A Comparative Evaluation of the Performance of Popular SDN Controllers

Lusani Mamushiane, Albert Lysko, Sabelo Dlamini
CSIR
Pretoria, South Africa
(lravhuanzwo, Alysko, SDlamini1)@csir.co.za

Abstract— Software Defined Networking (SDN) is an architecture that decouples the routing intelligence from the forwarding functions, using an entity called "controller". It is paramount that the performance of the controller is thoroughly understood prior to its deployment. However, the rapid introduction of many new controllers in the research community makes it difficult to choose a suitable controller. This paper studies and evaluates the performance of several popular open source controllers such as ONOS, Ryu, Floodlight and OpenDayLight in terms of latency and throughput using an OpenFlow benchmarking tool called Cbench. Additionally, a feature-based comparison of the controllers is presented. These experimental tests provide a decision making guideline when selecting a controller.

Keywords—Software Defined Networking (SDN); OpenFlow; controller; Ryu; Floodlight; OpenDayLight; ONOS

I. INTRODUCTION

Software Defined Networking (SDN) has emerged as a paradigm that advocates separation of the control plane and data plane. This paradigm shift promises to simplify network management and configuration and to deliver unprecedented scalability benefits. With SDN, the idea is to centralize the routing logic for many switches in a separate entity called a "controller". Based on its global view of the network, the controller optimally programs the forwarding behavior of the data plane. To date, several SDN controller implementations have been developed and deployed in both industry and academia. These controllers have diverse programming languages, and feature sets. Almost all these controllers have support for the OpenFlow protocol used to program routing instructions on the data plane via a secured southbound channel.

SDN controllers are predominantly used for large scale networks (e.g. SD-WAN) where performance is a critical metric. Two of the most important questions frequently asked are (a) how fast can a controller respond to PACKET_IN messages? (PACKET_IN requests are sent by the switch to the controller whenever there is no matching entry in the flow table of a switch and the controller then needs to make a decision); and (b) how many PACKET_IN messages can a controller handle per second? To answer these questions, it is paramount to quantitatively and qualitatively evaluate the performance of SDN controllers to understand their merits and faults so as to provide a clear guideline for selecting the most appropriate controller for a given scenario. This can be achieved by benchmarking the different choices of SDN controllers against various key performance indicators, such as latency, throughput and resiliency.

In this paper, we evaluate and compare the performance of open source controllers based on throughput and latency utilizing an open source benchmarking tool called Cbench. The controllers selected for this performance test are: Ryu [1], Floodlight [2], ONOS [3] and OpenDayLight [4]. These controllers were chosen based on their popularity.

The paper is organized as follows: Section II discusses the previous work on comparison of SDN controllers, Section III highlights our contribution, Section VI describes the features of the selected SDN controllers, Section V presents the test environment and the methodology used in the evaluation as well as the evaluation results and discussion. Lastly, Section VI concludes the paper.

II. STATE OF THE ART

To date, there have been a number of works on benchmarking various SDN controllers. The work done by Tootoonchian et al. [5] was one of the first to carry out a performance evaluation of SDN controllers (NOT-MT, Beacon and Maestro). However, these controllers are no longer used in most SDN implementations and have been replaced by other controllers such as OpenDayLight, ONOS, Ryu, Floodlight and POX. Khondoke et al. [6] presents a feature-based comparison of five topmost controllers (Ryu, Pox, Trema, Floodlight and OpenDayLight). To do this authors collect properties of each controller under evaluation: southbound interfaces, virtualization, GUI, REST API support, productivity in terms of coding speeds, programming language, modularity, operating system, TLS support, maturity, OpenFlow version supported, and OpenStack Neutron support. To carry out the comparison, authors used a Multi Criteria Decision Making (MCD) method called Analytical Hierarchy Process (AHP). In this method users have the liberty to define pairwise priorities of their desired features using a predefined scale. Based on the requirements of Khondoke et al, "Ryu" was selected to be the best controller. However, using this approach leads to subjective results since it is entirely dependent on the features the user prioritizes the most. Thus changing the priorities would lead to different results.

