International Journal of Scientific Research in Computer Science and Engineering
Vol.10, Issue.3, pp.47-51, June (2022)

Research Paper

E-ISSN: 2320-7639

Controllers Performance Analysis in Software-Defined Networking

Ayman Haggag[1*], Sanaa Awad[2], Ali Saad Gaballah[3]

[1]Department of Electronics and Communications Engineering, Faculty of Engineering, Helwan University, Cairo, Egypt
[2]Deptartment of Curricula and Teaching Methods, Faculty of Education, Banha University, Banha, Egypt

[*]*Corresponding Author: haggag@techedu.helwan.edu.eg, Tel.: +20-1011101699*

Available online at: www.isroset.org

Received: 28/Apr/2022, Accepted: 30/May/2022, Online: 30/Jun/2022

***Abstract*—**Software-Defined Networks (SDN) is a modern networking model characterized by many features by which the network can be configured more easily and managed at lower cost and higher efficiency to cope with the requirements of the current era of technology, which requires much more network automation and flexibility than traditional networks. The technical aspect is a new approach to network management, whereby the network administrator can manage the network in an abstract way away from knowing the technical details in the lower layers. OpenFlow protocol is the protocol used in SDN managements. Several SDN controllers that support OpenFlow now exist. In this research, we compare the features provided by the most popular SDN controllers available now. We apply existing benchmarking and network analysis tools to assess the performance SDN controllers for different network sizes using available SDN emulators such as Mininet.

***Keywords*—** Software-Defined Networking; SDN Controllers; OpenFlow protocol

I. INTRODUCTION

SDNs are a programmable network architecture that enables programmatic and dynamic network control. It is controlled centrally by the so-called network controller (SDN controller) [1]. Therefore, it is based mainly on the separation between the two main pillars of the network: control and command execution. Accordingly, the SDN has three main features: The first is the separation of the execution level from the control plane. The second feature: centralized control of the devices within the network. The third feature: the programming of the central controllers. That is, we have within these networks, as in Figure 1, a part responsible for decision-making and device management, and it is the mastermind of the network (special controllers). Devices that respond only to the commands of the controllers, and can be likened to muscles and are called the physical force of the network such as switches and routers [2].

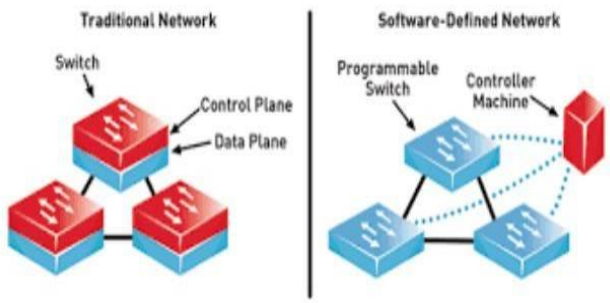


Figure 1. Architecture of Traditional Networks and Software-Defined Networks

The structure of SDNs has been divided into three layers shown in Figure 2, and the following is an explanation of each layer separately [3].

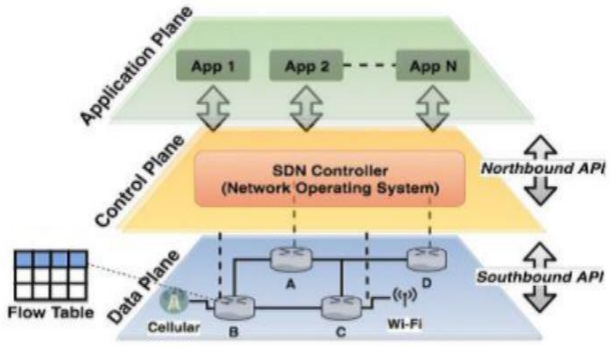


Figure 2. SDN Architecture

A. *Application Layer*

It is the first layer in the SDN architecture, and it consists of the services and applications that you provide to the user, and this layer communicates with the next layer through the Northbound API (Northbound Application Programming Interface). The application layer consists of the services and applications provided by the network to the user. Implementation of service Example: Routing filter ACL and QoS. This layer communicates with that of the control layer via API.

