Review of Computer Engineering Studies
Vol. 10, No. 3, September, 2023, pp. 41-47
Journal homepage: http://iieta.org/journals/rces

Comparative Analysis of SDN Controllers: A Study on Installation, Protocols Interaction, Network Topologies Monitoring, and GUI Experience

Daniel Alberto Priano*, María Claudia Abeledo, Javier Guevara, Matías Marsicano, Fabio Sergio Bruschetti, Iara Giniger

Centro de Investigación y Desarrollos en Informática (CIDI), Instituto de Tecnologías Emergentes y Ciencias Aplicadas (ITECA), Escuela de Ciencia y Tecnología (ECyT), Universidad Nacional de San Martín (UNSAM), Ciudad de San Martín, Provincia de Buenos Aires 1650, Argentina

Corresponding Author Email: dpriano@unsam.edu.ar

https://doi.org/10.18280/rces.100302

Received: 28 August 2023
Revised: 6 September 2023
Accepted: 11 September 2023
Available online: 30 September 2023

Keywords:
computer networks, Software Defined Networking, SDN, computer network management, software tools, SDN controller, network configuration management

ABSTRACT

This paper analyses four SDN controllers that support this architecture not only from a technical point of view but also from an academic point of view by including it in the university curriculum. The integration of network controller analysis into an academic curriculum can provide a comprehensive training in theoretical and practical aspects related to network management and SDN technologies. The controllers analyzed were FloodLight, HP SDN VAN Controller, ONOS (Open Network Operating System) and AGILE SDN. Their comparison was based on criteria such as ease of installation, interaction with other communication protocols, ability to monitor network topologies and experience in using their graphical user interfaces. ONOS was found to be the most secure, reliable, robust and scalable controller. Notwithstanding the above, it is important to note that the network technology landscape is constantly evolving, so it is essential to keep updating drivers and comparing features, performance, etc. on these platforms before making a decision. The following are the factors that make ONOS the best choice: 1. Flexibility and customization: ONOS is known for being highly flexible and customizable. This means that you can adapt and customize its functionality to meet the specific needs of your network. Extensions and custom applications can be implemented more easily in ONOS than in some other controllers. 2. Scalability: ONOS is designed to be scalable and can handle large networks with a large number of devices and flows. This makes it suitable for applications in service provider and enterprise network environments. 3. multi-technology support: ONOS is known for its ability to manage a variety of network technologies, including OpenFlow and others. This makes it versatile in terms of support for different network equipment and technologies.4. Active community and continuous development: ONOS has an active community of developers and continuous development. This means that updates and new features are more likely to be found on a regular basis. Among the criteria used, ease of installation was chosen, allowing the controller to be deployed quickly and efficiently, which is beneficial in terms of time and cost. On the other hand, the ability to monitor network topologies provides visibility and control, which is essential for network performance, efficiency and security.

1. INTRODUCTION

Software-defined networking (SDN) simplifies the design, monitoring and management of next-generation networks by separating a legacy network into two planes: the centralized control plane and the data plane. The intelligent, centralized SDN control plane manages the behavior of incoming packet forwarding devices and provides a holistic view of the entire network at a single point. Centralized management in SDN networks together with the possibility to apply scheduling algorithms facilitate the implementation of an adaptive and automated network control model. This can be implemented in three models: a) physically centralized, where the configuration of the entire network resides in a single SDN controller; b) physically distributed but logically centralized, with multiple SDN controllers to manage the network; and c) hybrid, where both models coexist.

Today, the installation and use of data networks is growing rapidly. IP networks are becoming larger and more complex. Today's large IP networks providing connectivity to a huge number of users worldwide pose a major challenge: effective and efficient network management.

Routers play a very important functional role in IP networks. The routing algorithms that route traffic must meet packet forwarding efficiency goals. These algorithms are the control plane of the router and are often referred to as the "router brain". The data plane of the network are the forwarding devices known as physically interconnected switches and routers [1].

