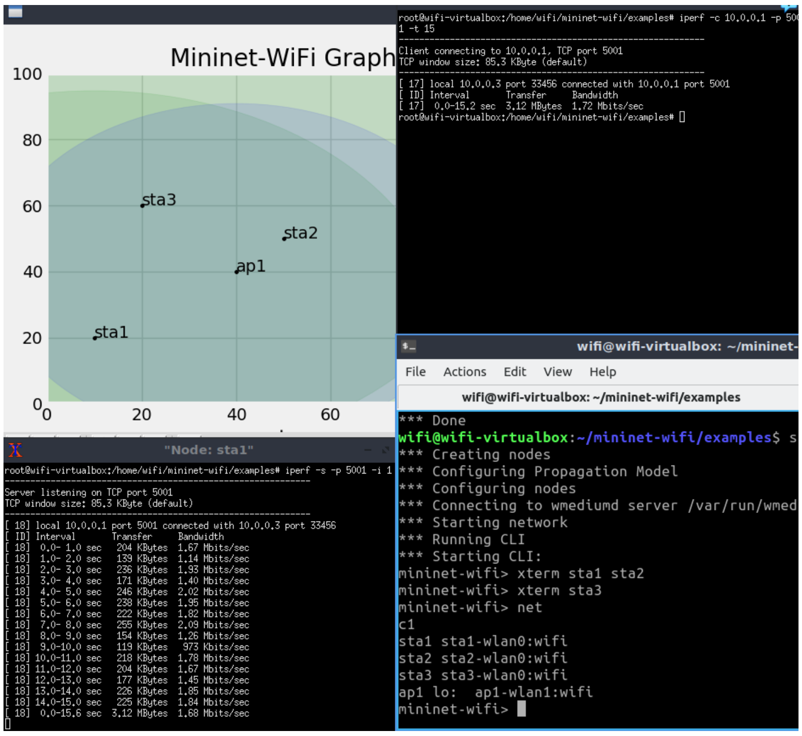
where w_i and w_j represent the weights of the normalized network parameters. The *GPI* ranges from 0 to 1. Hence, as the *GPI*'s value falls, the overall network performance improves.

2.6. Experimental Environment

The simulation was deployed on a virtual machine (VM) using Oracle VM VirtualBox, hosted on a desktop PC running Windows 11. The VM has 4 GB of RAM, and runs Ubuntu 20.04 as the guest Operating System. All SDN controllers are installed on the Linux VM and run on different IP addresses. Mininet-WiFi is being used to develop the data plane infrastructure (e.g., access points and wireless stations). To collect the performance metrics, we took the following steps:

- Run the SDN controller;
- Deploy the network topology, using Mininet-WiFi;
- Use several network tools, such as iperf and ping, to collect performance metrics.