B. *Control Layer*

It is the second layer in the architecture, and this layer consists of central controllers separated from the network infrastructure; Which performs the function of controlling, managing and giving commands to network devices, all of

them are routers and switches, which include the authority to pass data only. It is worth noting that the control layer communicates with the next layer - the infrastructure layer - through the Southbound API (Southbound Application Programming Interface), and this layer uses the Open Flow protocol to communicate with network devices.

C. Data Layer

It is the third and final layer within the SDN architecture, and it consists of virtual and physical network devices such as switches or routers. Devices in this layer receive and execute commands from Layer 2. Devices of this layer must support the Open Flow protocol [4].

D. Northbound Application Programming Interface

A software interface that allows the application layer on top of the SDN architecture to take an overview of the network and manage the operation of the controllers and the network as a whole.

E. Southbound Application Programming Interface

Bridges between the control layer elements and the routing elements in the infrastructure, OpenFlow is the Southern SDN API (3). Figure 3 shows a schematic diagram of SDN application programming interfaces.

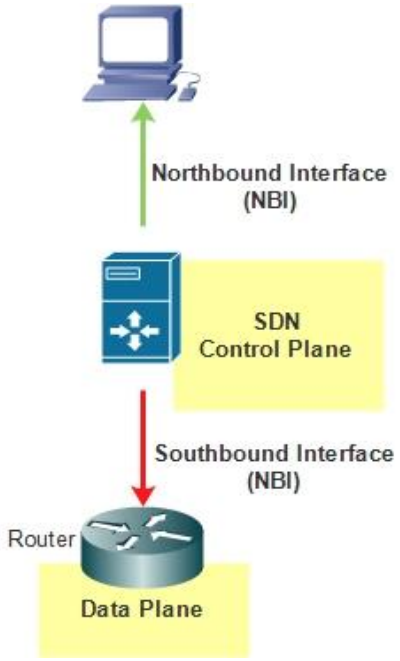


Figure 3. SDN Application Programming Interface

F. OpenFlow Protocol

The separation between the control and implementation layers (the infrastructure layer) necessitated the existence of a protocol that regulates the communication between the two layers, so the OpenFlow protocol was agreed upon; It is the mainstay in SDNs, and its main function is to determine the path of packets based on predefined rules by the network engineer. In addition, the protocol defines the appropriate function, i.e. the switch passes the data packet or discards it (1). Figure 4 shows a schematic diagram of OpenFlow protocol [5].

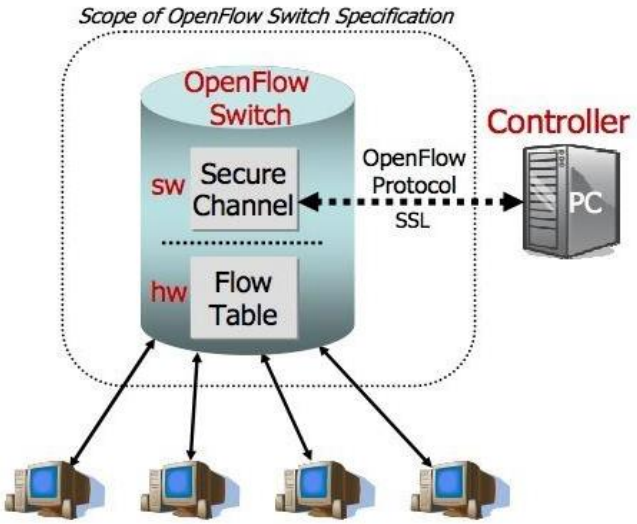


Figure 4. OpenFlow Protocol

This protocol consists of a series of messages sent from the controller to the switch, as well as a series of messages sent in the opposite direction. Those messages give the controller precise control of switching user traffic. A stream as a series of packets transmitted from one network endpoint (or set of endpoints) to another. A single set of rules defines the forwarding actions a device must take for all packets in a given flow. When the console creates a flow, the controller tells the switch how to handle incoming packets.