SDNs use software-based controllers to manage network traffic or APIs (Application Programming Interfaces) running on specific and dedicated hardware. Traditional networks use these hardware devices such as switches and routers to control network traffic. In contrast, the SDN approach allows defining the required behavior of a network on traditional non-specific hardware through an appropriate programming language (software).

Organizations today can segment one or more virtual networks within a single physical network through virtualization by connecting devices on different physical networks. SDN [2] enables a new way of controlling these virtual networks through a centralized server that manages all the routing requirements.

According to Blial et al. [3], the controller is the central component of an SDN infrastructure because it contains the entire view of the network including data plane devices. The network management software tools used for controller development allow flow control policies to be implemented for each and every device and resource in the data plane. Figure 1 illustrates the SDN controller architecture and its main components. The aim of this work is to investigate and understand the differences between the SDN controllers considered, their characteristics, advantages and disadvantages, where comparing different controllers can be fundamental. This is relevant for students, researchers and practitioners investigating to go deeper into SDN technology from a technical point of view.

The relevance of comparing different SDN controllers or focusing on the use of SDN in managing large IP networks depends on the specific objectives considered, context and needs. There is no single answer, as both perspectives may be relevant in different situations.

Other important components of the SDN architecture are the Northbound and Southbound interfaces [4]. These interfaces interact with both SDN controllers and applications (APIs) [5] from third parties. The SDN Network Operating System (SDN NOS) enables the performance of a network to be programmed by abstracting the essential services and interfaces.

In the control plane, an SDN controller provides state services, topology information, discovery and network configuration. Some of the main features of SDN controllers include:

- Flexible design of centralized or distributed strategies depending on expected network traffic performance.
- East/West APIs [6] to import and export data between controllers incorporating algorithms for data consistency, monitoring and reporting.
- The use of programming languages with a relatively low learning curve. Offering interoperability, multi-threaded execution and memory access and management capabilities.
- Support for OpenFlow and other protocols used in the Southbound interface [7].

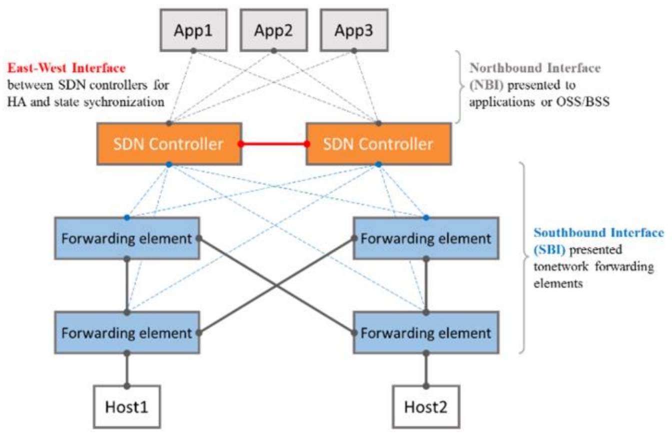

Figure 1. SDN controllers [8]

2. SDN CONTROLLERS

There are multiple software controllers developed for this architecture. The literature consulted characterizes them through the following variables:

1. *Supported OpenFlow protocol version*: Indicates the features and services included in each supported version of the OpenFlow driver. This will depend, for example, on IPv4 or IPv6 support, use of optical links, implementation of tunneling, among other parameters. As of the date of this document, the current versions of the OpenFlow protocol are 1.0 to 1.5.

Always having the latest supported version of the protocol can cause recognition and configuration errors if code changes and enhancements are introduced in new versions. It is therefore essential to check that the applications, transmission equipment and controller support the same version of the OpenFlow protocol.

2. *Southbound API and Northbound API:* Southbound Api indicates which protocols the driver supports to communicate with the data layer (network hardware). In most cases, controllers only support some versions of the OpenFlow protocol as well as some Ethernet network protocols such as ARP, DHCP, BGP, IS-IS. This enables to connect equipment with hybrid configurations such as SDN and Ethernet. This will also allow incorporating connectivity with virtualized networks and cloud functionalities. On the other hand, Northbound API indicates which protocols, APIs or programming languages are supported by the network controller to communicate devices and provide services to the application layer.

