A comparative evaluation of the performance of open-source SDN controllers

Evaluación comparativa del rendimiento de controladores SDN de código abierto

ARTICLE HISTORY

Received 01 October 2020
Accepted 16 October 2020

Daniel Haro Mendoza
Facultad de Ingeniería
Universidad Nacional de Chimborazo
Riobamba, Ecuador
dharo@unach.edu.ec

Luis Tello Oquendo
Facultad de Ingeniería
Universidad Nacional de Chimborazo
Riobamba, Ecuador
ltello@unach.edu.ec

Luis A. Marrone
LINTI
Universidad Nacional de La Plata
La Plata, Argentina
lmarrone@linti.unlp.edu.ar

A comparative evaluation of the performance of open-source SDN controllers

Evaluación comparativa del rendimiento de controladores SDN de código abierto

Daniel Haro Mendoza
Facultad de Ingeniería
Universidad Nacional de Chimborazo
Riobamba, Ecuador
dharo@unach.edu.ec

Luis Tello Oquendo
Facultad de Ingeniería
Universidad Nacional de Chimborazo
Riobamba, Ecuador
luis.tello@unach.edu.ec

Luis A. Marrone
LINTI
Universidad Nacional de La Plata
La Plata, Argentina
lmarrone@linti.unlp.edu.ar

Abstract — Software-defined Networking (SDN) constitutes a new era in the development of internetworking. The SDN paradigm splits the data plane from the control plane. It uses controller equipment, which is responsible for centrally managing several network devices simultaneously. This study analyzes three open-source controllers for SDN based on the OpenFlow protocol. Specifically, the performance of FloodLight, OpenDayLight (ODL), and Ryu controllers is evaluated in terms of latency, throughput, and scalability. In doing so, the Cbench tool is used in an emulated environment with Mininet. The results show that the Ryu controller presents the lowest performance in all the evaluated parameters; ODL provides lower latency and FloodLight higher throughput. Regarding scalability, we conclude that Floodlight can be used in small networks, whereas ODL can be used in dense networks. Furthermore, we evaluate their main characteristics, which must be considered for their choice prior to implementation and deployment.

Index Terms — SDN, controller, networking, performance.

Resumen — *Las redes definidas por software (SDN) constituyen una nueva era en el diseño de la interconexión de redes. El paradigma SDN separa el plano de datos del plano de control. Para esto utiliza un equipo controlador, que se encarga de gestionar de forma centralizada varios dispositivos de red al mismo tiempo. Este estudio analiza tres controladores SDN de código abierto basados en el protocol OpenFlow. Específicamente, el rendimiento de los controladores FloodLight, OpenDayLight (ODL) y Ryu son evaluados en términos de latencia, throughput y escalabilidad. Para ello se utilizó la herramienta Cbench en un entorno emulado con Mininet. Los resultados muestran que el controlador presenta un menor rendimiento en todos los parámetros evaluados; ODL tiene una menor latencia y Floodlight un mayor throughput. En lo que tiene que ver a escalabilidad, se concluye que Floodlight es recomendable para redes pequeñas y ODL para redes densas. Además, evaluamos sus principales características, las cuales deben ser tomadas en cuenta para su elección antes de su implementación y despliegue.*

Palabras clave — *SDN, controlador, redes, rendimiento.*

I. INTRODUCTION

For a long time, network technologies have evolved slower than other communication technologies [1]. This is mainly because manufacturers design their control logic in network equipment closely, making collaborative work and innovation impossible. The popularization of smart devices, virtualization, cloud computing, and data analytics are some of the innovations that have introduced multiple services and applications. They have diverse performance requirements, such as quality of service (QoS), security, mobility, and scalability. These required performance levels have forced data networks to evolve their traditional technologies. This is how the concept of softwarizing data networks appears to adapt the operation of equipment to the logic of the applications and business objectives. In this context, gaining great popularity, the SDN paradigm appears as an approach to promote innovation in the network through greater flexibility, programming capacity, management, and profitability [2].

The SDN principle is to centralize network intelligence by physically separating the control plane from the data plane through application programming. This intelligence is logically located in controllers that maintain a vision, global management, and communication with network devices through a protocol such as OpenFlow. The paradigm of traditional networks is changing rapidly with the advent of SDN; companies that process large amounts of data, such as Facebook and Google [3], are migrating their networks towards software-defined approaches.