Shah et al. [7] presents architectural guidelines that can be used to improve current controller implementations or to design a new controller. Two architectures were considered: static batching (used by Floodlight, Beacon and NOX) and adaptive batching (used by Maestro). Authors benchmark the key architectural components of several SDN controllers (Beacon, NOX, Maestro and Floodlight) under various performance metrics (latency, switch scalability and thread scalability) in a customized testbed. The evaluation results show that Beacon shows the best throughput performance results. However, in latency mode, Maestro presented better results, as compared to the other controllers.

In [8], Fernandez et al. compares the performances of different SDN controllers (NOX, POX, Trema and Floodlight) in reactive and proactive mode. For all evaluated controllers, the results showed that the best performance is achieved when the controller is operating in proactive mode. This is likely because forwarding rules are installed on the switch in advance unlike in the reactive mode where rules are installed upon receipt of new PACKET_IN requests. While this comparison raises awareness on the importance of controller mode of operation, it is not enough to make a decision regarding the best featured controller.

Rowshanrad et al. [9] evaluates and compares the performances of Floodlight and OpenDayLight under different QoS parameters including delay and loss in various topologies (single, linear, tree topology) and traffic loads (low, medium and heavy load). The results show that OpenDayLight has the best latency results under low traffic loads and also for tree topologies under medium traffic loads. However, Floodlight exhibit the best packet loss results under high traffic volumes for tree topologies and the best latency results in linear topologies.

Shalimov et al. [10] evaluates the perfomrmances of SDN controllers (NOX, POX, Beacon, Floodlight, MuL, Maestro, Ryu) based on latency, throughput, scalability, reliability and most importantly, security. To evaluate controller security, malformed packets are sent to the controller to check how it handles them. From this analysis, authors conclude that sending malformed packets can terminate TCP sessions with switches or even shutdown the controller resulting in a failure of a network segment or even of the whole network. Reliability analysis show that NOX, POX, Beacon, Floodlight and Ryu can endure long-term testing under average traffic load unlike MuL and Maestro controllers. Authors stress the need to improve the above mentioned controllers for production SDN deployments.

Erickson et al. [11] argues that the programing language used by an SDN controller has a significant impact on its performance. The author claims that Java is a good choice because it runs cross-platforms and supports multithreading. Python was ruled out because of its inability to support multithreading whilst the interpreter for C# lacks compatibility with other operating systems other than Windows. C/C++ was also ruled out because of its long runtimes (>10 minutes) and poor memory management. The author then evaluates the performances of several SDN controllers (Trema/C, POX/Python, Ryu/Python, Maestro/Java and Floodlight/Java) and concludes that Beacon which is Java-based has the best performance.

In [12], Salman et al. carries out a qualitative assessment of open source SDN controllers (MUL, Beacon, Maestro, ONOS, Ryu, OpenDayLight, Floodlight, NOX, IRIS, Libfluid-based, and POX). The metrics assessed are latency and throughput performances under varying number of switches and varying number of threads binding to the controller instance. The results obtained suggested that Mul and Libfluid have the best throughput performance while Maestro showed the best latency performance.

Most of these works focused on benchmarking their proposed SDN controllers to verify their advantages over others. However, the dramatic introduction of improved versions of SDN controllers renders past evaluations obsolete. Today most controllers are matured enough in their development necessitating the need to re-evaluate their performances. In this work we will re-evaluate and compare the performances of the most prominent open-source SDN controllers (Ryu, Floodlight, ONOS and OpenDayLight) considering latency and throughput as the key performance metrics. Additionally, since controllers are constantly evolving in terms of supported features, we will also present a feature-based comparison of the aforementioned controllers.

III. CONTRIBUTION

This work is an extension of the recent work presented by Salman et al. [12]. Instead of just evaluating the effect of thread count and switch number on controller performance, this work also evaluates the effect of network load on the performance of Ryu, Floodlight, ONOS and OpenDayLight. Moreover, our evaluation is more up-to-date in that it features the latest versions of the open-source SDN controllers as well as an up-to-date feature based comparison of the controllers. This work is offered to researchers and industry as a guideline in making decisions on the appropriate controller for their desired use case(s).