We chose the linear topology as the basic simulation model. In this topology, each access point is connected to the controller and each wireless station is connected to one access point. The number of access points and wireless stations ranges from 25 to 100. When the topology is created, we picked up pairs of nodes randomly and ran performance tests for a certain length of time. Figure 2 describes a simple example of a simulation environment. For measuring throughput, an active measurement tool, iperf, is used. Each iperf test generates TCP traffic, and the transmissions occur for 60 s. For counting the latency, jitter, and packet loss, the network tool ping is used. In this case, the first node sends a sequence of 10 packets to the last node. The ping test is repeated 10 times, and the average of each metric is calculated.



```
root@wifi-virtualbox:/home/wifi/mininet-wifi/examples# iperf -c 10.0.0.1 -p 50
1 -t 15
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 17] local 10.0.0.3 port 33456 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 17]  0.0-15.2 sec  3.12 MBytes  1.72 Mbits/sec
root@wifi-virtualbox:/home/wifi/mininet-wifi/examples# 
```

```
root@wifi-virtualbox:/home/wifi/mininet-wifi/examples# iperf -s -p 5001 -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 18] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 33456
[ ID] Interval       Transfer     Bandwidth
[ 18]  0.0- 1.0 sec   204 KBytes  1.67 Mbits/sec
[ 18]  1.0- 2.0 sec   139 KBytes  1.14 Mbits/sec
[ 18]  2.0- 3.0 sec   236 KBytes  1.93 Mbits/sec
[ 18]  3.0- 4.0 sec   171 KBytes  1.40 Mbits/sec
[ 18]  4.0- 5.0 sec   246 KBytes  2.02 Mbits/sec
[ 18]  5.0- 6.0 sec   238 KBytes  1.95 Mbits/sec
[ 18]  6.0- 7.0 sec   222 KBytes  1.82 Mbits/sec
[ 18]  7.0- 8.0 sec   255 KBytes  2.09 Mbits/sec
[ 18]  8.0- 9.0 sec   154 KBytes  1.26 Mbits/sec
[ 18]  9.0-10.0 sec   119 KBytes   973 Kbits/sec
[ 18] 10.0-11.0 sec   218 KBytes  1.78 Mbits/sec
[ 18] 11.0-12.0 sec   204 KBytes  1.67 Mbits/sec
[ 18] 12.0-13.0 sec   177 KBytes  1.45 Mbits/sec
[ 18] 13.0-14.0 sec   226 KBytes  1.85 Mbits/sec
[ 18] 14.0-15.0 sec   225 KBytes  1.84 Mbits/sec
[ 18]  0.0-15.6 sec  3.12 MBytes  1.68 Mbits/sec
```

```
*** Done
wifi@wifi-virtualbox:~/mininet-wifi/examples$ s
*** Creating nodes
*** Configuring Propagation Model
*** Configuring nodes
*** Connecting to wmediumd server /var/run/wmed
*** Starting network
*** Running CLI
*** Starting CLI:
mininet-wifi> xterm sta1 sta2
mininet-wifi> xterm sta3
mininet-wifi> net
c1
sta1 sta1-wlan0:wifi
sta2 sta2-wlan0:wifi
sta3 sta3-wlan0:wifi
ap1 lo:  ap1-wlan1:wifi
mininet-wifi> 
```

Figure 2. A simple instance of the experimental environment using Mininet-WiFi and the POX controller.

3. Simulation and Results

In this section, the values of the SDWN performance factors—the throughput, RTT latency, packet loss, and jitter—are presented in relation to the device population for various controllers. These results were obtained from the experimental trials that we carried out, and they are presented graphically below.

3.1. Throughput

Figure 3 shows the SDWN's average throughput (in Mbits/s) in relation to the device population for the four controllers. When the device population is below 50, the throughput is generally high, but in all cases, when the device population becomes greater than 50, the throughput decreases gradually for all the controllers.

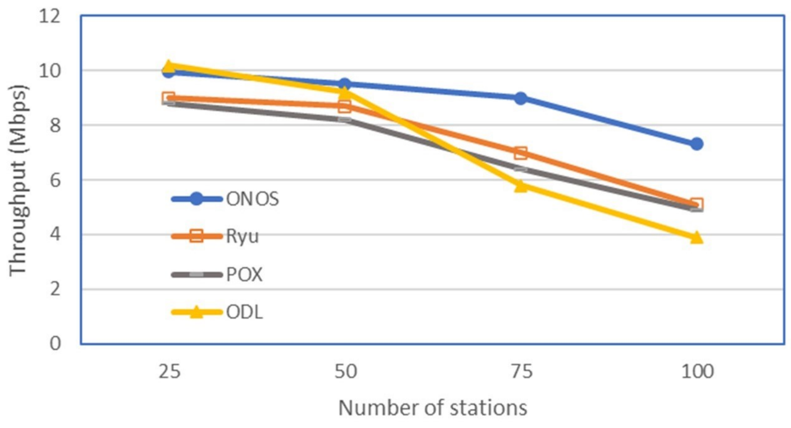


Figure 3. The average throughput versus the number of wireless devices in the SDWN for four controllers.

Figure 3 depicts that for device populations greater than 50, the performance of the ONOS controller is superior to the performance of all the other controllers in terms of throughput (blue line), while the worst performance is presented by the ODL controller (yellow line). For a device population equal to 100, a considerable difference is observed between ONOS and ODL, as the throughput of the ONOS is 45.5% better than that of the ODL.

3.2. RTT Latency

Figure 4 shows the average RTT latency of the four controllers in relation to the device population of the SDWN. The Ryu and ODL controllers provide the best (lowest) RTT latency values (for 100 devices, the values are 10.8 µs and 22 µs, respectively). On the other side, the ONOS and POX controllers (especially the ONOS controller—blue line) have unusually high (worst case) packet delay values.

