I.J. Modern Education and Computer Science, **2019, 8, 27-34**
Published Online August 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2019.08.03

Evaluating and Comparing the Performance of Using Multiple Controllers in Software Defined Networks

Mahmood Z. Abdullah, Nasir A. Al-awad and Fatima W. Hussein
Al-Mustansiriyah University/College of Engineering/ Computer Engineering Department, Baghdad, 10001, Iraq
Email: {drmzaali, nasir.awad, fatima.wadaa}@uomustansiriyah.edu.iq

Received: 15 June 2019; Accepted: 05 July 2019; Published: 08 August 2019

***Abstract*—**In Software Defined Networks (SDN) the control plane is removed to a separate device called the controller. The controller is the most important and main part in SDN architecture and large SDN networks may consist of multiple controllers or controller domains that distribute the network management between them. Because of the controller importance, it has been given a proper attention and many studies have been made to compare, test, and evaluate the performance of the controllers. This paper aims to evaluate and compare the performance of different SDN controllers which are Open Network Operating System (ONOS), OpenDaylight, POX and Ryu, using Two performance tests; the first test includes connecting two controllers of each of the four controllers to linear topology with different number of switches; and the other test includes connecting different number of controllers of each of the four controllers to linear topology with fixed number of switches. Then for these tests, the performance in terms of some Quality of Service (QoS) parameters such as average Round-Trip Time (RTT), throughput, and jitter are measured between the two end hosts in each network. After the evaluation of the performance has been completed, it had been seen that the controllers showed different behaviors, and that POX controller showed more stable and good performance results than other controllers.

***Index Terms*—**Software Defined Networks (SDN), ONOS, OpenDaylight, POX, Ryu, Mininet, RTT, Throughput, Jitter.

I. Introduction

Switches or routers in typical computer networks have two logic planes called the control plane and the data plane. The control plane is where the intelligence of the device is placed, and the data plane (sometimes called forwarding or infrastructure plane), is where packets are moved from one network interface on the machine to one of the many other network interfaces on the machine. From a point of view, the control and data plane can be considered to work like the brain and muscle [1].

Software Defined Networks (SDN) differs from traditional networks in the way that the control plane is separated from the data plane; the big aim behind SDN is to give an open interface to enable the development of software that controls the connectivity among network resources and flow of network traffic [2]. Fig. 1, presents the architecture of SDN which contains three planes, the application, the control, and the data plane; application and control plane communicate with each other through northbound Application Programming Interface (API), while control and data plane communicate with each other through southbound API.

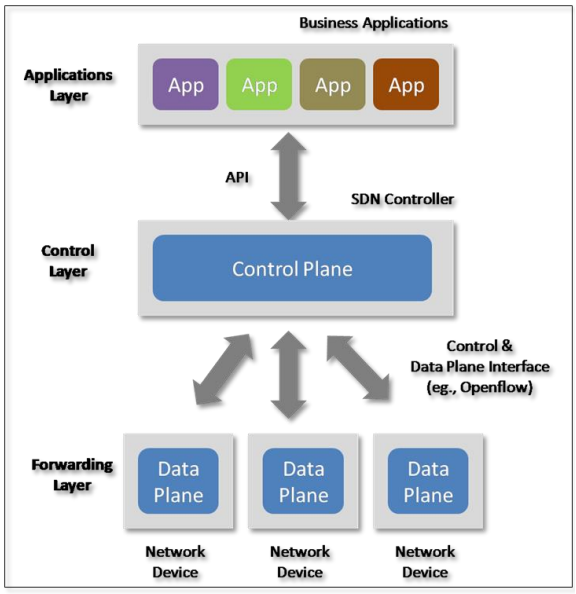


Fig.1. Software-Defined Network Architecture

The most popular standard example of southbound API is OpenFlow. Most projects related to SDN assume that the communication of the controller with the switches is OpenFlow based. OpenFlow defines how the controller adjusts the network and how it should interact with devices at data plane [3].

