Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Performance Evaluation of Ryu Controller in Software Defined Networks

Alaa Taima Albu-Salih [a]*

[a]*University of Al-Qadisiyah, College of Computer Sciences and Information Technology Diwaniyah, Iraq .Email: alaa.taima@qu.edu.iq*

ARTICLE INFO

Article history:
Received: 25 /12/2021
Rrevised form: 15 /01/2022
Accepted : 05 /02/2022
Available online: 28 /02/2022

Keywords:
Cbench, Latency,
Mininet, OpenFlow,
Ryu, SDN,
Throughput

ABSTRACT

Software-defined network (SDN) is based on separating data plane from control plane and decision-making processes to centralize network control. The OpenFlow (OF) protocol is the most common and widely used in software-defined network for communicating and controlling switches. With this protocol, the switch learns the routing information from the controller and then passes data packets based on this information. One of the most important components of the SDN is the controller, which is the smartest component of the network. Many controllers have been developed since the advent of SDN. One of the most important and famous controllers is Ryu. Due to the importance of the Ryu controller in SDN, in this article, the performance of the Ryu controller in terms of latency and throughput are evaluated. Cbench is used for measuring throughput and latency of this controller.

MSC.41A25; 41A35;

https://doi.org/10.29304/jqcm.2022.14.1.879

1. Introduction

Software Defined Networks (SDN) is a major development in network architecture in line with the requirements of continuous development in the world of networks and speed in performance, as it is based on the principle of separating the data plane layer and the control layer to become the role of devices limited to passing data and executing control commands issued by the controller, which is the main brain in SDN work[1]. As a result of this separation, the network architecture has become better[2]. Fig. 1 shows the main layers of SDN. And these layers are:

1) Application Layer: It consists of the services and applications that the network provides to the user.

2) Control Layer: It contains the controller, which is the basis of this technology, as it is responsible for routing, management and control decisions for all network functions. This layer communicates with the application layer through the Application Programming Interface (API). There are different types of controllers, the most important of which are: Ryu, POX, OpenDayLight, and FloodLight[3][4].

∗ *Corresponding autho :* Alaa Taima Albu-Salih

Email addresses: alaa.taima@qu.edu.iq

Communicated by: Dr. Rana Jumaa Surayh aljanabi.

3) Infrastructure Layer: It consists of network devices (Routers or Switches), receive commands from the control layer and implement them, as this layer communicates with the controller through the (OpenFlow) protocol, which is the main protocol in the work of SDN technology, as it regulates the communication between the controller and the infrastructure[5].

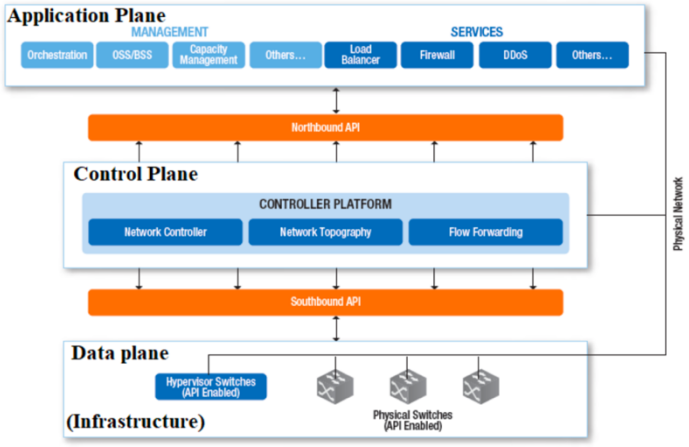

Fig. 1: The main layers of SDN

This research aims to evaluate the performance of the Ryu controller in software defined networks, especially that the controller can be programmed to implement the application we want, which facilitates possible improvements in order to enhance the efficiency of using SDN in the world of networks.

The organization of this paper is as follows: In Section 2, the fundamentals and concepts of the SDN are presented; section 3 describes the OpenFlow protocol; section 4 describes the Ryu controller and its components. Section 5 and 6 describe the emulation tools used in this paper. in section 7, the results of performance evaluation for Ryu controllers are presented; Finally, section 5 concludes the evaluation.