The popularization of SDNs has led to many controllers' appearance-based mainly on the OpenFlow protocol; therefore, choosing the right controller for implementation becomes a primary task for the network operator. Moreover, the analysis of parameters and selection criteria constitutes a research opportunity. These drivers were developed by universities, device manufacturers, or research groups, written in different programming languages for multiple applications, and used various techniques to improve their performance.

A comparative study of a controller's main operating parameters in a software-defined network with open source tools is presented in this work. For this, the three controllers that are among the most widespread and implemented have been chosen, namely OpenDaylight (ODL), Ryu, and Floodlight. Additionally, tools for their evaluation, challenges, and research opportunities that can be exploited in future work on this topic are reviewed.

The remainder of this article is structured as follows. Section II reviews the related work on the comparison of SDN controllers. Section III presents a theoretical foundation, background information on software-defined networks, and relevant works related to a controller's role within the SDN network. Section IV presents the network scenario and the software used to evaluate the controllers chosen for the study; a summary of the essential characteristics and functions of the tools used is also presented. Section V presents the evaluations carried out in each of the controllers in terms of latency, throughput, and scalability. Finally, conclusions and future challenges are described in Section VI and VII, respectively.

II. RELATED WORK

Concerning the comparison of SDN controllers, studies such as [4] have been carried out where the ability to handle high traffic loads by SDN controllers is evaluated, carrying out Denial of Service (DoS) attacks on the underlying hosts of the net. It is concluded that the ONOS and ODL drivers are the ones that achieve the best performance. This study uses Mininet with a linear topology with three hosts, while JMeter is used as a tool for load and stress tests in the communication between the controller and the SDN applications.

In [5], the performance of SDN controllers such as Beacon, Pox, Floodlight, and Ryu is compared considering latency and throughput depending on the network's size, up to 100 switches. For the evaluation, the Cbench tool is used in an Ubuntu virtual machine. The results show that Floodlight has a lower latency as the number of Switches in the network increases; in terms of performance, the best performers are Floodlight and Beacon, with almost identical behavior.

In [6], the Cbench tool is used to evaluate the ONOS, Ryu, Floodlight, and OpenDayLight drivers' throughput and latency. In this study, a comparison of each controller's essential operating characteristics is made first, recommending OpenDayLight for its support. Regarding the performance evaluation, ONOS is recommended, while Ryu presented the best results in latency. One of the limitations of this study is that the comparison is made for network environments with up to 32 switches; that is, the performance in large networks is not evaluated. Another study [7] compares the Floodlight and Ryu controllers' performance in terms of latency and throughput using Mininet and the qperf performance measurement tool. The comparison of the two controllers is made through different experiments on different SDN network topologies. The results show that Floodlight performs better than Ryu, transmitting higher bandwidth and lower latency in all deployed topologies.

In [8], 12 widely used controllers are qualitatively and quantitatively evaluated. In addition to the performance analysis and support criteria, a performance test in terms of latency, throughput, and threads is carried out with the Cbench tool on Ubuntu 14.04. The results describe better performance in C and Java-based drivers, while Python-based drivers such as POX did not show efficient performance. It is also concluded that the best controller's choice to implement will depend on the requirements and its application.

In general, the related works focus on performance evaluation in terms of latency and throughput. We can also realize that the most efficient, used, and recommended controllers are Floodlight, ODL, and Ryu, limiting our study for these three controllers. One of the study's contributions is the scalability evaluation that has not been deepened in previous studies. With the help of the latency and throughput parameters, we will know what the network's performance is in saturated or large-scale scenarios so that the reader can decide which controller to use for different sizes of the network. For this, in the test scenarios, the number of switches increases, reaching up to 256 with 10,000 hosts. Like other studies, we used the Mininet tool to emulate different SDN and Cbench topologies for performance testing.

III. BACKGROUND INFORMATION

Considering scenarios with multiple operators, applications, networks sharing the same infrastructure, services with very different QoS requirements, and a highly dynamic and changing environment, using traditional interconnectivity and network management techniques are not the best option [9]. This is mainly due to the following reasons:

- It is almost impossible to find an optimal performance by configuring quality of service and prioritization techniques for traffic, users, and applications in environments as complex as those mentioned above.