Data plane consists of network devices like switches and routers and they are simple packet forwarding

Copyright © 2019 MECS

devices, these switches contains flow tables that are used to manage the flows of packets; the functions that are executed on incoming new packets are determined by these flow tables after matching the packets to a particular flow [4,5].

Network intelligence in the form of software control program, referred to as the Controller, reside in the control plane [5]. The SDN controller is the main governing entity in the whole SDN ecosystem [6]. The controller is designed to control the data plane and receive from the application layer the necessary elements to determine the type of control that needs to be applied. Controllers have information about interconnection between network devices, global view of them, and best paths between hosts. Having this single global map of the network enables the controller to make swift, intelligent, and agile decisions with regard to flow direction, control, and speedy network reconciliation when a link fails [1].

An important role of an SDN controllers is to make forwarding decisions or set up rules for packets that arrive at switches and pass these decisions or rules down to the switches to execute them. Also, the controller has global controlling and viewing on the entire network [7].

Many SDN controllers exist nowadays and the usage purpose of such controllers is different. There is a necessity to compare and evaluate the different controllers because of the importance of these controllers [8]. In this paper, the performance of (ONOS, OpenDaylight, POX and Ryu) controllers will be evaluated and compered.

The other sections of the paper are arranged as follows: Section II discusses the related works, Section III shortly review (ONOS, OpenDaylight, POX and Ryu) controllers, and Mininet emulator, Section IV talks about the two performance tests and shows of results of these tests. At last, in Section V conclusion is presented.

II. Related Works

A review of the studies that have been done in the past years for evaluating and comparing the performance is presented in this section.

D. Turull et al., in [9] introduced a collection of measurements for Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic to make comparison between Trema, Floodlight, and NOX controllers. The RTT, TCP transfer time, and UDP packet losses were measured. Finally, Equivalent Packet Losses (EPL) was introduced as a measurement of how often packet losses occur for UDP traffic. The results showed large differences in performance between controllers, and that performance depends on switch-controller delay and flow set-up strategy.

Y. Zhao et al., in [8] selected five centralized controllers (POX, NOX, Beacon, Floodlight and Ryu) and used Controller benchmark (Cbench) tool to test the performance (throughput and latency) of these controllers in cases of single-thread and multi-thread, with different number of switches. Finally, they measured the fairness of service of the controllers.

A. Stancu et al., in [10] measured and compared the performance of four SDN controllers (ONOS, OpenDaylight, POX, and Ryu). A tree topology with 15 switches and 16 hosts was used to be connected to the controllers. The switches were instructed to act as a simple hub in the first phase of the test, and in the second phase as a simple L2 learning. In each phase, the average RTT and the TCP bandwidth between the two end hosts of the topology was measured using Ping and Iperf.

S. Rowshanrad et al., in [11] presented an evaluation and comparison of some Quality of Service (QoS) parameters of Floodlight and OpenDaylight controllers. The delay and packet loss of the controllers were measured in single, linear and tree topologies, and in different network loads using Mininet emulator. The results of the comparison showed that the controllers had a competitive behavior.

O. Salman et al., in [12] conducted a performance comparison using Cbench tool among many open-source controllers like: POX, OpenDaylight, Floodlight, Ryu, and other controllers. This test was made in two modes, throughput and latency; in the first mode, the number of switches was changed, and in the second mode, the number of switches and threads was changed.

A. Jasim and D. Hamid, in [13] studied and evaluated the performance of four controllers (Open-IRIS, Open-MUL, Beacon and Floodlight), using custom topology, and then improved the performance of the network by means of QoS method with Floodlight. The performance evaluation was done in ICMP, TCP and UDP traffics by using Iperf and Ping, and the measurements was done in two cases: idle network and with Background Traffic (BT) network. The results showed that the controllers had different behaviors, and the performance of Floodlight got better when QoS was used.