2. Software Defined Networks (SDN)

SDN have provided a major development in network architecture in line with the requirements of continuous development in the world of networks[5][6]. SDN technology is based on separating the control element from the rest of the devices as shown in Fig. 2(b), and this distinguishes the network from traditional networks in which the control element is embedded within the routing element as shown in Fig. 2(b)[7].

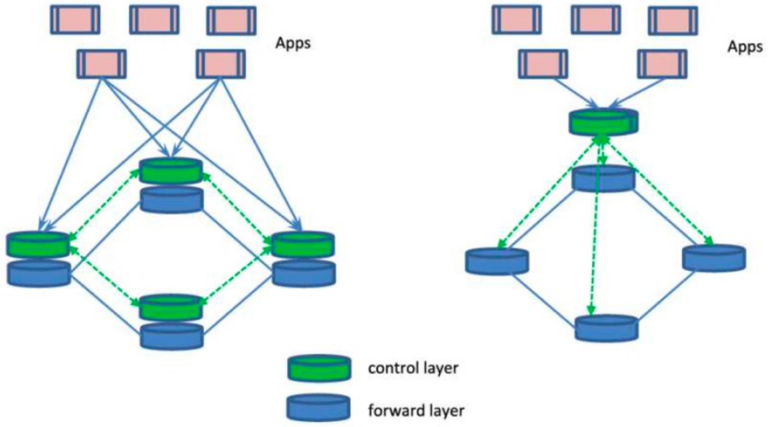

Fig. 2 : (a) Traditional Network; **(b)** Software Defined Network

Therefore, SDN represents a new method to prepare the network, manage it with flexibility and high efficiency, in contrast to the traditional networks[8]. One of the most important components of this new technology is the controller that communicates with the rest of the devices through the OpenFlow Protocol (OF), which works according to TCP and stores routing rules within flow tables so that each of these entrances consists of a set of rules that explain either packet routing or ignoring it according to what the controller decides[9].

3. OpenFlow Protocol (OF)

OpenFlow (OF) is an open-source protocol that regulates all communication between network devices and the SDN controller. It was first introduced in 2009 by the Open Network Foundation (ONF) according to OpenFlow v1.0, and updates have continued to be made[10]. Updates continued until d the current version OpenFlow v1.5 will be release.

The OF protocol works according to TCP, and stores the routing rules of network devices in flow tables composed of flow entries. The flow entry is defined as the event or instruction to be executed on these incoming packets and includes either adding, modifying or deleting an existing path, according to what the controller decides[11].

The OpenFlow protocol is considered the nerve center in software-defined networks, as it manages the switches in the network and allows external components such as the controller to modify the data flow in the network. Recently, many manufacturers have started manufacturing switches, as these switches have tables [12] that display the entry and exit paths (ingress, egress) for all packages for this switch. The OpenFlow protocol makes these tables available to the controller. An OpenFlow switch receives its table inputs and modifications from the controller through a secure channel[13]. Fig. 3 shows main components of OF switch.

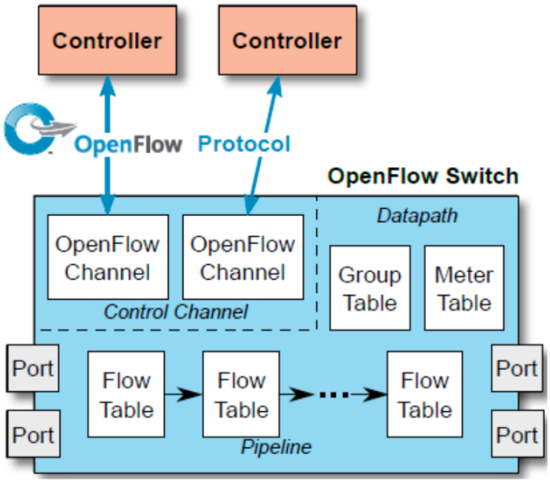

Fig. 3: Main components of OF switch.

When a new packet arrives at the OpenFlow switch, it searches the stream table for a match. If it does not find a match, the switch sends a packet to the controller[14]. The controller processes and marks the packet with the event as follows:

- Adding a new stream for similar packets received.
- Neglecting similar packets.
- Indication of the packet with the queue ID.