3. *Interface type*: Indicate whether it provides a basic command line interface (CLI) or a graphical user interface (GUI) accessible through specific applications or a simple WEB browser (WEB GUI) [9].

4. *Routing applications*: Indicate which applications have been included in the controller or developed by third parties to provide routing services in the network such as STP [10], DHCP, ARP, NAT [11], load balancing, etc.

5. *Metering and monitoring applications*: In the same way, it indicates applications to provide metering and monitoring services such as network monitoring, topology review, statistics management, etc.

6. *Security and trust applications*: Indicates applications that will be able to provide network security and resilience services such as fault correction, access control, application and user behavior tracking, etc.

7. *Cloud integration and virtualization*: Indicates whether the controller supports such applications including virtualization and network functions virtualization (NFV) services.

8. *Maximum number of supported flows*: Indicates the measured maximum number of connections supported by the controller during the performance tests. This value varies depending on the capacity of the hardware used in the tests. Since tests with real network equipment are very expensive, most of these tests have been developed using network simulation software such as Mininet.

9. *Open source or proprietary*: Indicates whether the driver software has been developed and distributed under a license that allows users to have access to the source code to study, modify or distribute it under the same terms and conditions as the original license acquired.

10. *Operating systems*: Indicates the version of the operating system that must be installed to run the driver

software.

11. Multithreading support: Indicates whether the driver performs linear or multithreaded information processing. Multithreaded architectures allow any task to be divided into independent threads that will run simultaneously, reducing the processing time of the entire task.

12. Information consistency: Indicates whether the design of the controller software has specific functions to ensure consistency and stability of the network information. These functions ensure that the information is distributed and executed simultaneously on all nodes of the network. This will prevent possible network configuration errors when resuming communication between the controller and the devices or when changing the role of the controller from slave to master.

13. Usage environments: Indicates the types of networks for which the controller is designed. For example, for small networks with few connection flows, for networks with applications using cloud functionalities, etc.

14. Distributed or centralized control system: Indicates whether the controller design is centralized or not. Centralized controllers offer high consistency of information, but at the same time, they are a single point of failure and vulnerability of the entire network. On the other hand, distributed controllers allow for greater resilience to failures. In these cases, emphasis should be placed on maintaining the consistency of the network information located in each of the controller instances. There are two types of controller architectures: flat and hierarchical distribution. In flat architecture all equipment communicates within the same hierarchical level, and in hierarchical, controllers at a higher hierarchical level concentrate information from controllers at a lower hierarchical level.

15. Fault tolerance: Indicates whether the controller supports recovery from component failure while maintaining service with as little interruption as possible. It is also considered whether its architecture allows redundancy to avoid single points of failure.

16. Manufacturer: Indicates the company that developed the controller.

17. Documentation: Indicates how complete and extensive the driver's documentation is. It can be classified as follows: a) Poor, when there is no information on the developer's website and only one or two articles are published online; b) Good, when the controller has its own website but only contains basic configuration information and up to three articles are published online; and c) High, when the controller has its own website with complete and detailed information on its configuration and use and more than three articles are published online.

18. Type of license of use: Indicates the type of license or permissions of use granted by the developer of the driver. Thus, there are GPL (GNU General Public License) licenses that allow the free use, study, modifications and distribution of the software, but always under the same GPL license. This license requires the publication of the modified source code.

Apache licenses [12] and BSD (Berkeley Software Distribution) allow the same uses as the GPL license but do not require distribution of the modified software under the same original license or open source. The EPL (Eclipse Public License) and the LGPL (GNU Lesser General Public License) allow the combination of free and proprietary software by requiring the publication of the source code only when it is considered a derivative work of the original. Proprietary licenses generally do not allow the study, modification or distribution of their source code, they only allow their use under the express conditions granted by the developer.

19. Programming language: Indicates the language used to develop the controller. The processing speed, modularity, integration capacity with other controllers or applications, etc. will depend on this language.

20. Controller version: Indicates the version to follow the evolution of the controller and to know its particular functions and features.

21. Driver page: Indicates the URL of the driver's web page to find basic information about the manufacturer, driver configuration, developer contact information, etc.