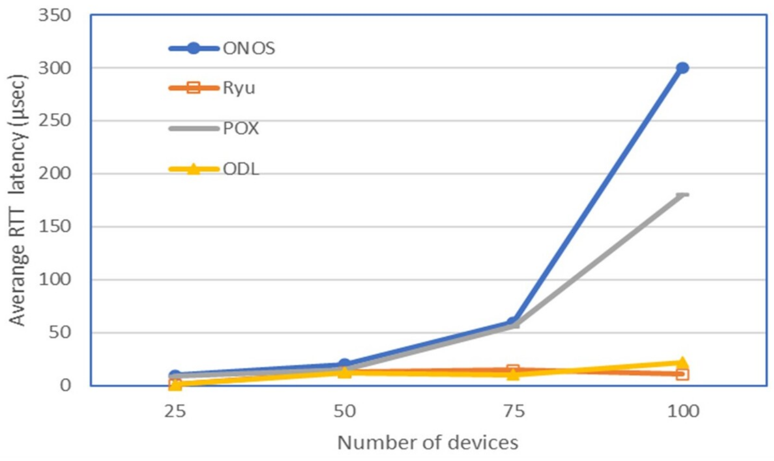


Figure 4. Average RTT latency versus the number of wireless devices in the SDWN for four controllers.

3.3. Packet Loss

Figure 5 illustrates the average packet loss versus the number of wireless devices in the SDWN for the four controllers. In all cases, the ODL controller (yellow line) presents the worst behavior (the highest packet loss values).

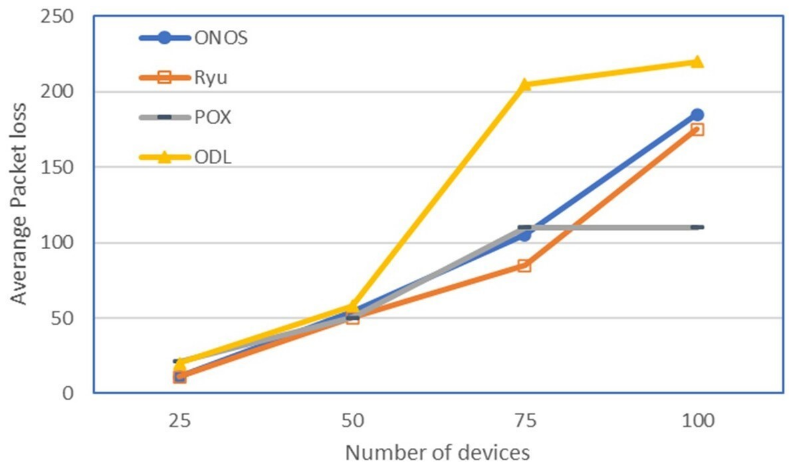


Figure 5. Average packet loss versus the number of wireless devices in the SDWN for four controllers.

It is noteworthy that the POX controller exhibits a rather constant packet loss rate for device counts greater than 75, which, in addition to being a relatively stable value, is also particularly low compared to the corresponding values of the other controllers. In the intermediate band (device population between 50 and 75), the Ryu controller (orange line) provides the best behavior (lowest packet loss).

3.4. Average Jitter

Figure 6 illustrates the average jitter (in ms) versus the device population of the SDWN for the four controllers.

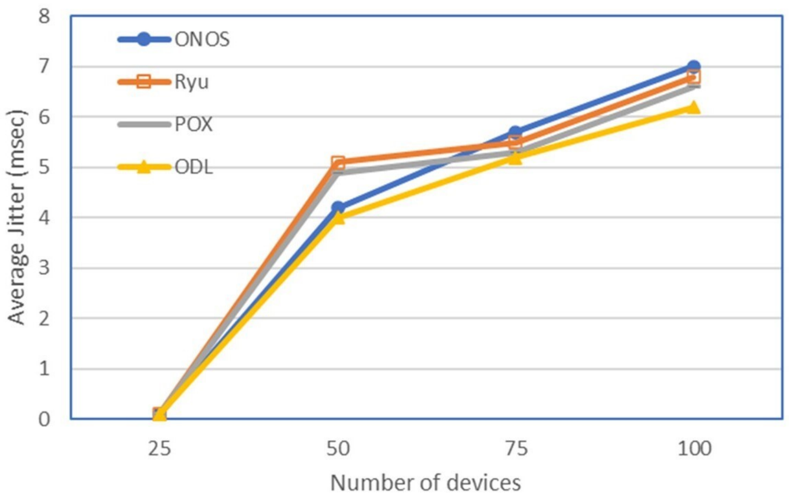


Figure 6. Average jitter versus the number of wireless devices in the SDWN for four controllers.