4. Ryu Controller:

The most important component of the SDN is the controller, through which the various network applications are programmed. The controllers differ from each other in the programming language that they support, the version of the protocol they operate in, and support for multi-threading, in addition to other differences related to the field of use of these controllers, whether in data centers, cloud computing, and so on. One of the most famous of these controllers is the open-source RYU controller, which is programmed in Python, and supports up to OFF 1.5 of the OF protocol[15].

Figure (4) shows the components of the RYU controller that facilitate the development of network applications, facilitating network management, including OFconfig which is used for the OF protocol settings, and the Library Open Virtual Switch DataBase (OVSDB) which is used for the switch settings to deal with the protocol OF, including

creating, deleting, and modifying rules in a flow table and the NETConfig library that helps implement settings on devices in the network[16].

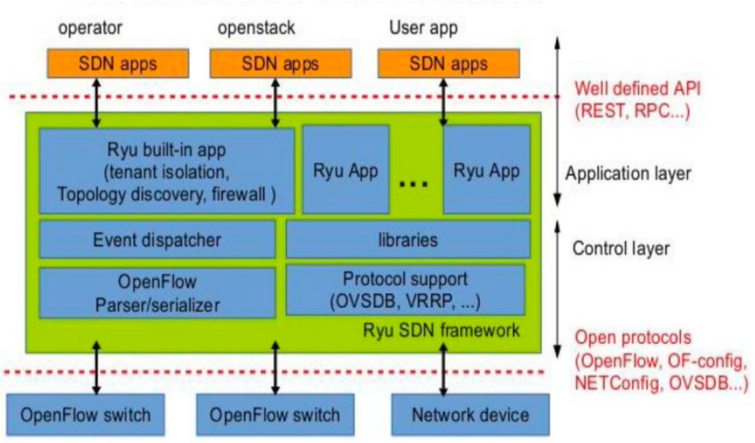

Fig. 4: The architecture of the RYU Controller.

5.Mininet:

It is an emulator that creates a network of virtual hosts, switches, controllers, and links (see Fig. 5)). Mininet hosts can run standard Linux software, and its switches support OpenFlow for highly flexible dedicated packet routing[17]. Mininet supports researching, developing, learning, prototyping, testing, debugging and any other tasks by creating an entire network on a laptop or other device[18][19].

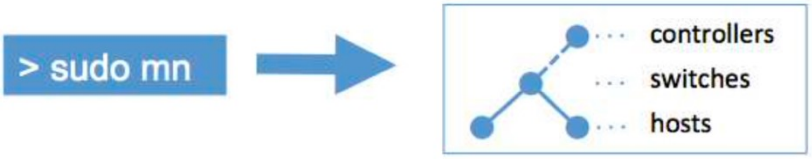

Fig. 5: Mininet.

6. Cbench:

Cbench is the most widely used tool for evaluating the performance of controllers in SDNs. This tool works with two main modes of work, Throughput and Latency, and Cbench uses these two criteria to evaluate the performance of controllers[20]. In throughput mode[21], the largest number of packets that the controller can process is sent, while in latency mode, Cbench sends only one packet and waits for a response to calculate how long it takes the controller to process a single packet[22]. The evaluation was run with the following command[23]:

```
./cbench -c localhost -p 6653 -l 14 -m 20000 -M 100 -s 8 -t
```

Where:

-c : Controller by his name or his IP address.

-p: Controller port number.

-l : The number of times the experiment.

-m : The test time is in seconds.

-M : The number of physical addresses in each switch.

-s : number of switches.

-t : Throughput Mode.

7. Performance evaluation:

The overall performance evaluation of the SDN network was carried out using the Mininet emulator, building the network topology using Miniedit, analyzing the OpenFlow protocol messages using Wireshark, and the last step is to measure the throughput and latency of the Ryu controller using the Cbench tool. In this research, Mininet is used, which is preferred by most SDN researchers, because it is open source and provides easy modeling of nodes, links, controllers and all network components.

7.1 Simulation setup

Evaluation Using computer had an Intel® Core i5 with RAM 8 GB DDR3 on macOS with a VMware Fusion virtual machine installed on Ubuntu 21 64 bit.

7.2 Results and analysis

This paper deals with measuring the performance of the Ryu controller using the Cbench tool in terms of throughput and latency by sending messages to the controller according to the OpenFlow protocol and waiting for the Flow_mod message. The Cbench calculates throughput and latency as follows:

- Cbench sends several packets to the controller (Packet_in) and waits for the controller's response by sending (Flow_mod), and this is repeated several times and then the latency value is calculated.
- As for the throughput, it is calculated by the number of (Flow_mod) messages sent by the controller.