22. Page update date: Indicates the last update date of the website, which gives an idea of the project's timeline and the dedication of its developers [13].

An SDN controller is part of the control plane of the SDN architecture. This can be better understood in Figure 2:

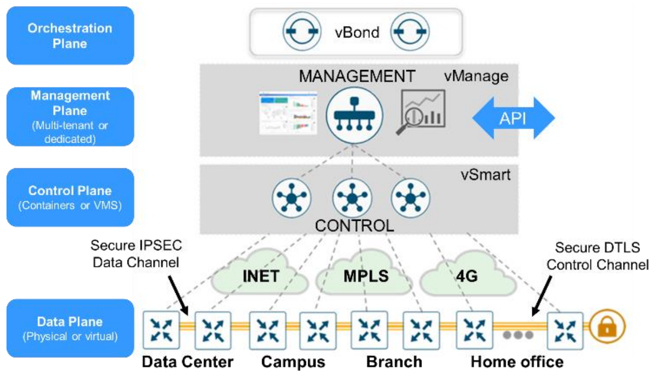
Orchestration Plane
Management Plane (Multi-tenant or dedicated)
Control Plane (Containers or VMS)
Data Plane (Physical or virtual)
vBond
MANAGEMENT
vManage
API
vSmart
CONTROL
Secure IPSEC Data Channel
Secure DTLS Control Channel
INET
MPLS
4G
Data Center
Campus
Branch
Home office

Figure 2. Overview of SDN architecture with SDN controllers [14]

3. CONTROLLERS SELECTED ON THE BASIS OF THEIR LEVEL OF UPDATING

The following group of controllers found in the literature is distinguished by having the most complete and updated information together with a dedicated web page for each of the characterization variables mentioned before. Some of these controllers are developed based on free software and others based on proprietary software. As an introduction, a Software-Defined Networking (SDN) controller is a centralized piece of software that acts as the brain of a software-defined network. Its primary function is to manage and control network traffic, making decisions about data routing and resource allocation. SDN controllers separate the control plane (where decisions are made) from the data plane (where data is sent), allowing for more flexible and centralized management of the network. This separation and centralization allow for greater automation, scalability and adaptability in modern networks. We present the following selected SDN controllers:

1. FloodLight: Created by the equipment company Big Switch Networks as the evolution of the Beacon controller. It was developed in JAVA and released in 2012 under the Apache free software license supporting OpenFlow protocol versions 1.0 to 1.5. This version is supported by programmers from the non-governmental organization Floodlight and Big Switch Networks itself to make the driver evolve. The Open Network Foundation (ONF), created to support the promotion of developments and implementations of the SDN architecture, is leading its evolution and development.

This controller is used in numerous research projects because of its good documentation and easy configuration. It supports hybrid networks and works on physical and virtual switches. It has a graphical interface for managing network topology diagrams, topology monitoring, routing services, load balancing, client isolation, fast failover, quality of service, firewall and access control, among other features [15].

2. HP SDN VAN Controller: It was created and marketed by equipment manufacturer Hewlett Packard in JAVA. HP developers are working on further development, enhancements and bug fixes to the controller. The product is evolving rapidly with new and improved versions. The latest 2.8.8 was released in 2018 and supports OpenFlow protocol versions 1.0 and 1.3 [16]. Hewlett-Packard (HP) has discontinued development and support for its SDN controller called "HP VAN SDN Controller". HP changed its approach to SDN and stopped developing its own SDN controller. Instead, it focused on collaborating with other SDN controller vendors and offering SDN network solutions based on open standards. Despite this, the controller is incorporated into this paper as a case study, given the benefits associated with it, such as interoperability with a wide variety of network equipment. This allowed organizations to deploy SDN solutions in heterogeneous environments. Moreover, centralized network management facilitated the configuration and management of network policies from a single point of control. It also highlights the automation of network tasks, flexibility in scheduling and optimization of resources based on traffic.

In addition, it has its own application shop for SDN networks similar to Android. It has a set of software development tools to create and maintain applications for the controller [17, 18].