III. Review of Selected Controllers and Emulator

The selected controllers in this paper ONOS, OpenDaylight, POX, and Ryu are presented in this section along with Mininet emulator. Also, a summary of the main characteristics of these controllers are presented in Table 1.

A. ONOS

ONOS is funded and supported by a number of vendors and service providers, including AT&T, Intel, NEC, Nippon Telegraph and Telephone (NTT) Communications, and many others. It is an open source community written in java that was released in 2014, and it provides Java-based and web-based Graphical User Interface (GUI) and system applications as well [14,15].

B. OpenDaylight

The OpenDaylight which is an open source controller that is programmed in Java. The project started in early

Copyright © 2019 MECS

2013 and was originally led by IBM and Cisco and was hosted under Linux Foundation and is currently supported by several vendors (e.g., NEC, VMware, Huawei, and others) [14,16,17].

C. POX

POX (Pythonic Network Operating System) is an open source OpenFlow controller used by various SDN engineers and developers, and its main objective is research. POX is NOX's younger sibling, NOX controller was developed by Nicira based on C++, while POX is developed using python. POX is also a platform used for prototyping and rapidly developing network applications [5].

D. Ryu

It is an open source framework programmed in Python and developed by NTT Corporation [18]. It has well-defined API, software components, and logically centralized controller [5]. Ryu supports different protocols and presents fair features, which makes it suitable for research applications and small businesses. However, its use in writing applications for real market is limited because of its inability to run cross-platforms and lack of high modularity [12].

Table 1. Comparison based on Features among controllers [12]

Name of the Controller	Written programming language	GUI	OpenFlow version Support	Developed by
ONOS	Java	Exist	OpenFlow 1.0, 1.3	Ciena, ON.LAB, AT&T, Ericsson, Fujitsu, NEC, Cisco, Huawei, NTT, Intel, and others
OpenDaylight	Java	Exist	OpenFlow 1.0, 1.3, 1.4	Linux Foundation with Memberships Casing more than 40 Corporations, Such as NEC, IBM, Cisco and others
POX	Python	Exist	OpenFlow 1.0	Nicira
Ryu	Python	Exist	OpenFlow 1.0, 1.2, 1.3, 1.4	NTT Corporation

E. Mininet

Mininet is a freely available open source network emulator [19]. It is a popular SDN platform that researchers use due to its flexibility, availability, and simplicity. Furthermore, it is devoted entirely to OpenFlow architecture [20]. In Mininet the user is allowed to create, customize and share various topologies that consists of controllers, switches, routers, links, and end-hosts, and perform tests on them very easily.

Mininet contains predefined common topologies such as single, linear and tree. Additionally, custom topologies can be created [9,19,21]. Mininet can be connected to a remote controller, and there are also local controllers. Mininet also includes a Command Line Interface (CLI) and a simple GUI editor called MiniEdit.

IV. Performance Evaluation Tests

In this paper two tests have been made on four controllers. These tests were done in Dell Laptop with 8 GB of RAM and Windows 8.1 pro 64-bit installed operating system. The used virtual operating system was Ubuntu 14.04 (64-bit) installed in Virtual Box, with allocated base memory equals 5000 MB.

In first test, linear topology with different number of switches (or hosts) will be connected to two controllers of each one of the four controllers. Then in the second test, different number of controllers will be connected to a linear topology with fixed 64 switches. For each of these two tests, the basic network performance parameters (average RTT, throughput, and jitter) will be measured. The reason that these tests are only limited to linear topology is because hand-written python code was used to start it. For single and tree topologies it will be hard to write such code because two controllers are not allowed to control a predefined single topology and for tree topology it is hard to write python code to start such topology with different depth and fanout.