IV. FEATURE-BASED COMPARISON

This section gives an overview of the basic features of an SDN Controller and presents a feature-based comparison of Ryu, Floodlight, ONOS and OpenDayLight (recorded in Table 1). This table is an updated version of the table presented in [6] and [12], taking into account that new features are constantly being added to SDN controllers. This evaluation is useful to facilitate decision making on the controller that best meets the desired feature criteria. To generate the properties of each controller, a combination of online sources such as journals, conferences, workshops and official websites of controllers were used.

To verify each property, information from different sources was compared against the same property to avoid biased information from the developers. Where there were discrepancies, that feature was eliminated from the comparison altogether.

This feature-based controller comparison does not include other controller implementations like MUL, NOX, POX, Maestro, Beacon and Trema largely because they are poorly documented and not fully matured in their development.

The selection criteria set for the controllers under evaluation includes: Southbound interfaces, REST API, Graphical user interface (GUI), Modularity, Orchestrator support, operating system (OS) supported, Partnership, Documentation, Programming language, Multi-threading support, TLS support, Virtualization, Application domain and architecture.

A. Southbound API

Southbound APIs are used to dynamically enforce forwarding rules and policies on the data plane devices (switches and routers). While OpenFlow is the most popular southbound protocol, it is not the only one available or in development. There have been efforts both in academia and industry to develop other southbound protocols to address the limitations of OpenFlow, such as lack of management functions and support for hybrid SDN, to ensure a smooth migration from the traditional network model to SDN. These include but not limited to NETCONF/YANG, OF-Config, PCEP, BGP/LS, and LISP. As shown in Table 1, OpenDayLight followed by Ryu support the most southbound interfaces compared to other controllers. Floodlight exclusively supports OpenFlow. This restricts its implementation to pure SDN deployments.

B. Northbound API

Northbound APIs are used by applications or higher layer control programs running on top of the controller to communicate with the controller. The application layer is an integral part of the SDN technology since the value of SDN is pinned to the innovative applications it can potentially enable. Northbound APIs are also used to integrate the controller with cloud orchestrators such as OpenStack and CloudStack used for cloud management. Currently REST API has been the most used API [14] and most controllers (including RYU, Floodlight, ONOS and OpenDayLight) support it.

C. Controller Efficiency

Controller efficiency defines the metrics such as performance (e.g. latency and throughput), reliability and load balancing. The centralization of the control plane presents formidable challenges in terms of the aforementioned metrics. Therefore, the distributed framework supported by some controllers aims to address this issue. To date only ONOS and OpenDayLight support the distributed scheme, among the evaluated controllers. This makes them suitable for application across various domains (e.g. campus networks, data center networks, and wide area networks (WANs)).

D. Partnership

To ensure maintenance and quality contributions to improve an SDN controller, it is paramount that the controller is being developed under good and reputable partnership [8]. The financial capacity (coupled with the experience in the networking domain) is the key factor that stimulates trust and consumption of products. Cisco, Huawei, Ericsson, Linux Foundation, etc. are in the list of organizations entering the SDN market and actively contributing to controllers' development. Among the evaluated controllers, OpenDayLight has the most partners. Following OpenDayLight is ONOS. The controllers with the least partners are Floodlight and Ryu.

E. Feature Comparison Discussion

In summary, it is clear that OpenDayLight and ONOS are the most feature rich controllers. Both these controllers can run cross-platforms. Leveraging OSGI, these controllers are highly modular and have excellent runtimes for loading bundles. Moreover, they both have a user friendly GUI for application developers. They also have an active and reputable community that consistently contributes new improvement ideas. Their distributed architecture makes them ideal for realistic SDN deployments. Lastly these controllers support southbound interfaces designed for hybrid SDN and are thus suitable for such application scenarios. However, ONOS does not support cloud orchestration (e.g. OpenStack) which is imperative for virtual resource management.