G. SDN Controller

The controller provides a northbound API for applications, maintains a view of the entire network, executes policy choices, controls all SDN devices that make up the network architecture, and provides a view of the entire network. When we indicated that the controller makes policy decisions about routing, forwarding, redirecting, load balancing, and the like, we meant the controller as well as the applications that use it. We meant the controller as well as the applications that use it when we said the controller makes policy judgments about routing, forwarding, redirecting, load balancing, and the like. Keep your attention solely on the controller. The rest of the devices in the SDN network are stripped of their control functions. They are only used to transfer the services and applications using the OpenFlow Protocol as a common language between them [6].

The rest of this paper is organized as follows, Section I contains an introduction to Software Defined Networking, Section II contain the related work to SDN controller evaluation, Section III describes our proposed evaluation methodology, Section IV describes our results and discussions, Section V concludes our research work with proposed future research directions.

II. RELATED WORK

Since the introduction of the concept of Software Defined Networking, several studies and researches were directed

to analysis and evaluate this new emerging paradigm. Many OpenFlow controllers have been developed and released for research and commercial use. Earlier studies on SDN controllers only focused on propagation delays and ignored the traffic load balancing. Heller et al. [7] tried to solve problems of how to minimize the average and maximum controller–switch latency. The packet processing latency in the controllers is typically longer than the propagation transmission latency. However, in real networks, the round-trip propagation latency is quite significant. The author in [8] provided network optimization for improved performance and speed and in [9] provided security overview of software defined networks: threats and countermeasures. The author [10] in provided benchmarking and performance analysis of Software Defined Networking controllers in normal and failsafe operations using multiple redundant controllers. Authors in [11] provided security analysis of Software Defined Networking without much consideration of network performance.

The main contribution of our research is to provide a comparative study of features of various SDN controllers that support OpenFlow and to assess the performance of the Ryu controller in terms of latency and throughput for various SDN network sizes.

III. PROPOSED EVALUATION METHODOLOGY

Several tools are needed to build SDN networks and to test and compare the performance of various controllers. Table 1 shows examples and explanation of various programs that may be used to simulate or measure the performance SDN networks.

Table 1. SDN simulation and evaluation tools

No.	Product Name	Description
1	Cbench	A special tool for measuring the efficiency of the OpenFlow Controller, by creating a variety of switch devices that send a packet in message to the controller, and then note the received reply and, accordingly, measure the performance of the controller efficiency [12].
2	Oflops	A tool that plays the opposite role, the Cbench tool, which is to measure the efficiency of the OpenFlow Switch, by creating a vitual controller that sends messages to the switch and then notes the reply and accordingly measures the performance and efficiency of the switch device [13].
3	Mininet	The famous Mennet program, which is an emulator designed to create large-sized networks from switch and hosts devices, in addition to its support for SDN technology, is widely spread among researchers to simulate Openflow switches and controllers and establish a connection between them [14].
4	Oftest	A tool used in testing a range of OpenFlow parameters in switch devices that support up to version 1.2 of OpenFlow [15].

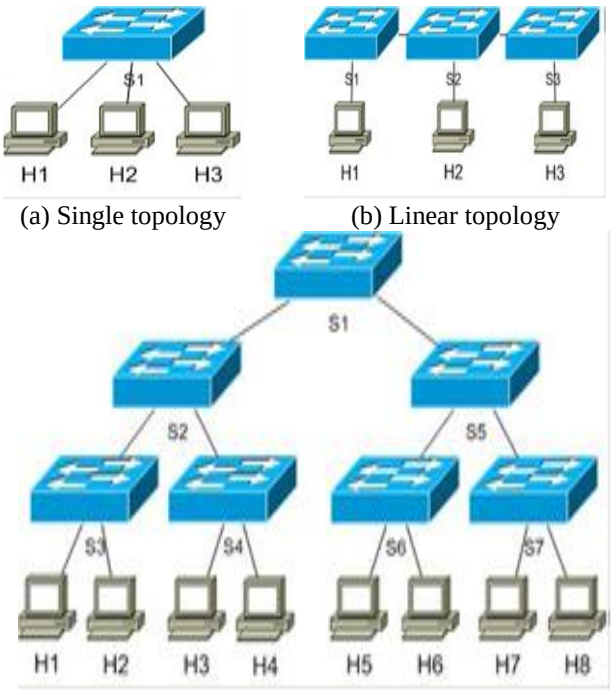
There are many controllers, some are open source and some depend on the vendor. They also differ according to the programming languages in which the network is controlled. The most famous among them are: Pox, Nox, Floodlight and Open Daylight; and Ryu