Latency: To measure latency in case of changing the number of switches (2,4,8,16,32, and 64), the following command used to calculate the latency:

```
./cbench -c localhost -p 6653 -s 2 -l 4
```

Which means to measure the latency running the Cbench tool to measure the latency of the Ryu controller which has IP (127.0.0.1) and the port (6653), executing the test if using two switches (s=2) and repeating the test twice (l=4). Figure (6) shows the average latency values according to the number of switches (2,4,8,16,32 and 64).

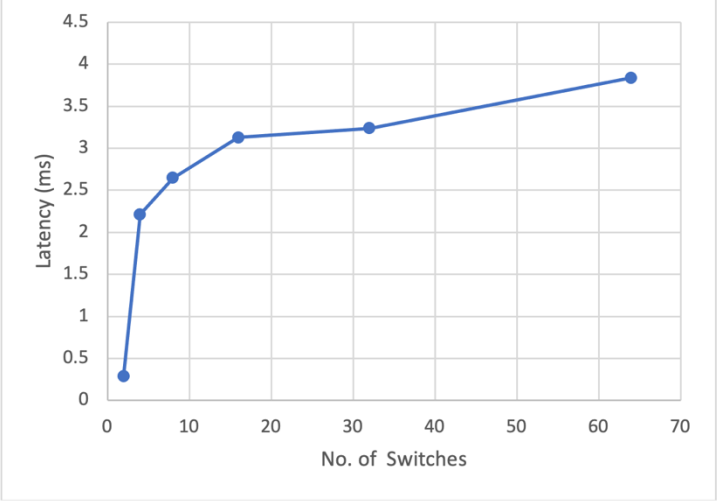

Fig. 6: The latency of the Ryu controller.

Fig. 6 shows that an increase in the number of switches causes an increase in the delay at the controller because any packet incoming to any switch and it will not find a base for it in the switch will be sent to the controller, and thus the time interval between sending the packet_in and flow_mod sent by the controller.

Throughput: To measure throughput in case of changing the number of switches (2,4,8,16,32 and 64), the following command used to calculate the throughput:

```
./cbench -c localhost -p 6653 -s 2 -l 4 -t
```

Which means to measure the throughput (-t) by going to oflops and running the Cbench tool to measure the throughput of the Ryu controller which has IP (127.0.0.1) and the port (6653). The throughput measurement was repeated with different number of switches. The results for throughput are shown in figure (6).

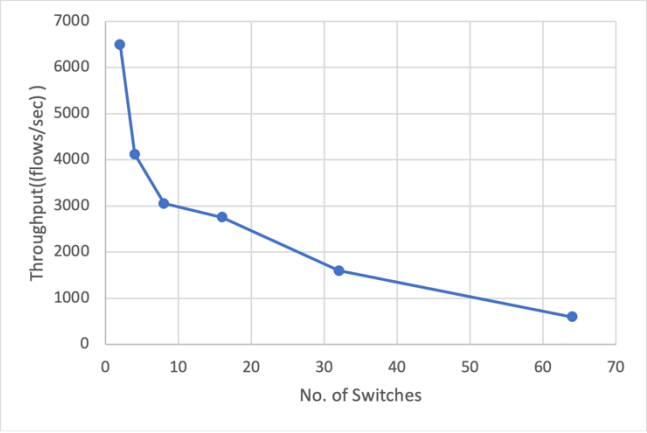

Fig. 7: The throughput of the Ryu controller.

Fig.7 shows the decrease in the value of the controller's throughput with the increase in the number of switches in the network, and this is because the number of flow_mod messages sent by the controller will decrease due to the queue being filled with unprocessed packet_in messages, until the throughput value becomes almost zero when the number of switches is 64 due to a condition of the network being flooded with messages (packet_in) that were not sent to the controller.

8. Conclusion:

In this paper, the performance of the Ryu controller in terms of two performance metrics (latency and throughput) were evaluated. Based on the results obtained in the article, the following conclusion have been obtained:

- The results showed that an increase in the number of switches causes an increase in the latency and a decrease in the value of the throughput.
- The number of switches affects the performance of the Ryu controller, and this effect also varies according to the application used and the network topology.