Ryu has a fair number of features making it ideal for small scale SDN deployments. Its medium modularity, use of Python, centralized architecture and exclusive support for Linux OS limits its deployment to small scale networks.

From the feature comparison results, it is clear that Floodlight has the least features. One of the most important yet limited feature is the number of supported southbound interfaces. Floodlight only supports OpenFlow. Moreover, Floodlight lacks in modularity, does not support the distributed scheme and has no support for cloud orchestration. All these shortcomings likely make it suitable only for small scale applications.

Table 1: Feature-bade comparison of SDN controllers

	Ryu	**Floodlight**	**OpenDayLight**	**ONOS**
Southbound Interfaces	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OFCONFIG, OVSDB	OF1.0,1.1,1.2, 1.3, 1.4,1.5	OF1.0, 1.3,1.4,1.5 NETCONF/YANG,OVSDB, PCEP, BGP/LS, LISP, SNMP, OFCONFIG	OF1.0, 1.3,1.4, 1.5 NETCONF
REST API	Yes (For SB only)	Yes	Yes	Yes
GUI	Yes (Initial phase)	Web/ Java- based	Web-based	Web-based
Modularity	Medium	Medium	High	High
Orchestrator Support	Yes	Yes	Yes	No
OS Support	Most supported on Linux	Linux, Windows , and MAC	Linux, Windows , and MAC	Linux, Windows , and MAC
Partner	Nippo Telegraph And Telephone Corporation (NTT)	Big Switch Networks	Linux Foundation With Memberships Covering Over 40 Companies, like Cisco, IBM,	ON.LAB, Sk Telecom, Cisco, Ericsson, Fujitsu, Huawei, Intel, AT&T, Nec, Ciena, Nsf.Ntt Comunnication
Documentation	Medium	Good	Very good	Good
Programming Language	Python	Java	Java	Java
Multi-threading support	Yes	Yes	Yes	Yes
TLS Support	Yes	Yes	Yes	Yes
Virtualization	Mininet and OVS	Mininet and OVS	Mininet and OVS	Mininet and OVS
Application Domain	Campus	Campus	Data center and Transport-SDN WAN	Data center and Transport-SDN WAN
Distributed/Centralized	Centralized	Centralized	Distributed	Distributed

V. Performance Evaluation

Our evaluation only considers Ryu, Floodlight, ONOS and OpenDayLight. The performance metrics considered are: latency and throughput. The main goal is to investigate which controller gives the highest throughput and lowest latency under various workloads. The evaluation is carried out using Cbench [13], a performance measurement tool to benchmark OpenFlow-compatible controllers. Cbench has two modes of operation: latency mode and throughput mode.

A. Test Environment

Both Cbench and controllers were implemented on the same machine (Intel® Core™ i7-5600U CPU @ 2.6GHZ (4 cores)) to overcome the Ethernet interface speed limitations. 8 GB of memory was available. The system was running Ubuntu 16.04 LTS-64 bit.

B. Methodology

The experiment setup was as shown in Figure 1 below. Cbench was used to emulate different number of switches (1, 4, 8, 12, 16, 20, 24, 28, and 32) which connect to the controller under test (CUT), send PACKET-IN messages and count the number of responses (PACKET-OUTs) received per second as well as the latency. Here the number of unique MACs (hosts) was kept at 1000 MACs while varying the number of emulated switches. Each test was repeated 10 times and an average was used as the result for both modes of operation (latency and throughput). The number of worker threads was kept at 4. The purpose of this test was to investigate the impact of increasing the number of emulated switches on the controller's southbound performance.