A. SDN measuring tool

Wireshark can be used to see all data sent between Controller and Switch. This will help to identify faults in the communication between the control plane and the data plane. The nature of this data is in the form of messaging requests from the switch to consult the controller for a particular packet, or the controller's response to a switch request, all done through the previously mentioned protocol, openFlow. To capture this data, it is sufficient to run Wireshark version above 1.12.

B. SDN simulation tool

Mininet can be used to emulate SDN networks by building virtual network devices. Mininet creates a virtual SDN network, running a real kernel, switch, and application code, on a single virtual machine. Figure 5 shows a single, linear, and tree topologies created in Mininet.



(a) Single topology (b) Linear topology

(c) Tree topology

Figure 5. Topologies in Mininet

C. Ryu controller

The Ryu controller is an open source controller and its code is written in Python and is available under the Apache 2.0 license. The Ryu Controller has three layers, application layer, control layer, and infrastructure layer as shown in Figure 6. The Ryu controller communicates with forwarding plane switches and routers using the OpenFlow protocol. The Ryu controller is tested and certified to work with several OpenFlow switches.

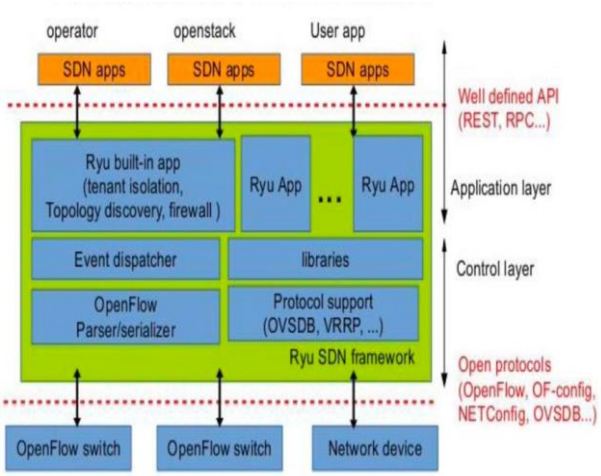


Figure 6. The architecture of the RYU Controller

IV. RESULTS AND DISCUSSION

The performance of five notable SDN controllers, Pox, Nox, Floodlight and Open Daylight; and Ryu, are compared. Comparison is based on open flow supported versions, supported platforms, programming language and open source. The results of this comparison is shown in Table 2.

Table 2. Comparison between SDN controllers

Controller Name	OpenFlow Supported	Platform Support	Prog. Lang.	Open Source
Pox	V1.0	Linux Mac Windows	Python	Yes
Nox	V1.0	Linux Mac Windows	C++	Yes
FloodLight	V1.0	Linux	Java	Yes
Open Daylight	V1.0	Linux Mac windows	Java	Yes
Ryu	V1.0 V1.2 V1.3	Linux	Python	Yes

A. Overall SDN performance analysis

We use Mininet simulator to evaluate the overall performance of the SDN. We use Miniedit to build the network topology, and WireShark to analyze OpenFlow messages.

B. Ryu controller assessment

To evaluate the performance of the Ryu controller, Mininet installed over Ubuntu provides an emulation environment for rapid topology construction using the graphical user interface Miniedit. Two terminals are opened, one for the Mininet and another for the Ryu, for creating network topology and installing the Ryu controller. A topology created for our evaluation is shown in Figure 7. We evaluate the Ryu controller performance at different number of switches and hosts in the network.