3. ONOS: Open Network Operating System is an open-source controller written in JAVA and developed by the Open Networking Laboratory (ON Lab) foundation in 2014. It is based on the FloodLight controller and was developed with programmers from both organizations. Its architecture is distributed and oriented to the administration, configuration and deployment of new services. Its design follows the OSGI (Open Service Gateway Initiative) architecture that includes the necessary abstractions to easily develop new services and functions. It divides the network into 7 subsystems: Devices, links, hosts, topologies, routes, flow rules and packets. It is supported by developers from the ONOSProject.org foundation and they deliver updates approximately every three months.

4. Huawei AGILE SDN Controller: This controller is designed and marketed by the Chinese multinational Huawei. It is based on the ONOS controller and is compatible with the OpenDaylight controller through a REST API interface.

Among its main design features, it allows the interconnection of physical networks with cloud functionalities through an interface with OpenStack. This facilitates the implementation of Network Functions Virtualization (NFV) and Internet of Things (IoT) technologies. Its graphical user interface enables monitoring and management of physical and virtual network topologies for rapid deployment of applications and network changes. It is compatible with OpenFlow, Netconf [19], PCEP [20], BGP-LS [21], SNMP [22] and other protocols. It uses REST API [23] user interface to communicate the controller with the application layer. It also has applications for monitoring, routing, network failover, load balancing, user authentication and control, quality of service, among others [24].

This work presents a selection of 4 SDN controller alternatives. These SDN controllers have been chosen to determine which one best suit the needs of an enterprise network. The authors of this paper have evaluated and selected each decision criteria based on the SDN controller characterization information they have found. In order to decide which SDN controller is the most suitable, the authors have integrated their decision criteria using a methodological approach by obtaining a weighted score calculation (overall priority) of each of the alternatives.

After this description of the controllers that constitute the object of study of this work, the methodology used for the evaluation will be presented. Obviously, two open source and two proprietary controllers have been chosen. In the case of one of the proprietary controllers, despite the fact that the company has discontinued it, it was the right choice, given its characteristics.

4. WORK METHODOLOGY

The methodology used in this work, which is based on clearly defined stages, is presented below. Figure 3 shows the work approach proposed by the methodology where the sequence of the stages of the process can be appreciated. To reduce execution times, the evaluators performed the tests in parallel where each author evaluated all the controllers.

The evaluation matrix was based on the following general criteria:

- *Controller capabilities*: Features, supported protocols, QoS, load balancing, etc.
- *Standards compliance*: Interoperability through open standards.
- *Ease of installation and operation*: User interfaces and tools to configure and use the controller. Documentation provided, clarity and depth.
- *Scalability*: Ability to handle large networks and loads.
- *Stability*: Fault tolerance and failover.
- *Performance*: Latency, throughput, traffic management, etc.
- *Security*: Authentication, authorization, encryption and protection.
- *Integration*: APIs included and integration with third party systems or applications.
- *Community*: Controller user community activity.
- *Compatibility with network hardware*: Existing compatible network infrastructure.
- *Hosted network support*: Support for data centers, network service providers.
- *Cost and licensing*: Consider total cost of ownership.
- *Vendor support and reputation*: Level of support and technical assistance, frequency of software updates. Track record of the controller vendor and reliability of their products.
- *Testing*: Simulation to validate driver functionality, performance and compatibility through use cases.

The Likert scale was used to weight each evaluation criterion. The Likert scale consists of a series of statements on a specific topic. The authors indicated, according to the tests carried out, their degree of agreement or disagreement with each item on a predefined scale of options. For this paper, a typical five-point Likert scale was considered. The extreme points of the scale usually represent options such as "Fully complies" and "Does not comply at all", while the points in between offer more neutral options.

The methodological scheme in Figure 3 shows the conceptual process in which each author arrived at his or her score (perform evaluation), using the factor matrix.