From the plot, it can be seen that the ODL controller presents the lowest jitter in all cases (yellow line). Additionally, for device populations greater than 75, the ONOS controller should be avoided if the jitter factor is important for a given application.

Nevertheless, to evaluate various SDWNs that are described by different performance factors, where each factor presents its own behavior, we need a special methodology to decide which of the networks has the best behavior. Next, a multi-criteria methodology is presented that will help us decide which network is the best (in terms of performance) in relation to the wireless device population.

3.5. General Performance Factor Results

From the initial experiments, it became apparent that the values of the SDWN controllers' throughput and the values of the RTT packet latency, jitter, etc., are inversely proportional to each other for various device populations. Therefore, a broad assessment using only one indicator—as explained in the previous section—is extremely practical. Since each performance factor has its own units of measurement and range, the performance factors must be normalized to obtain a common field of reference values. According to the previous analysis, the smaller the *GPI* index, the better the network performance.

Figure 7 shows the *GPI* values for an SDWN versus the wireless device population that the network has for the four controllers under study. The overall index (*GPI*) values shown in Figure 7 were calculated by considering the four performance factors to have equal weights.

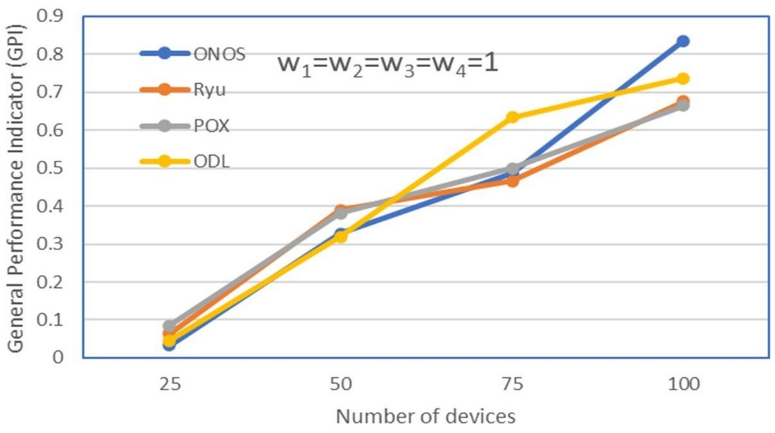


Figure 7. General performance indicator values of an SDWN versus the number of wireless devices for four controllers (ONOS, Ryu, POX, and ODL), considering the four performance factors to have equal weights.

In this scenario, it is observed that for device populations greater than 72, the Ryu controller, as well as the POX controller, present a potent behavior. On the other hand, for lower device populations, the ONOS and ODL controllers are more suitable in terms of performance.

Additionally, Figure 8 illustrates the *GPI* values for an SDWN versus the wireless device population for the four controllers under study, but in this case, the throughput factor is considered the most important factor ($w_1 = 3$ and $w_2 = w_3 = w_4 = 1$). The plot reveals that the Ryu and POX controllers still have the best overall performance for device populations greater than 80. For device populations between 50 and 78, the ONOS controller provides the best performance (blue line), and for small networks (<50 devices), the ONOS and ODL controllers still provide the best performance.

On the other hand, there are applications that have little tolerance in terms of the packet transmission time (e.g., real-time applications, video streaming, voice over Internet Protocol (IP), etc.). In these instances, it is critical that the network maintains a low packet transmission latency and, in many cases, low jitter values. In such circumstances, the RTT delay of the packets, as well as the jitter, are considered crucial considerations.

Figure 9 reveals that for device populations greater than 60, the Ryu controller presents the best performance behavior (orange line). For device populations less than 60, the ONOS and ODL controllers provide the best performance results.