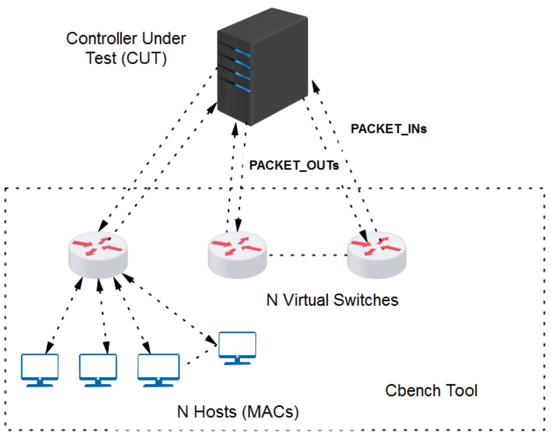

Figure 1: Experiment setup

The second test involved varying the number of MACs (1K, 10K, 100K, 1000K, 10000K) with the number of switches fixed at 16 both in throughput and latency mode. Having a large number of unique source MAC addresses results in a write-intensive workload. Thus this test was done to determine the effect of the number of end hosts on controller performance. Each test was repeated 14 times, each lasting for 10 second. The first 10 seconds (first two loops) are considered controller warm-up and their results are ignored. The number of worker threads was kept at 4. The following example command was used for running tests:

```
./cbench –c localhost –p 6633 –l 14 -m 10000 –M 1000 –s 8 –t
```

where the command line parameters are named as follows:

- c is the controller (IP or hostname);
- p is the controller port number;
- l is the number of loops per test;
- m denotes the test time per s;
- M is the number of mac addresses per switch;
- s is the number of switches;
- t means cbench is running on throughput mode;

C. *Results of Analysis*

This section presents the results obtained after running the tests described above.

Throughput: The throughput evaluation results shown in Figure 2 show that Floodlight and OpenDayLight are drastically affected by an increase in the number of active switches. This is because having a large number of switches causes contention at the data layer which demands high processing power. Ryu's throughput performance is the poorest and remains constant independent of the number of switches emulated. ONOS exhibit the best throughput performance. This is likely because of its inherent support for very large scale networks.

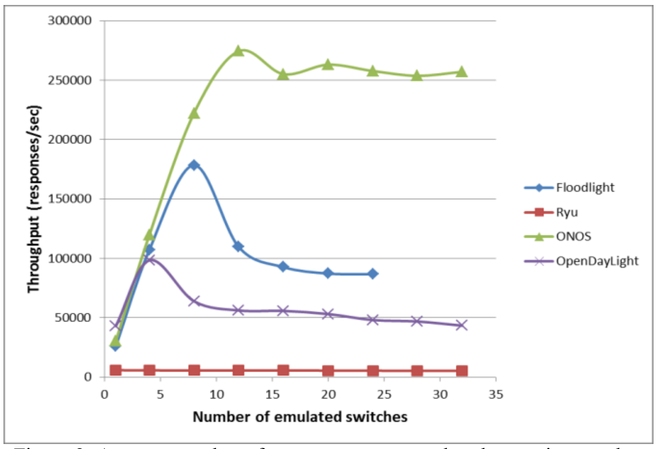

Figure 2: Average number of responses per second under varying number of switches (MACs =1000, threads=4)

Latency: When operating in latency mode, the results shown in Figure 3 suggest that Ryu and OpenDayLight have the best latency. ONOS and Floodlight show the worst latency results as the number of switches are increased.

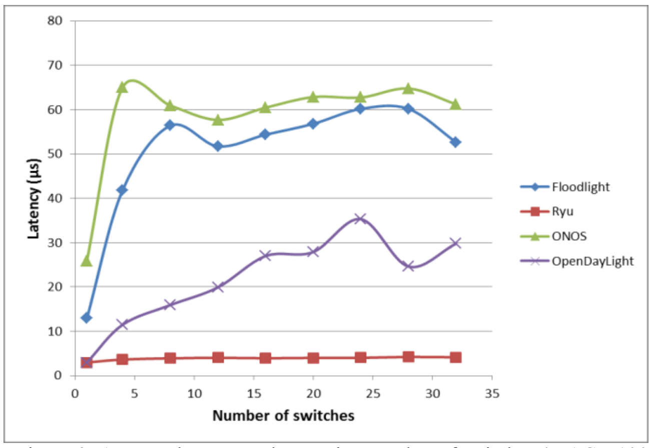

Figure 3: Average latency under varying number of switches (MACs=1000, thread=4)

Scalability: As presented in Figure 4, OpenDayLight, Ryu and Floodlight are significantly affected by the workload resulting from large number of MACs. However, ONOS does not show a similar behavior. The throughput performance of ONOS is almost constant starting from 10K switches. That is likely because ONOS's switch application manages contention between MACs by dividing the network's MAC address table among a collection of hash tables selected by the hash of the MAC address.