- The dynamism of these networks makes it difficult for them to operate in a stationary regime, introducing complexity to add or remove devices, difficulty applying policies, dependence on suppliers, and inability to respond to changing business needs.

Then, SDN is presented as a new concept to meet new needs. It aims to transform the architectures and management of networks, as is known today [2].

A. SDN Architecture

SDN has become a popular concept, so there is no universal vision of it, as it still depends on the manufacturer that implements it in its infrastructure [10].

There is a variety of definitions for SDN. However, the one made by the Open Networking Foundation (ONF) in [11] can be cited, which is widely accepted: "Software-defined networks are defined as dynamic, manageable network architecture, adaptable, cost-efficient, which makes it ideal for the high bandwidth demands and dynamic nature of today's applications. This architecture decouples the control of the network and the functionality of forwarding of information, allowing that the control of the network can be completely programmable, achieving that the applications and network services are abstracted from the underlying network infrastructure."

The network devices' control function is eliminated and transferred to the controller by separating the control and data planes, as illustrated in Fig. 1.

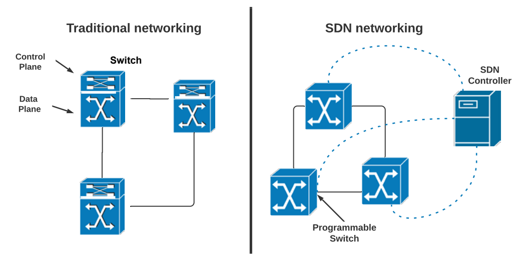

Fig. 1. SDN basic diagram.

The ONF is a non-profit consortium made up of institutions and companies in the IT area dedicated to developing, standardization, and commercializing SDNs. Fig. 2 illustrates the three-layer architecture proposed by the ONF. The devices only care about the forwarding of packets in the infrastructure layer, based on the rules defined by one or more SDN controllers of the control layer, according to the programming logic required and predefined in the application layer [12].

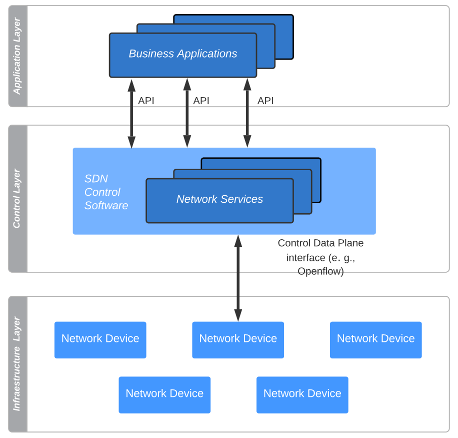

Fig. 2. SDN architecture.

According to ONF [13], the key aspects that characterize the implementation of an SDN architecture are programmability, logically centralized architecture, abstraction, independence, and the use of open standard, as described in Table I.

TABLE I. CHARACTERISTICS OF AN SDN ARCHITECTURE

Characteristic	Description
Programmability	Because forwarding functions were decoupled, it allows that the applications interact with the network.
Logically centralized intelligence	Flexibility to network control due to the global vission of the network. Make dynamic changes according to requirements.
Abstraction	The capability to support multivendor devices management and technologies in the infrastructure layer.
Independence	Applications for managing network resources that do not depend on proprietary software.
Use of open standard	Facilitates network design and operation.

B. SDN controllers

As illustrated in Fig. 2, the SDN network's intelligence is (logically) centralized in software-based controllers, which maintain a global vision of it. Network devices delegate the responsibility for managing communications to the controller and become simple traffic relay units.

The controller is a piece of equipment that can convert the SDN applications' requests into orders towards the lower layers' devices and thus provide the SDN applications with an abstract vision of the network. It is responsible for deciding how the packets should be forwarded by one or more network elements and scheduling the network nodes to implement it. Its function is to keep updated the data plane's forwarding tables or infrastructure devices based on the network topology or requests for external services [11].

The SDN controller is in charge of translating the application layer's needs or requirements to the infrastructure layer's network elements through the southbound protocols. It also provides an API (northbound) so that flows can be programmed from the application layer and provide feedback to the application with information on the network topology or packet traffic [6].