The designed network for the first test is shown in Fig. 2, where two controllers (C0 and C1) of each of (ONOS, OpenDaylight, POX and Ryu) controllers will be connected to linear topology with number of switches (2, 4, 8, 16, 32, 64 and 128), This network is started using python code that askes to enter an integer number (N) to specify the number of switches and hosts required to create the linear topology that will be connected to the two controllers. It should be mentioned that in this test the case of (128) switches for ONOS controller will not be evaluated and compered with the other controllers due to memory limitation in the used test environment.

And the designed network for the second test is shown in Fig. 3, where (1, 2, 4, 8, and 16) controllers of each controller will be connected to linear topology with fixed 64 switches. This network is started using a second python code that askes to enter an integer number (N) to determine the number of controllers, then these controllers will be connected to an equal portion of the 64 switches.

For example, if the entered number (N) equals (4) then four controllers (C0, C1, C2 and C3) will be connected to the network and each of them will connect to 16 switches, the (j) in the figure is a counter that starts from the number (2) and represents the reduplication of 64 switches divided by the entered number of controllers (N), to clarify this in this example, C0 will be connected to S1-S(64/4) which is (S1-S16), C1 will be connected to S(64/4+1)-S(64/4*2) which is S17-S32, C2 will be

Copyright © 2019 MECS

connected to S(64/4*2+1)-S(64/4*3) which is (S33-S48), and C3 will be connected to S(64/4*3+1)-S(64/4*4) which is (S49-S64); this is only for clarification of who the switches is divided between the controllers.

It also should be mentioned that in this test only POX, Ryu, and the case of (1, 2, and 4) ONOS controllers will be evaluated and compered due to same reason of memory limitation in the used environment.