Moreover, Figure 10 displays the *GPI* values obtained for the four controllers when the RTT latency and the jitter have weights of 2. The plot reveals that the Ryu and POX controllers remain the best choice for device populations greater than 70 (orange and gray lines).

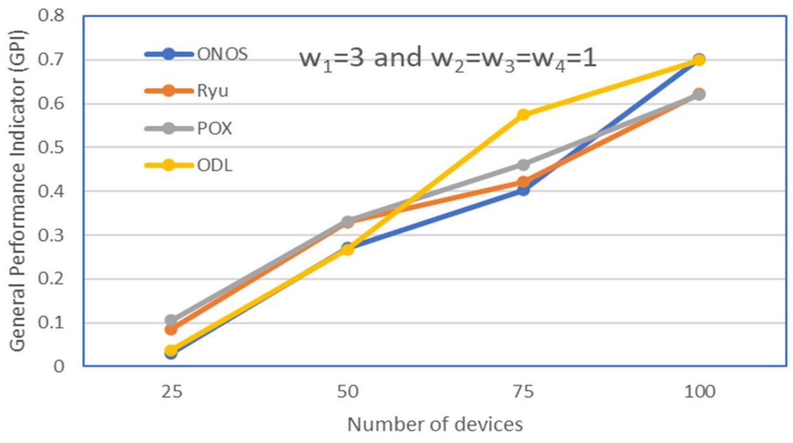


Figure 8. General performance indicator of an SDWN network versus number of wireless devices for four Controllers (ONOS, Ryu, POX, and ODL) considering the throughput as a significant performance factor ($w_1 = 3$).

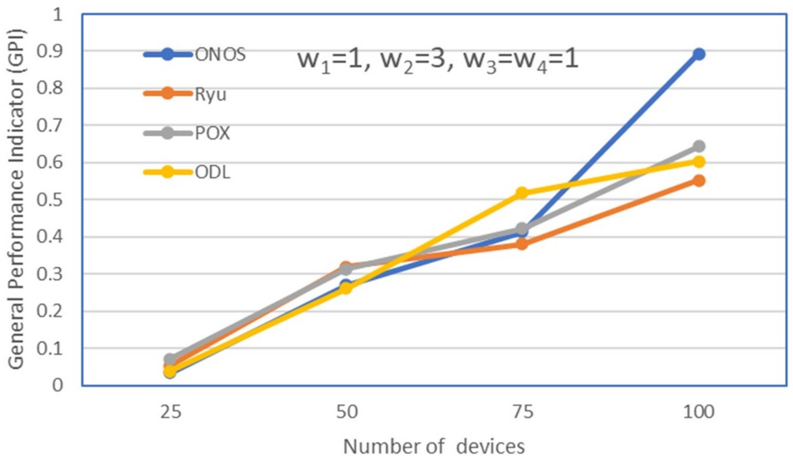


Figure 9. General performance indicator values of an SDWN versus the number of wireless devices for four controllers, considering the RTT latency to be the most significant performance factor ($w_2 = 3$).

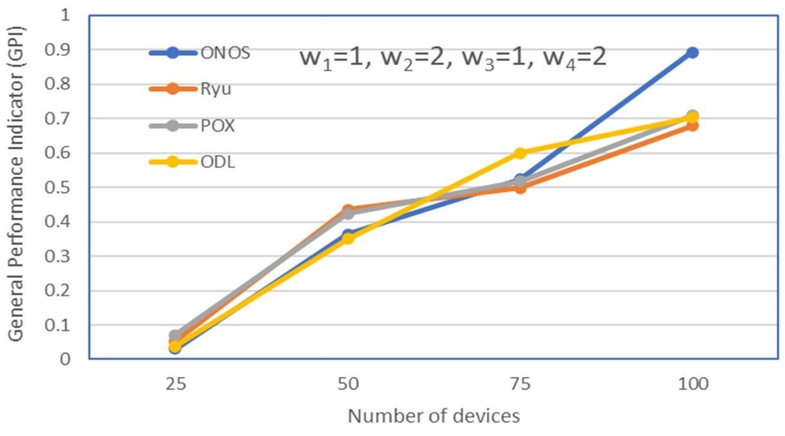


Figure 10. General performance indicator values of an SDWN versus the number of wireless devices for four controllers, considering the RTT latency and jitter to be the most important performance factors ($w_2 = 2$ and $w_4 = 2$).