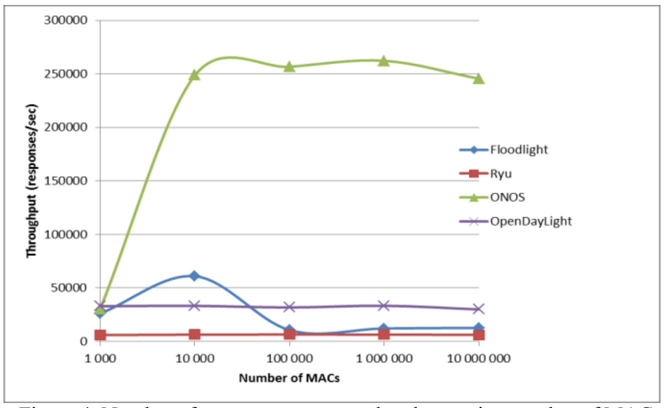

Figure 4: Number of responses per second under varying number of MACs (s=16, threads=4)

Latency: As shown in Figure 5, when tests were performed in latency mode, ONOS exhibited the worse latency performance. Ryu's performance degradation is negligibly small. OpenDayLight latency slightly increased with increasing workload and Floodlight displayed better latency performance compared to OpenDayLight when the number of MACs was set to 100K.

Aggregate Performance: We define a figure of merit for aggregate performance by taking a ratio of throughput (responses/sec) to latency (sec). Figure 6 illustrates the performance results in consideration of both latency and throughtput under varying data plane sizes (number of switches). This results indicate that ONOS has in overall the best performance as the data plane size increases.

Under varying workloads (MACs) as shown in Figure 7, ONOS still shows an outstanding scalability, while OpenDayLight and Ryu display comparatively the same performance for 100 000 MACs and more. Floodlight has the worst aggregate performance for higher workloads.

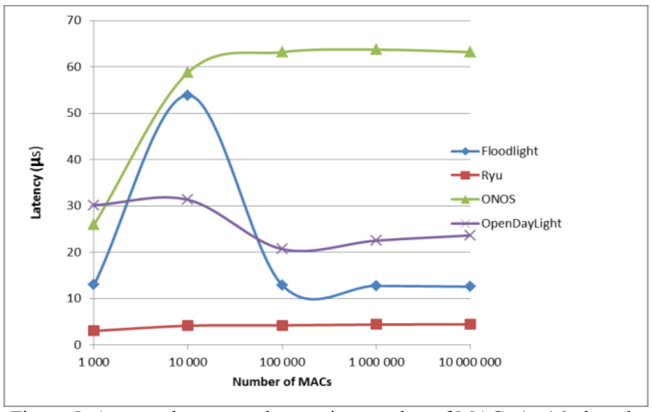

Figure 5: Average latency under varying number of MACs (s=16, threads=4)

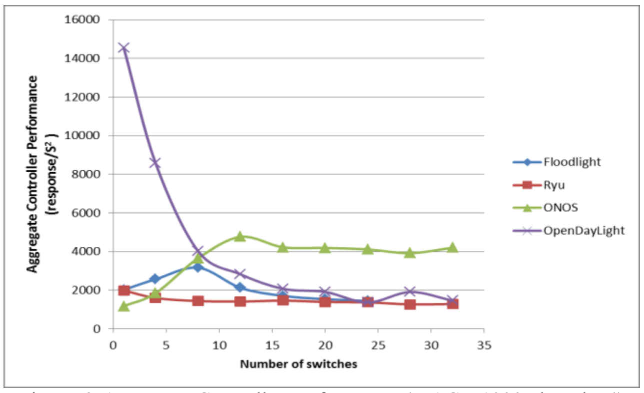

Figure 6: Aggregate Controller Performance (MACs=1000, threads=4)