The data plane control interface is formalized through the OpenFlow protocol, which has become the official protocol for the remote connection between the SDN controller and the switches. As a result, creating an SDN network should be based on selecting devices and control software that support the OpenFlow standard.

In [12], the main characteristics to consider when evaluating an SDN controller are presented. Table II summarizes these characteristics for some of the existing open-source drivers. It is worth noting that additional characteristics should be considered, such as:

- Scalability, the number of devices that a controller can support.
- Throughput, the number of flows that the controller can set per second.
- Reliability, ability to discover several paths from origin to destination to ensure availability due to link interruption.
- Network security, controller's ability to support authentication, and authorization.

TABLE II. OPEN-SOURCE SDN CONTROLLER

Characteristic	***Floodlight***	***Ryu***	***ODL***
OpenFlow support	Yes	Yes	Yes
Virtualization	Mininet and Open vSwitch	Mininet and Open vSwitch	Mininet and Open vSwitch
Development language	Java	Python	Java
Graphic interface	Web	Web	Web
Platforms support	Linux, Mac OS, Android	Linux	Linux, Mac OS, Windows
OpenStack support	Yes	Yes	Yes
Multitask	Yes	No	Yes
Time in the market (Years)	8	7	6
Documentation	Good	Average	Average

Due to the characteristics and functionalities, such as being open-source, supporting OpenFlow, Openstack support, popularity, and existing documentation, we evaluate three controllers: ODL, Floodlight, and Ryu.

1) OpenDaylight (ODL) controller: It is an open-source project. It is a driver implemented in the software included within its own Java Virtual Machine (JVM). The project is currently supported by Citrix, IBM, Juniper, Microsoft, VMware, and others. It supports OpenFlow, although it can support other protocols such as BGP. It defines a service level agreement (SLA) that you must adhere to regardless of the controller and network devices' underlying protocol. Its most recent version, Beryllium, includes high availability, clustering, security features, and the improvement or addition of new protocols. The SDN OpenDaylight driver has multiple layers. The upper layer comprises network and business logic applications, the middle layer where the SDN abstractions are located, and the lower layer where the physical and virtual devices are located. The middle layer is the framework that houses northbound and southbound APIs. The business logic resides in the applications that are above the middle tier. These applications use the controller to gather network intelligence, run algorithms to perform analysis, and then use the controller to orchestrate the new rules throughout the network [14].

2) Ryu controller: It is an open-source SDN controller designed to increase network agility by making it easier to manage and adapt traffic. The Ryu controller offers software components with well-defined APIs that make it easy for developers to create new network control and management applications. With this component approach, organizations can customize deployments so that specific needs can be satisfied. Developers can promptly and smoothly modify existing components or implement their own to guarantee that the underlying network can meet their applications' changing demands [15]. It supports various protocols to manage devices, including OpenFlow. Its main features are the following:

- Ability to receive and manage events;
- Ability to analyze incoming packets and create new packets to send to the network;
- Ability to create and send OpenFlow messages to reprogram switches.

3) Floodlight controller: It is an open-source SDN controller written in Java. It supports the OpenFlow protocol as a communication interface (southbound interface) with network elements. It is an enterprise-class driver available under the Apache license for almost any purpose and is supported by a large community of developers. Floodlight is designed to work with many switches, routers, virtual switches, and access points that support the OpenFlow protocol [11]. It is a multiplatform system since it works on the Java virtual machine. The Floodlight architecture is modular, with components that include the functionality described in the ODL controller model. It has

a web-based user interface and a Java-based graphical user interface, called Avior. It is one of the most documented drivers today.

C. Benchmark tools for controllers' evaluation

In [12], two tools are proposed for evaluating the controllers, namely Cbench and Hcprobe. These tools are very effective in performing performance, scalability, availability, and security tests.

1) Cbench: It is a tool to monitor OpenFlow controllers through the generation of events. Cbench emulates a configurable number of OpenFlow switches that communicate with an OpenFlow controller to measure different aspects of its performance and latency. Its essential operation consists in that each emulated switch sends a configurable number of new flow messages (OpenFlow packet-in messages) to the OpenFlow controller, waiting for the appropriate flow configuration responses (OpenFlow packet-out messages or OpenFlow messages for modification of flow-mod flows) and records statistics of the time difference between requests and responses, as well as other performance metrics.