Finally, there are applications that, due to their nature, cannot tolerate packet loss (e.g., accounting or stock market applications). It goes without saying that this requirement should be met by the network. In these scenarios, the most important factor is packet loss.

Figure 11 depicts the *GPI* values obtained for various device populations when the packet loss has a weight of 3.

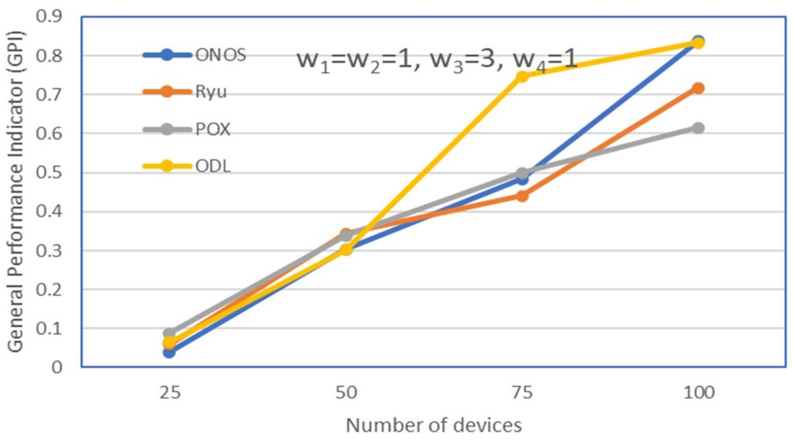


Figure 11. General performance indicator values of an SDWN versus the number of wireless devices for four controllers considering the packet loss to be the most important performance factor ($w_3 = 3$).

This illustration shows that, again, the Ryu and POX controllers present efficient performances compared to the other two controllers (ONOS and ODL). This image demonstrates that the Ryu and POX controllers outperform the other two controllers (ONOS and ODL) for device counts of more than approximately 70. Furthermore, it is evident that the usage of the ODL controller should be avoided for applications that are sensitive to packet loss because it is prone to significant packet loss rates.

4. Conclusions

In this work, SDWNs were studied using the most common controllers (ONOS, Ryu, POX, and ODL). Furthermore, by employing the Mininet-WiFi emulation platform, we assessed four significant performance variables for each considered controller (throughput, RTT latency, packet loss, and jitter). An evaluation methodology was applied to SDWNs. The general performance indicator that was defined for the multi-criteria evaluation of SDWNs can play a significant role in decision making for the configuration of SDWNs.

From this study, it is obvious that the Ryu and POX controllers are more powerful for large SDWNs (number of devices greater than approximately 70) compared to the other two controllers (ONOS and ODL).

Our work focuses on specific wireless network topologies (linear) and investigates the usability of the general performance indicator to assess the SDN controllers. However, our study has potential limitations. There are, therefore, some other factors that influenced the network performance that were not addressed in this work, such as the dynamic nature of wireless networks, the nodes mobility and devices' heterogeneity, the multiple transmissions, and consequent interference issues. SDN technology has potential solutions and benefits for all these cases and could be the subject of future work.

Moreover, future research will concentrate on examining the controller placement problem (CPP) to determine the ideal number and locations of SDN controllers in a large-scale network.

The approach and findings of this performance study can be valuable tools for analyzing communication, particularly in the context of SDNs, as well as for designers or teams reviewing, monitoring, or optimizing SDNs.

Author Contributions: Conceptualization, methodology, and supervision E.S. and C.S.; investigation, I.K. and I.B.; writing original draft preparation, S.V.M. and E.S.; review and editing, E.S. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data not available for this study.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software-Defined Networking
SDWN	Software-Defined Wireless Network
MIMO	Multiple Input, Multiple Output
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ONOS	Open Network Operating System
ODL	Open Daylight
GPI	General Performance Indicator
RTT	Round-Trip Time