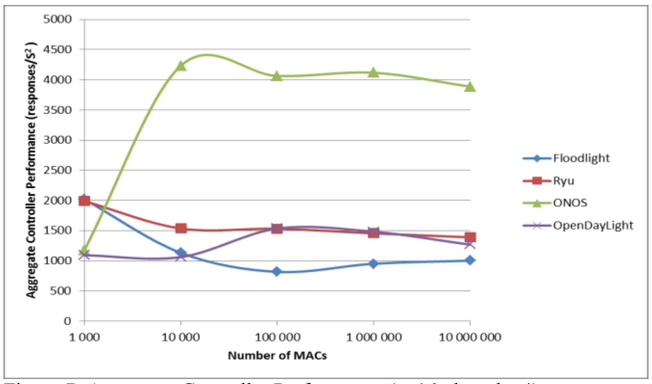

Figure 7: Aggregate Controller Performance (s=16, threads=4)

VI. Conclussion

This paper presents both a feature-based comparison and performance evaluation of widely used open source controller implementations (Ryu, Floodlight, ONOS and OpenDayLight). From the feature based comparison, the merits and faults of the controllers were presented. From this analysis, we recommend the adoption of OpenDayLight since it is more feature rich in terms of interfaces vendor support.

From the performance evaluation, ONOS exhibited the best throughput results showing that it is able to respond to requests more promptly under various traffic loads. However, in latency mode, Ryu displayed the best latency results making it more suitable for delay sensitive applications.

From the above observations, our conclusion is that the choice of which controller to use is entirely dependent on the requirements of the user. This work provides users with guidelines towards making informed controller selection decisions.

Future Work: It is our intention to evaluate the security aspect of SDN controllers in future. This entails sending malformed packets to the controller to investigate the impact on the performance of the controllers. Additionally, this work will be extended by investigating the impact of increasing thread count on the performance of the controller.

References

[1] "Ryu," [Online]. Available: https :// osrg. github.io/ryu/.

[2] "Project Floodlight," [Online]. Available: http :// www. projectfloodlight.org/floodlight/.

[3] "Open Network Operating System (ONOS)," [Online]. Available: https :// wiki .onosproject.org/display/ONOS/Wiki+Home.

[4] "Opendaylight," [Online]. Available: https :// www. opendaylight.org/.

[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado and R. Sherwood, "On Controller Performance in Software-Defined Networks," *Hot-ICE,* vol. 12, pp. 1-6, 2012.

[6] R. Khondoker, A. Zaalouk, R. Marx and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," in *Computer Applications and Information Systems (WCCAIS)*, 2014.

[7] S. A. Shah, J. Faiz, M. Farooq, A. Shafi and S. A. Mehdi, "An architectural evaluation of SDN controllers," in *Communications (ICC)*, 2013.

[8] M. P. Fernandez, " Comparing openflow controller paradigms scalability: Reactive and proactive," in *Advanced Information Networking and Applications (AINA)*, 2013.

[9] S. Rowshanrad, V. Abdi and M. Keshtgari, "Performance evaluation of SDN controllers: Floodlight and OpenDayLight," *IIUM Engineering Journal,* vol. 17, no. 2, pp. 47-57, 2016.

[10] 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.

[11] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, ACM*, 2013.

[12] Salman, O., Elhajj, I.H., Kayssi, A. and Chehab, A., 2016, April. SDN controllers: A comparative study. In Electrotechnical Conference (MELECON), 2016 18th Mediterranean (pp. 1-6). IEEE.

[13] "What are SDN Northbound APIs," Sdxcentral, [Online]. Available: https :// www .sdxcentral.com/sdn/north-bound-interfaces-api.