This work will be extended by evaluating the performance of Ryu controller in term of other performance metrics such as packet drop, round trip time (RTT) and jitter.

9. References:

[1] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.
[2] Goransson, P., Black, C., & Culver, T. (2016). Software defined networks: a comprehensive approach. Morgan Kaufmann.
[3] Zhu, L., Karim, M. M., Sharif, K., Li, F., Du, X., & Guizani, M. (2019). SDN controllers: Benchmarking & performance evaluation. arXiv preprint arXiv:1902.04491.
[4] Mamushiane, L., Lysko, A., & Dlamini, S. (2018, April). A comparative evaluation of the performance of popular SDN controllers. In 2018 Wireless Days (WD) (pp. 54-59). IEEE.
[5] Albu-Salih, A. T., & Khudhair, H. A. (2021). ASR-FANET: An adaptive SDN-based routing framework for FANET. International Journal of Electrical & Computer Engineering (2088-8708), 11(5).
[6] Rana, D. S., Dhondiyal, S. A., & Chamoli, S. K. (2019). Software defined networking (SDN) challenges, issues and solution. International journal of computer sciences and engineering, 7(1), 884-889.
[7] Zhu, L., Karim, M. M., Sharif, K., Li, F., Du, X., & Guizani, M. (2019). SDN controllers: Benchmarking & performance evaluation. arXiv preprint arXiv:1902.04491.
[8] Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. Security and communication networks, 9(18), 5803-5833.
[9] Ominike Akpovi, A., Adebayo, A. O., & Osisanwo, F. Y. (2016). Introduction to Software Defined Networks (SDN).
[10] Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., & Flinck, H. (2018). Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. IEEE Communications Surveys & Tutorials, 20(3), 2429-2453.

[11] Fernandez, C., & Muñoz, J. (2016). Software Defined Networking (SDN) with Open Flow 1.3. Open vSwitch and Ryu,(June 2010), 183.
[12] Bholebawa, I. Z., & Dalal, U. D. (2018). Performance analysis of SDN/OpenFlow controllers: POX versus floodlight. Wireless Personal Communications, 98(2), 1679-1699.
[13] Jha, R. K., Kharga, P., Bholebawa, I. Z., Satyarthi, S., Kumari, A., & Kumari, S. (2014). OpenFlow Technology: A Journey of Simulation Tools. International Journal of Computer Network & Information Security, 6(11).
[14] Braun, W., & Menth, M. (2014). Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. Future Internet, 6(2), 302-336.
[15] Foundation, O.N.: Openflow switch specification version 1.3.1. Tech. rep., Open Networking Foundation (September 2012)
[16] Mamushiane, L., Lysko, A., & Dlamini, S. (2018, April). A comparative evaluation of the performance of popular SDN controllers. In 2018 Wireless Days (WD) (pp. 54-59). IEEE.
[17] Zhu, L., Karim, M. M., Sharif, K., Li, F., Du, X., & Guizani, M. (2019). SDN controllers: Benchmarking & performance evaluation. arXiv preprint arXiv:1902.04491.
[18] http://mininet.org, last visited at January 15, 2022.
[19] Hadeel, M., Alnuami. (2018) "Comparison Between The Efficient Of Routing Protocol In Flying Ad-Hoc Networks (FANET)", Journal of Al-Qadisiyah for computer science and mathematics, 10(1), pp. Comp Page 9 - 15.Al-Somaidai, M. B., & Yahya, E. B. (2014). Survey of software components to emulate OpenFlow protocol as an SDN implementation. American Journal of Software Engineering and Applications, 3(6), 74-82.
[20] Braun, W., & Menth, M. (2014). Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. Future Internet, 6(2), 302-336.
[21] Jaber Jawad Al-Asfoor, M. (2017). A Simulation of a Networked Video Monitoring System Using NS2. Journal of Al-Qadisiyah for Computer Science and Mathematics, 9(1), 117-131.
[22] Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016, April). SDN controllers: A comparative study. In 2016 18th mediterranean electrotechnical conference (MELECON) (pp. 1-6). IEEE.
[23] https://github.com/aorogat/CBench, last visited at January 29, 2022.