References

1. Feamster, N.; Rexford, J.; Zegura, E. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–98. [CrossRef]
2. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
3. Tong, H.; Li, X.; Shi, Z.; Tian, Y. A Novel and Efficient Link Discovery Mechanism in SDN. In Proceedings of the 2020 IEEE 3rd International Conference on Electronics and Communication Engineering (ICECE), Xi'an, China, 14–16 December 2020 ; pp. 97–101. [CrossRef]
4. Ranji, R.; Javed, U.; Boltjes, B.; Bouhafs, F.; Den Hartog, F. Optimizing wireless network throughput under the condition of Physical Layer Security using Software-Defined Networking enabled collaboration. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 1–6. [CrossRef]
5. Stergiou, E.; Angelis, C.; Margariti, S.V. Evaluation Methodology of MIMO Networks Performance over Rayleigh Fading. *Int. J. Comput. Netw. Commun.* **2020**, *12*, 37–52. [CrossRef]
6. Costanzo, S.; Galluccio, L.; Morabito, G.; Palazzo, S. Software Defined Wireless Network (SDWN): An evolvable architecture for W-PANs. In Proceedings of the 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Torino, Italy, 16–18 September 2015; pp. 23–28. [CrossRef]
7. Chadi Aljundi, A.; Dekeyser, J.L.; Kechadi, M.T.; Scherson, I.D. A universal performance factor for multi-criteria evaluation of multistage interconnection networks. *Future Gener. Comput. Syst.* **2006**, *22*, 794–804. [CrossRef]
8. Stergiou, E.; Angelis, C.T.; Giannakeas, N.; Tsoumanis, G.; Tzallas, A.T.; Glavas, E. Optimum Capacity over Power Consumption Requirements in MIMO Systems. In Proceedings of the 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 1–3 July 2019; pp. 415–419. [CrossRef]
9. Zhu, L.; Karim, M.M.; Sharif, K.; Xu, C.; Li, F.; Du, X.; Guizani, M. SDN controllers: A comprehensive analysis and performance evaluation study. *ACM Comput. Surv.* **2020**, *53*, 1–40. [CrossRef]
10. Assegie, S.; Nair, P. Performance analysis of emulated software defined wireless network. *Indones. J. Electr. Eng. Comput. Sci.* **2019**, *16*, 311–318. [CrossRef]
11. Chaurasia, A.; Mishra, S.N.; Chinara, S. Performance Evaluation of Software-Defined Wireless Networks in IT-SDN and Mininet-WiFi. In Proceedings of the 2019 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 24–27 July 2019; pp. 315–319. [CrossRef]
12. Mamushiane, L.; Lysko, A.; Dlamini, S. A comparative evaluation of the performance of popular SDN controllers. In Proceedings of the 2018 Wireless Days (WD), Dubai, United Arab Emirates, 3–5 April 2018; IEEE: New York, NY, USA , 2018; pp. 54–59.
13. Koulouras, I.; Margariti, S.V.; Bobotsaris, I.; Stergiou, E.; Stylios, C. On the Performance of SDN Controllers in Real World Topologies. In Proceedings of the 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Chandler, AZ, USA, 14–16 November 2022; IEEE: New York, NY, USA, 2022; pp. 143–148.
14. Islam, S.; Khan, M.A.I.; Shorno, S.T.; Sarker, S.; Siddik, M.A. Performance evaluation of SDN controllers in wireless network. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; IEEE: New York, NY, USA, 2019; pp. 1–5.
15. Khondoker, R.; Zaalouk, A.; Marx, R.; Bayarou, K. Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In Proceedings of the 2014 World Congress on Computer Applications and Information Systems (WCCAIS), Hammamet, Tunisia, 17–19 January 2014; IEEE: New York, NY, USA, 2014; pp. 1–7.