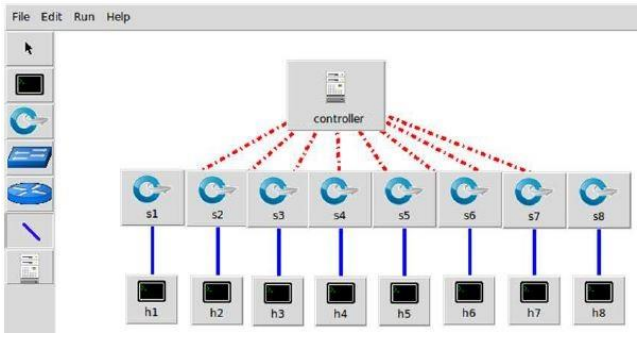


Figure 7. SDN topology created using Miniedit

We use the Cbench tool to measure the performance of the Ryu console in terms of throughput and latency by delivering messages to the console using the OpenFlow protocol.

We use the Cbench program to assess the latency of the Ryu controller performing the test with 5, 10, 15, 20, 25, and 30 switches. The average latency is shown in Figure 8.

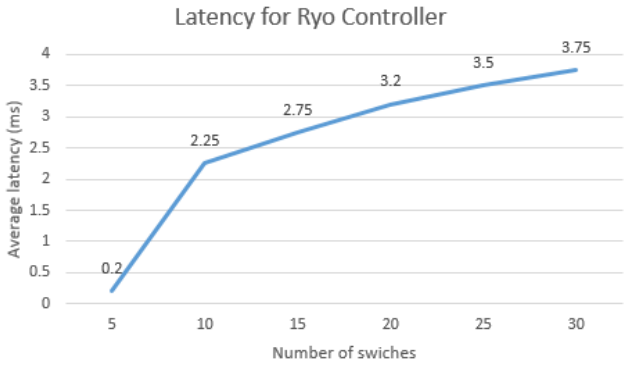


Figure 8. Latency for Ryu Controller

We also use the Cbench program with oflops to assess the throughput of the Ryu controller performing the test with 5, 10, 15, 20, 25, and 30 switches. The throughput is shown in Figure 9.

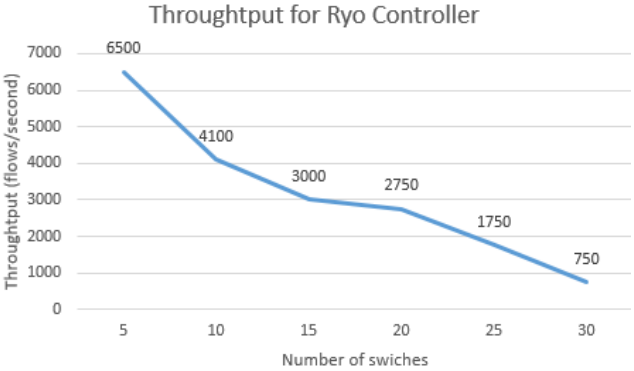


Figure 9. Throughtput for Ryu Controller

After analyzing the obtained results, we can notice from Figure 8 a big increase in latency from 0.2 ms to 2.5 ms as the number of switches change from 5 to 10, however, the increase in latency becomes almost linear with a slow increase rate as the number of switches increase to 15, 20,

25, and 30 switches. However, can see from Figure 9, the throughput values decrease dramatically, going down from 6500 flows/second to only 750 flows/second making this controller rather in efficient in controlling moderate size networks of around 30 switches.

V. CONCLUSION AND FUTURE SCOPE

In this research we provided an overview of Software Defined Networking terminology, we reviewed the most popular SDN controllers that support OpenFlow, and detailed the necessary simulation, evaluation and benchmarking tools for SDN performance analysis. We provided a comparison of the features provided by various SDN controllers with a more in depth analysis and evaluation of the Ryu controller in terms of latency and throughput. We can conclude from our results the performance degradation of the Ryu controller with the increase of the number of switches making it unsuitable for medium size networks. Our plan for future work is to test in details the performance of other SDN controllers and to develop hardware implementations of these controllers to test them in realistic networks rather than using network emulators that may result in inaccurate performance measurements due to dependency on the hosting environment.

REFERENCES

[1] T. Li, J. Chen, and H. Fu, "Application Scenarios based on SDN: An Overview," J. Phys. Conf. Ser., **vol. 1187, no. 5, 2019**.