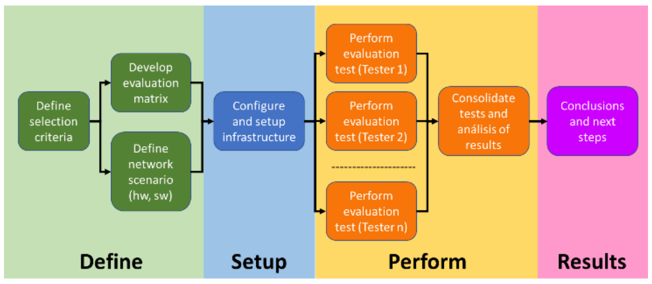
Define selection criteria
Develop evaluation matrix
Define network scenario (hw, sw)
Configure and setup infrastructure
Perform evaluation test (Tester 1)
Perform evaluation test (Tester 2)
Perform evaluation test (Tester n)
Consolidate tests and análisis of results
Conclusions and next steps
Define
Setup
Perform
Results

Figure 3. Methodological approach diagram

In each case, for example, the assessment of ease of installation and operation can be subjective to some extent, as it depends on the experience and expectations of the evaluators. However, by following a structured process as was done in this work, it was possible to obtain more objective and useful evaluations for decision making in the rating.

This can be seen in the schematic methodological approach in Figure 3 and network configuration base scenario for the test in Figure 4, in the next section.

The process included evaluators with knowledge of SDN and the proposed controllers. The authors performed installations and evaluations independently of each other for all controllers. To generate the conclusions, they have worked together.

Authors are considered to be qualified assessors. Each of them has the necessary experience, knowledge and skills to competently and accurately conduct evaluations in the specific field of this work. The qualification of each author/assessor is considered certified as capable of conducting assessments in a professional and objective manner.

This section has detailed the evaluation work, the evaluation matrix, the methodology and the authors' probity in carrying out this work have been detailed. In the next section, closely connected to this one, the test scenarios considered will be discussed.

5. TEST SCENARIOS

In Figure 4, we present the base network topology scenario for testing as a starting point. The testers can modify or adjust it according to the requirements of the software controller.

L3 switches are configured as SD boxes to simulate the WAN environment by performing routing functions. L2 switches are defined as traditional or conventional equipment. The SDN controller configures the L3 switches to manage traffic parameters and hosts are included to verify that data flows as expected. L3 switches are configured with SDN protocols, such as OpenFlow. SDN Controller is installed on a cloud device accessed through Internet.

Although the authors/qualifiers used the same scenario, modifications were made as necessary for the implementation and evaluation of each controller. The authors consider that these modifications were minimal and did not influence the assessment of the items in the matrix presented in the previous section.

The evaluation environment of a controller refers to the context in which the controller is tested and evaluated in order to determine its performance match with the evaluation matrix under consideration. Each author considered and slightly modified the described test scenario. Although these modifications can be considered substantial differences between authors, compatibilities in the criteria used were discussed after the evaluations.

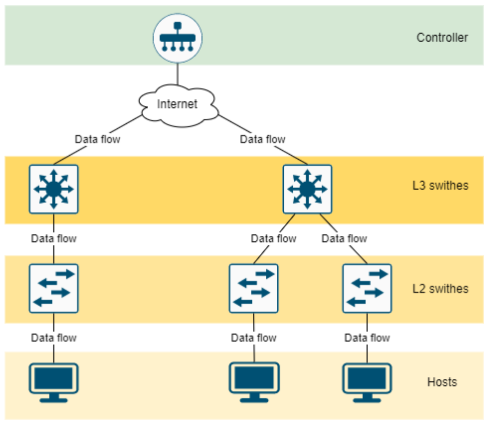

Figure 4. Network configuration base scenario

6. RESULTS OBTAINED FOR EACH CONTROLLER

Once the complete evaluation matrices had been analyzed, it was necessary to adjust some of the values obtained in order to calculate the final results. The results obtained are presented in Table 1 and Figure 5.