16. Jagadeesan, N.A.; Krishnamachari, B. Software-defined networking paradigms in wireless networks: A survey. *ACM Comput. Surv.* **2014**, *47*, 1–11. [CrossRef]
17. Babu, S.; Mithun, P.V.; Manoj, B.S. A Novel Framework for Resource Discovery and Self-Configuration in Software Defined Wireless Mesh Networks. *IEEE Trans. Netw. Serv. Manag.* **2019**, *17*, 132–146. [CrossRef]
18. Rashid, S.; Alkababji, A.; Khidhir, A. Performance evaluation of software-defined networking controllers in wired and wireless networks. *TELKOMNIKA Telecommun. Comput. Electron. Control* **2023**, *21*, 49–59. [CrossRef]
19. Team, M. Mininet: An Instant Virtual Network on Your Laptop (or Other PC). Available online: https://mininet.org/ (accessed on 29 June 2023).
20. Kaur, K.; Kaur, S.; Kumar, K.; Aggarwal, N.; Mangat, V. Mininet-WiFi as Software-Defined Wireless Network Testing Platform. In *Security and Resilience of Cyber Physical Systems*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2022; pp. 91–102.
21. Get Started. Available online: https://mininet-wifi.github.io/get-started/ (accessed on 29 June 2023).
22. Fontes, R.D.R.; Rothenberg, C.E. Mininet-wifi: A platform for hybrid physical-virtual software-defined wireless networking research. In Proceedings of the 2016 ACM SIGCOMM Conference, Florianópolis, Brazil, 22–26 August 2016; pp. 607–608.
23. Al Somaidai, M. Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation. *Am. J. Softw. Eng. Appl.* **2014**, *3*, 74–82. [CrossRef]
24. 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. [CrossRef]
25. Wazirali, R.; Ahmad, R.; Alhiyari, S. SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Appl. Sci.* **2021**, *11*, 6999. [CrossRef]
26. Liatifis, A.; Sarigiannidis, P.; Argyriou, V.; Lagkas, T. Advancing SDN: From OpenFlow to P4, a Survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
27. Soares, A.A.Z.; Vieira, J.L.; Quincozes, S.E.; Ferreira, V.C.; Uchôa, L.M.; Lopes, Y.; Passos, D.; Fernandes, N.C.; Moraes, I.M.; Muchalua, D. SDN-based teleprotection and control power systems: A study of available controllers and their suitability. *Int. J. Netw. Manag.* **2021**, *31*, e2112. [CrossRef]
28. Badaró V. Neto, F.J.; Miguel, C.J.; de Jesus, A.C.d.S.; Sampaio, P.N. SDN Controllers—A Comparative approach to Market Trends. In Proceedings of the 9th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2021), Zaragoza, Spain, 2–4 February 2021; pp. 48–51. [CrossRef]
29. Open Networking Foundation. Available online: https://opennetworking.org/ (accessed on 29 June 2023).
30. ONOS—ONOS. Available online: https://wiki.onosproject.org/display/ONOS/ONOS (accessed on 29 June 2023).
31. GitHub—Faucetsdn/Ryu: Ryu Component-Based Software Defined Networking Framework. Available online: https://github.com/faucetsdn/ryu (accessed on 29 June 2023).
32. Ahmad, S.; Mir, A.H. Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers. *J. Netw. Syst. Manag.* **2021**, *29*, 9. [CrossRef]
33. Ha, N.V.; Quan, D.D.; Nguyen, T.T.T. Graphical User Interface for RYU Software Defined Network Controller. In Proceedings of the 2022 IEEE 8th Information Technology International Seminar (ITIS), Surabaya, Indonesia, 19–21 October 2022; pp. 312–317. [CrossRef]
34. Installing POX—POX Manual Current Documentation. Available online: https://noxrepo.github.io/pox-doc/html/ (accessed on 29 June 2023).
35. Stancu, A.L.; Halunga, S.; Vulpe, A.; Suciu, G.; Fratu, O.; Popovici, E.C. A comparison between several Software Defined Networking controllers. In Proceedings of the 2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), Nis, Serbia, 14–17 October 2015; pp. 223–226. [CrossRef]
36. Home—OpenDaylight. Available online: https://www.opendaylight.org/ (accessed on 29 June 2023).
37. Eftimie, A.; Borcoci, E. SDN controller implementation using OpenDaylight: Experiments. In Proceedings of the 2020 13th International Conference on Communications (COMM), Bucharest, Romania, 18–20 June 2020; pp. 477–481. [CrossRef]
38. Elmoslemany, M.; Tag Eldien, A.; Selim, M. Performance Analysis in Software Defined Network (SDN) Multi-Controllers. *Delta Univ. Sci. J.* **2023**, *6*, 181–192. [CrossRef]
39. Hu, H.; Ye, M.; Zhao, C.; Jiang, Q.; Wang, Y.; Qiu, H.; Deng, X. Intelligent multicast routing method based on multi-agent deep reinforcement learning in SDWN. *arXiv* **2023**, arXiv:2305.10440.