2) Hcprobe: It allows for creating scenarios for SDN control tests quickly. It is capable of simulating a large number of switches and hosts connected to a controller. By using Hcprobe, various controller performance indices can be flexibly tested and analyzed. It allows specifying patterns for generating OpenFlow messages (including malformed ones), setting the traffic profile, among others. It is written in Haskell and allows users to create their driver testing scenarios easily. The key features of the Hcprobe are:

- Generation of OpenFlow packages;
- API for designing custom tests;
- An embedded domain-specific language (eDSL) for creating tests.

Besides, Hcprobe provides the framework for creating several test cases to study the behavior of OpenFlow controllers processing different types of messages. It can generate all kinds of messages and reproduce different communications scenarios, even the most sophisticated ones, which cannot be easily reproduced using hardware and software OpenFlow switches. It is useful for developers to conduct driver security testing, regression, and functionality [16].

IV. EXPERIMENTAL SETUP

An ASUS ZenBook computer, with a 2GHz Intel Core i7 processor and 8 GB of RAM was used to make a quantitative comparison between SDN controllers. In this equipment, the Oracle VM VirtualBox tools and a virtual machine were installed using Ubuntu 14.04.06 operating system with 4096 MB of RAM. The drivers to be evaluated were installed on this virtual machine and the Mininet emulator and the Wireshark protocol analyzer with support for OpenFlow, which were used to check the drivers' operation.

We use Mininet to prototyping large networks on a single computer. It is used to create software-defined networks using lightweight virtualization mechanisms. These features allow Mininet to create, interact, customize, and share prototypes quickly. The prototypes should be easily shared with other collaborators, who can run and modify the experiments [17]. Mininet networks run real code, including standard Linux kernel and network applications. Notably, a Mininet-running design can usually go straight to physical switches. It is worth mentioning that different network topologies can be created.

As explained, SDN controllers have different properties, as well, the choice must be subjected to a multi-criteria decision-making process, depending on what characteristics they have and the scope in which they will be implemented. The tests carried out are performance calls; they aim to measure latency, data transfer rate (throughput), and scalability. Latency is defined as the time that a controller needs to respond to a request and is measured in seconds. The transfer rate is the number of transmissions the controller can handle measured in flows/second. Finally, scalability will be measured as the controller's performance against increasing network size and measured as flows/second for a different number of switches in the network. Both for the latency and throughput evaluation, Cbench values of 10,000 hosts and 16 switches are used. For scalability, the number of switches used is varied. In the three test scenarios carried out, the environment presented in Fig. 3 is used, the controller will be ODL, Ryu, or Floodlight.

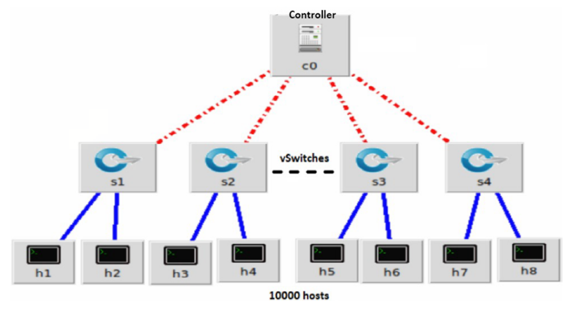

Fig. 3. Experimental Setup

The Cbench benchmark was used to conduct the performance evaluation. Each Cbench run consists of 20 test loops with a duration of 10 secs. We take an average of five runs of each experiment as a result.

V. RESULTS

The obtained results are presented in this section, considering the evaluated parameters: latency, throughput, and scalability.