Table 1. Evaluation results

Controller	*T1*[*]	*T2*	*T3*	*T4*	*T5*	*Average*
Floodlight	15	14	14	13	14	**14**
HP SDN VAN	13	14	14	15	16	**14.4**
Huawei SDN Agile	14	10	14	15	13	**13.2**
ONOS	17	20	22	20	18	**19.4**

*Tester or evaluator

In this evaluation of SDN controllers, the process was considered to be complex and subjective given the evaluation matrix (section 4). Therefore, it is essential to clarify that the authors had a common understanding of what was found in the matrix items. This helped to minimize the differences in the evaluation criteria and ensure a homogeneous evaluation but with a clear pre-eminence of ONOS.

Thus, the evaluators arrive at similar scores, which means that there is a strong indication of the quality of the solutions evaluated. However, it has been clarified that the differences generated discussions that enriched the understanding of the technology evaluated.

7. CONCLUSIONS

• The most outstanding features of the ONOS controller are its performance and scalability to support growth towards larger networks. It allows the installation of controllers working in clusters, enhancing its capacity for fault tolerance. It is an open-source project and supports a wide variety of southbound protocols to connect to different equipment and technologies.

• In the case of HP SDN VAN, its integration with HP hardware products can be highlighted as a benefit; additionally, it is compatible with OpenFlow to be able to connect with other equipment that supports it. It has a large number of management and control tools, and everything can be managed centrally.

• In FloodLight we can see that it is an open-source project with an active community. It allows an important parameterization that allows it to be adjusted to the user's needs. It provides several APIs for Northbound processing. Its documentation requires more time to learn. The level of integration does not reach that of other controllers.

• The Huawei Agile SDN controller is also highly integrated with Huawei products and can be a limiting factor if you want to use, operate with or migrate to products from another vendor. It has centralized management and a large number of easily configurable functions.

• It was concluded that we recommend updating SDN controllers' information and characteristics after 2023 to choose the SDN controller that best suits future needs.

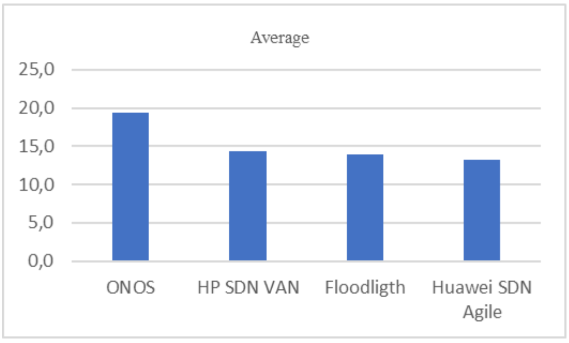

Figure 5. Evaluation results ordered by average

REFERENCES

[1] Paliwal, M., Shrimankar, D., Tembhurne, O. (2018). Controllers in SDN: A review report. IEEE Access, 6: 36256-36270. https://doi.org/10.1109/ACCESS.2018.2846236

[2] What is Software-Defined Networking (SDN)? VMware Glossary. https://www.vmware.com/topics/glossary/content/software-defined-networking.html.

[3] Blial, O., Ben Mamoun, M., Benaini, R. (2016). An overview on SDN architectures with multiple controllers. Journal of Computer Networks and Communications, 2016: 9396525. https://doi.org/10.1155/2016/9396525

[4] Ferro, G. (2012). Northbound API, Southbound API, East/North – LAN navigation in an OpenFlow world and an SDN compass. EtherealMing. https://etherealmind.com/northbound-api-southbound-api-eastnorth-lan-navigation-in-an-openflow-world-and-an-sdn-compass/.

[5] Qué son las "apps" y para qué sirven. (2011). BBC News. https://www.bbc.com/mundo/noticias/2011/04/110408_1336_tecnologia_apps_negocios_celulares_telefonos_inteligentes_dc.

[6] Rodríguez Herlein, D.R., Talay, C.A., González, C.N. (2020). Explorando las redes definidas por software (SDN). In XXII Workshop de Investigadores en Ciencias de la Computación. http://sedici.unlp.edu.ar/bitstream/handle/10915/103546/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y.

[7] Neto, F.J.-B.V., Miguel, C.J., de Jesus, A.C.D.S., Sampaio, P.N.M. (2021). SDN controllers-A comparative approach to market trends. In 9th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2021), pp. 48-51. https://hal.archives-ouvertes.fr/hal-03133692.