[2] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Network Functions Virtualization: The Long Road to Commercial Deployments," IEEE Access, **vol. 7, pp. 60439–60464, 2019**.

[3] L. Yue, C. Junyan, L. Chuxin, and L. Xiaochun, "Research on SDN Multi Controller Deployment based on K-means++," J. Phys. Conf. Ser., **vol. 1606, no. 1, 2020.**

[4] A. Mahmoud, A. Abo Naser, M. Abu-Amara, T. Sheltami, and N. Nasser, "Software-defined networking approach for enhanced evolved packet core network," Int. J. Commun. Syst., **vol. 31, no. 1, pp. 1–15, 2018**.

[5] X. Tian, L. Wen, X. Yang, L. Chen, G. Min, and Z. Shu, "Research on Network Routing Control Algorithm Based on OpenFlow and IGP," J. Phys. Conf. Ser., **vol. 2218, no. 1, 2022**.

[6] H. Babbar and S. Rani, "Performance evaluation of QoS metrics in software defined networking using ryu controller," IOP Conf. Ser. Mater. Sci. Eng., **vol. 1022, no. 1, 2021**.

[7] B. Heller, R. Sherwood, and N. Mckeown, "The controller placement problem," Comput. Commun. Rev., **vol. 42, no. 4, pp. 473–478, 2012**.

[8] A. Haggag, "Network Optimization for Improved Performance and Speed for SDN and Security Analysis of SDN Vulnerabilities," J. Comput. Networks Commun., **vol. 7, no. 5, pp. 83–90, 2019.**

[9] A. Haggag, H. Youssef, I. Ali, and F. M. Salem, "Security Overview of Software Defined Networks : Threats and Countermeasures," **vol. 9, no. 5, pp. 348–355, 2022.**

[10] A. Haggag, "Benchmarking and Performance Analysis of Software Defined Networking Controllers in Normal and Failsafe Operations using Multiple Redundant Controllers," **vol. 12, no. 13, pp. 5192–5202, 2021.**

[11] A. Haggag and D. Hanafy, "Network Performance and Security Analysis of Software Defined Networking," **vol. 9, no. 6, pp. 41–47, 2021.**

[12] A. Orogat, I. Liu, and A. El-Roby, "Cbench: Towards better evaluation of question answering knowledge graphs," Proc. VLDB Endow., **vol. 14, no. 8, pp. 1325–1337, 2021**.

[13] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An open framework for OpenFlow switch evaluation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), **vol. 7192** LNCS, no. March, **pp. 85–95, 2012**.

[14] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform," Int. Conf. Commun. Comput. Syst., no. August, **pp. 3–6, 2014.**

[15] Y. D. Lin, Y. K. Lai, C. Y. Wang, and Y. C. Lai, "OFBench: Performance test suite on OpenFlow switches," IEEE Syst. J., **vol. 12, no. 3, pp. 2949–2959, 2018.**

AUTHORS PROFILE

Ayman Haggag was born in Cairo, Egypt in 1971. He received his B.Sc. degree from Ain Shams University, Egypt, in June 1994, M.Sc. degree from Eindhoven University of Technology, The Netherlands, in December 1997, and Ph.D. degree from Chiba University, Japan, in September 2008. Presently, he is an Associate Professor of Communications Engineering at the Electronics Technology Department, Faculty of Technology and Education, Helwan University, Egypt. His current research interests are in the fields of Network Security, Wireless Security, Software Defined Networking and Wireless Sensor Networks.

Sana Awad Master Student, Electronics Technology Department, Faculty of Technology and Education, Helwan University, Egypt. She received her Bachelor degre in Industrial Education from Beni Suef university in June 2006. She started persuing her master study in September 2019. Her research interests are in the field of network performance analysis and Software Defined Networking.

Ali Saad Gaballah Professor of Curriculum, Teaching Arabic and Islamic Studies, Department of Curriculum, Teaching Methods and Instructional Technology, Faculty of Education, Benha University. Professor Ali was born on 13/12/1958. He received his B.Sc. in May 1981, his M.Sc. on 18/2/1987, and his Ph.D. on 8/2/1992. He authered and published serveral books and research articles in the filed of education and linguistics.