A. Latency

This test allows a comparison of the latency introduced by different controllers running a learning switch application. For this test, we must use Cbench in latency mode. In this mode, each switch requests a new flow (OFPT_PACKET_IN) and waits for the response (OFPT_PACKET_OUT or OFPT_FLOW_MOD) before requesting the next request. This mode measures the controller processing time in low traffic conditions. In (1), the command to execute Cbench in latency mode is presented; for a driver running on the local machine on port 6633, it must be understood that the drivers, Cbench, and all its add-ons were correctly installed [18].

root@edh-VirtualBox:/openflow/ofloops/cbench# cbench- c localhost -p 6633 -m 10000 -l 10 -s 16 -M 10000 (1)

The results of the average latency obtained can be observed in Fig. 4. Knowing that lower latency is better for the network, it can be seen that the Ryu controller is the one with the highest latency compared to ODL and Floodlight, which have similar latency with a small advantage for ODL that it has an average latency of less than 5 ms. The two drivers can be said to have low latency.

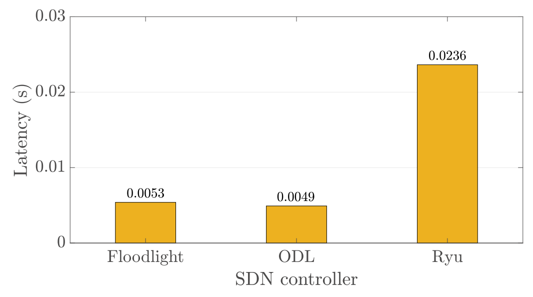

Fig. 4. Average latency per controller

B. Throughput

For this test, Cbench is used in throughput mode (2); the -t option must be added at the end to specify the mode. This mode evaluates the packet flow (number of packets) that the controller can process in a second.

root@edh-VirtualBox:/openflow/ofloops/ cbench#cbench-c localhost -p 6633 -m 10000 -l 10-s 16 -M 10000-t (2)

In throughput mode, each switch requests a new flow (OFPT_PACKET_IN) and keeps as many pending requests as its buffers allow. This mode is used to evaluate the maximum rate of establishment of flows that a controller can maintain [11]. Fig. 5 illustrates the controller response's average results to the messages issued by Cbench in throughput mode to evaluate the ODL, Ryu, and Floodlight controllers. Knowing that the higher the throughput, the greater the efficiency in the network, it can be seen that the Floodlight controller has the highest efficiency in the transfer rate, exceeding 90 000 responses per second; below it is OpenDaylight, reaching a throughput of almost 43 450 response packets per second. On the contrary, the Ryu controller is the one that presents the least number of responses per second. That is, it is the one with the lowest throughput that handles only 12865 responses per second.

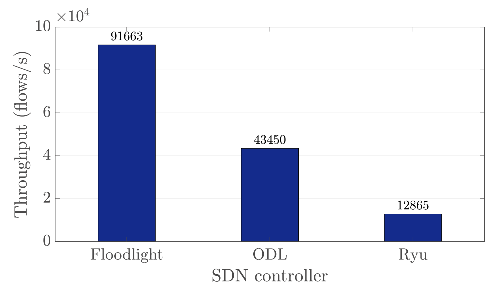

Fig. 5. Average throughput per controller

C. Scalability

This test analyzes the network performance when the number of switches is increased, keeping the number of hosts at 10000. We can define scalability as a controller's ability to handle a large number of nodes and accept new nodes without the need for changes or a decrease in network performance. The three controllers' performance in terms of throughput will be compared when the number of switches in the SDN network is increased to perform the scalability evaluation. The tests were carried out varying the size of the network. We used networks with different controllers in a sequence of 4, 16, 64, 128, and 256 switches, respectively. As in the previous cases, the Cbench tool was used with the command shown in (3).

root@edh-VirtualBox:/openflow/ofloops/ cbench#cbench-c localhost -p 6633 -m 10000 -l 10-s 256 -M 10000-t (3)

Fig. 6 depicts the average results obtained when evaluating the ODL, Ryu, and Floodlight controllers with Cbench in throughput mode for networks with a different number of switches. As can be seen, the Floodlight controller is the one that performs best when the SDN network has few switches; with four, its performance exceeds 100 000 responses per second, although it is evident that its performance drops drastically when the number of switches is increased to 64 (32 765), then 128 and 256 switches to reach an apparent balance in performance. As far as ODL is concerned, it has a substantial performance drop when increasing the number of controllers from 4 to 16 (87 548 to 43 450 flows per second), from where it can be said that it stabilizes its operation by increasing switches. The Ryu controller starts from low performance, 13 610 responses per second with four switches, a performance that does not drop drastically as the number of controllers increases, although, as observed, it ends with a performance of 1 321 responses/s.