[8] Bouguerra, F. (2021). Data center networking: What is SDN? Ubuntu Blog. https://ubuntu.com/blog/data-centre-networking-what-is-sdn.

[9] The Web GUI component. (2021). IBM Documentation. https://www.ibm.com/docs/sv/netcoolomnibus/7.4?topic=components-web-gui-component.

[10] Spanning Tree Protocol. Cisco Technology Supporting. https://www.cisco.com/c/en/us/tech/lan-switching/spanning-tree-protocol/index.html.

[11] Network Address Translation (NAT). Hanna, K.T., Burke, J. (2021). TechTarget Networking. https://www.techtarget.com/searchnetworking/definition/Network-Address-Translation-NAT.

[12] Apache license version 2.0. The Apache Software Foundation. https://www.apache.org/licenses/LICENSE-2.0.

[13] Julio, C., Rodríguez, Q. (2020). Propuesta metodológica para la selección de controladores de redes SDN a nivel empresarial. Universidad Santo Tomás. https://repository.usta.edu.co/handle/11634/30425?show=full.

[14] Cisco SD-WAN design guide. Cisco Solutions. https://www.cisco.com/c/en/us/td/docs/solutions/CVD/SDWAN/cisco-sdwan-design-guide.html.

[15] Floodlight projects. Project FloodLight. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343647/Floodlight+Projects.

[16] HP VAN SDN Controller Administrator Guide. https://support.hpe.com/hpesc/public/docDisplay?cc=cl&docId=emr_na-c04003114-2&lang=es-cl.

[17] Understanding the controller architecture. https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_6/5998-8472admin/content/c_controller-apps-sdn.html.

[18] HPE VAN SDN Controller and applications support matrix. Hewlett Packard Enterprise. https://www.hpe.com/psnow/doc/c04647298.

[19] Information about the NETCONF protocol. SCRIBD. https://es.scribd.com/document/491711300/Net-Conf?utm_medium=cpc&utm_source=google_pmax&utm_campaign=3Q_Google_Performance-Max_RoW&utm_term=&utm_device=c&gclid=Cj0KCQjw7aqkBhDPARIsAKGa0oIDRJw0nbbtkKHQRjJYHsZTho7XHZZWmJeaBg7MwdnTEU0wfnxVQooaArnUEALw_wcB.

[20] PCEP configuration. Juniper Networks. https://www.juniper.net/documentation/us/en/software/junos/mpls/topics/topic-map/pcep-configuration.html.

[21] Border Gateway Protocol Link-State. Cisco Systems. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-16-6/irg-xe-16-6-book/bgp-ls.pdf.

[22] What is SNMP? ManageEngine.

https://www.manageengine.com/network-monitoring/what-is-snmp.html.
[23] What is REST API? Red Hat. https://www.redhat.com/es/topics/api/what-is-a-rest-api.
[24] Agile Controller-DCN. Huawei. https://support.huawei.com/enterprise/en/network-management-control-analysis/agile-controller-dcn-pid-21481886.

NOMENCLATURE

APIs	Application Program Interfaces
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol Link-State
BSD	Berkeley Software Dirstribution
DHCP	Dynamic Host Configuration Protocol
DTLS	Datagram Transport Layer Security
EPL	Eclipse Public License
GNU	GNU's Not Unix software code
GPL	GNU General Public License
GUI	Graphical User Interface
HP	Hewlett-Packard
INET	Internet Networking
IoT	Internet of Things
IP	Internet Protocol
IPSEC	Protocol Suite for Encrypting Network Communications
IS-IS	Intermediate System - Intermediate System Protocol
L2	Level 2
L3	Level 3
LGPL	GNU Lesser General Public License
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NFV	Network Functions Virtualization
NOS	Network Operating System
ON Lab	Open Networking Laboratory
ONF	Open Network Foundation
OSGI	Open Service Gateway Initiative
PCEP	Path Computation Element Protocol
REST	REpresentational State Transfer
SD	Software Defined
SDN	Software Defined Networks
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
Tn	Tester n
VMS	Virtual Machines