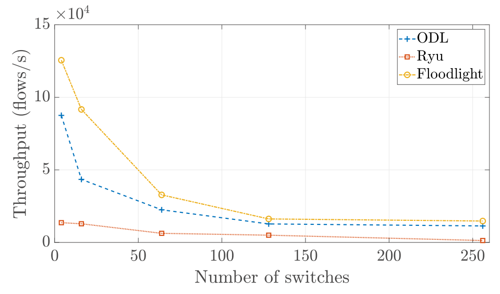

Fig. 6. Scalability evaluation

Finally, an important fact that can be highlighted when choosing a controller is its technical support: the contributions and work that specialized groups dedicated to its development. In this sense, since its appearance in 2013, OpenDaylight has presented a new version each year [19], the result of the lines of code provided by groups of developers sponsored by the "partner" companies, and the users who participate in this community, have a clear advantage over the other controllers as can be seen in Table III. These statistics show that the OpenDaylight community aims at obtaining as soon as possible a reliable, functional, and full-featured platform that will erect itself as the reference SDN controller in the eyes of the entire open-source community.

TABLE III. DEVELOPMENT PER CONTROLLER

	ODL	RYU	Floodlight
Source lines of code	*2'500.000*	*116.000*	*100.000*
Development time (years)	*3*	*4*	*4*

VI. CONCLUSIONS

Benchmarking the performance of a controller is a difficult task. In this work, we qualitatively and quantitatively compare three controllers. With the Cbench tool, it has been possible to compare latency, performance, and scalability between three open-source protocols such as OpenDaylight, Ryu Controller, and Floodlight. The results clearly show that the Ryu Controller presents high latency, low performance, and an inadequate response to small networks and networks with many switches, so its implementation is not recommended. Regarding the Floodlight controller, it shows an acceptable latency response, the best response throughput of packets per second, but it can be observed that in networks with a large number of switches, its performance decreases considerably. In this context, its implementation in small SDN networks can be recommended. OpenDaylight has low latency, the lowest compared to the controllers studied; its performance is acceptable. Although it has a decreasing performance in scalability, when reaching 64 switches, this tends to stabilize and presents good scalability. It is recommended its deployment in networks with a large number of switches or a growth trend. Both OpenDaylight and Floodlight are Java-based and suffer from inherent bugs and problems with the language. For example, in both throughput and latency modes, Cbench sometimes does not provide results, displaying a "controller disconnected" message, displaying only zeros in the output, or even exiting the Cbench loop without finishing the calculations.

VII. FUTURE WORK

We plan to evaluate additional SDN controllers using other driver performance measurement tools such as Qperf, Hcprobe, or Pktblaster. Also, we plan to evaluate the management and security issues of the SDN controllers. Note that we used centralized controllers in this study with a common campus topology. Therefore, evaluating distributed controllers' performance in WAN environments and/or data centers is relevant, and it is proposed as future work for the project members.

VIII. REFERENCES

[1] Y. Li and M. Chen. "Software-defined network function virtualization: A survey", in IEEE Access, vol. 3, pp.2542–2553, 2015.

[2] Open Networking Foundation, "Software definednetworking: The new norm for networks", ONF White Paper, vol. 2, pp. 2-6, 2012.

[3] S. Higginbotham, "Google launches andromeda, a software-defined network underlying its cloud", 2014.

[4] M. Latah and L. Toker, "Load, and stress testing for SDN's northbound API", SN Appl. Sci. 2,vol. 122, 2020. https://doi.org/10.1007/s42452-019-1917-y

[5] C. Laissaoui, N. Idboufker, R. Elassali, and K. El Baamrani, "A measurement of the response times of various OpenFlow/SDN controllers with CBench," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-2, DOI: 10.1109/AICCSA.2015.7507203.

[6] L. Mamushiane, A. Lysko and S. Dlamini, "A comparative evaluation of the performance of popular SDN controllers," Wireless Days (WD), Dubai, pp. 54-59, 2018. DOI: 10.1109/WD.2018.8361694.

[7] Y. Li, X. Guo, X. Pang, B. Peng, X. Li and P. Zhang, "Performance Analysis of Floodlight and Ryu SDN Controllers under Mininet Simulator," 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops), Chongqing, China, 2020, pp. 85-90, DOI: 10.1109/ICCCWorkshops49972.2020.9209935.

[8] O. Salman, I. H. Elhajj, I. A. Kayssi, and A. Chehab, "SDN controllers: A comparative Study", in 18th Mediterranean Electrotechnical Conference (MELECON), 2016. doi:10.1109/melcon.2016.7495430

[9] D. Rodríguez Herlein, N. Talay, C. González, L. Marrone, "Explorando las redes definidas por software (SDN)", in XXII Workshop de Investigadores en Ciencias de la Computación, 2020. http://sedici.unlp.edu.ar/handle/10915/103546

[10] G. Salazar and L. Marrone, "SDN– Redes Definidas por Software", 2019.

[11] Y. A. Marín Muro, "Plataforma de pruebas para evaluar el desempeño de las redes definidas por software basadas en el protocolo Openflow", PhD thesis, Universidad Central "Marta Abreu" de Las Villas. Facultad de Ingeniería, 2016.

[12] A. García Centeno, C. M. Rodríguez Vergel, C. A. Calderón, and F. C. Casmartiño Bondarenko, "Controladores SDN, elementos para su selección y evaluación", Revista Telemática, vol. 13, no. 3, pp.10–20, 2014.

[13] Open Networking Foundation, "SDN Migration Considerations and Use Cases", ONF Solution Brief, 2014.

[14] Z. K. Khattak, M. Awais, and A. Iqbal, "Performance evaluation of opendaylight SDN controller", in 2014 20th IEEE international conference on parallel and distributed systems (ICPADS), 2014, pp. 671–676.

[15] G. V. Sánchez Guindulain et al., "Aplicación de SDN para el control del tráfico de red en base a usuarios", 2017.

[16] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers", in Proceedings of the 9th Central & Eastern European Software Engineering Conference, 2013, pp. 1–6.

[17] R. L. Santos De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. Rodrigues Prete, "Using mininet for emulation and prototyping software defined networks", in 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), pp. 1–6, 2014.

[18] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow", in 9th International Conference on Network and Service Management (CNSM 2013), pp. 122–125, 2013.

[19] R. Farre, "Opendaylight: Beryllium released", 2016 [online] Available: https://blog.rojerfarre.com/2016/02/22/opendaylightberyllium-released/. [Accessed: Sep. 25, 2020].

AUTHORS

Daniel Haro Mendoza

Received the Electronic and Computer Engineering degree from Escuela Superior Politécnica de Chimborazo, Ecuador (2005), the M.Sc. degree in Networking from Escuela Superior Politécnica de Chimborazo, Ecuador (2009), student of the Doctorate of Informatic Science at the Universidad Nacional de La Plata, Argentina since 2018. He was Director of Technologies of Escuela Superior Politécnica de Chimborazo (2014-2015) and Universidad Nacional de Chimborazo (2016-2019). He is currently an Associate Professor with the Universidad Nacional de Chimborazo.

Luis Tello Oquendo

Received the Electronic and Computer Engineering degree (Hons.) from Escuela Superior Politécnica de Chimborazo, Ecuador (2010), the M.Sc. degree in Telecommunication Technologies, Systems, and Networks (2013), and the Ph.D. degree (Cum Laude) in Telecommunications from Universitat Politécnica de Valencia (UPV), Spain (2018). He was Graduate Research Assistant with the Broadband Internetworking Research Group, UPV (2013 - 2018) and Research Scholar with the Broadband Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA, USA (2016-2017). He is currently an Associate Professor with the Universidad Nacional de Chimborazo. His research interest includes 5G and beyond cellular systems, IoT, machine learning.

Luis A. Marrone

Received the Electromechanic orientation Electronic Engineering degree (Hons.) from Universidad de Buenos Aires, Argentina (1978). He is currently Vice-Dean of the Faculty of Informatics, researcher and a full member of the board of directors at the Information Technology Laboratory (LINTI) at the Universidad Nacional de La Plata. His research interest includes Computers Networks, Wireless Communications, Traffic Engineering, Wireless Sensor Networks.