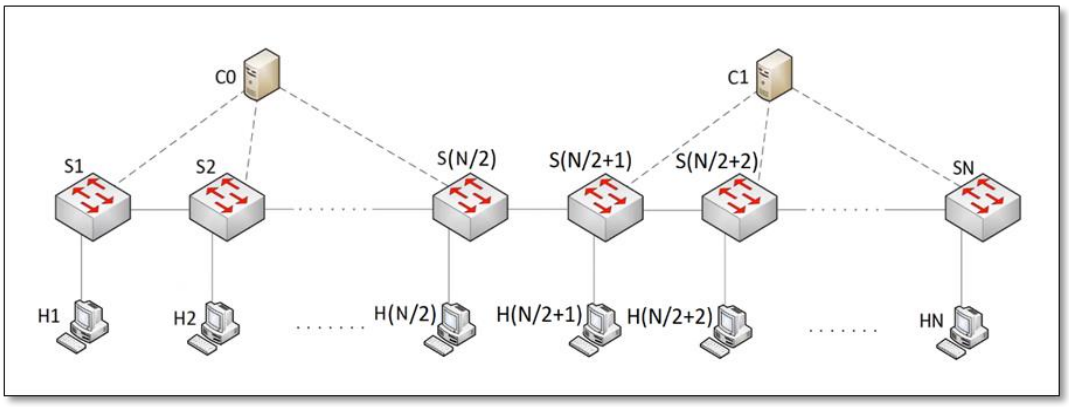


Fig.2. Setup of the designed network for first test

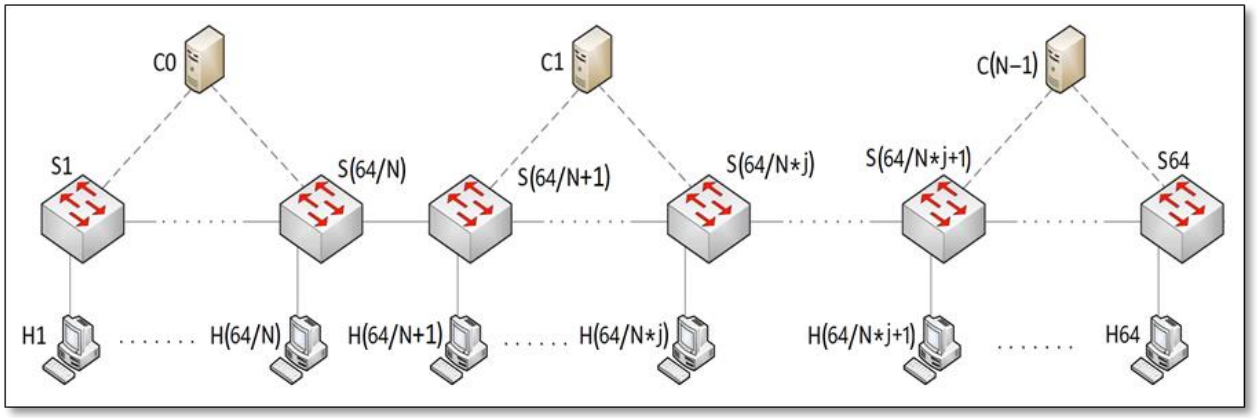


Fig. 3. Setup of the designed network for second test

A. Implementation of the Performance Tests

For each mentioned test, the basic network performance parameters (RTT, Throughput, and Jitter) will be measured to evaluate the performance of the controllers, these parameters are measured for each number of switches and controllers in the two tests between the two end hosts of each network (i.e. between h1 and h2, h1 and h4, …).

1) RTT Measurement

RTT, also called Round-Trip Delay, is the time required for a packet to travel from source to a destination and back again. For each of the two tests, the average RTT in milliseconds (ms) is measured using Ping command. Ping is a very common tool used for checking the connectivity between two hosts in a network and to determine, host reachability, network congestion and travel length. Ping works by sending an ICMP echo-request packet to an address, and then waiting for an ICMP echo-reply [22]. For each test, the RTT is measured two times; one with default ping command parameters and the other with larger packer size and smaller time interval, to see the effect of increasing the load on the response of the controllers.

2) Throughput Measurement

The second performance parameter is the Throughput. Throughput defines how much useful data can be transmitted per unit time; it is equal to the bandwidth if there is no protocol; however, in most practical cases the throughput is less than the bandwidth [23]. For each test, the throughput in Megabits per second (Mbps) is measured using Iperf command by making a TCP connection between iperf client and server, this command is also repeated two times; one with default parameters and the other with larger TCP Window size, which is the amount of data that can be buffered during a connection.

3) Jitter Measurement

Finally, the Jitter (which is the variability in delay of the packet) in ms is measured using Iperf, by making a UDP connection between iperf client and server as, UDP does not use any algorithm to ensure the arrival of the packet to the destination and sends datagrams one after another without retransmitting [24]. For each test, Jitter is

Copyright © 2019 MECS

also measured two times; one with default parameters and the other with larger UDP buffer size.

However, it should be mentioned that the performance in terms of RTT, throughput, and jitter should not be the only factors used to choose among the different controllers; other factors like the reliability and usability is also important [12].

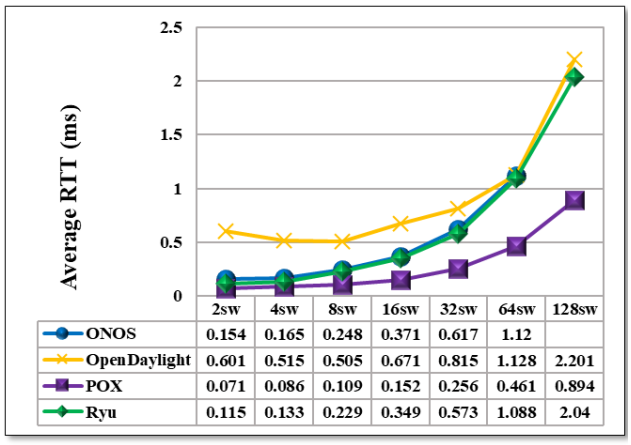
B. Analysis of Results

The results of performance comparison of the controllers when using different number of switches and controllers in the two tests that was mentioned are described as follows:

1) Results of the First Test

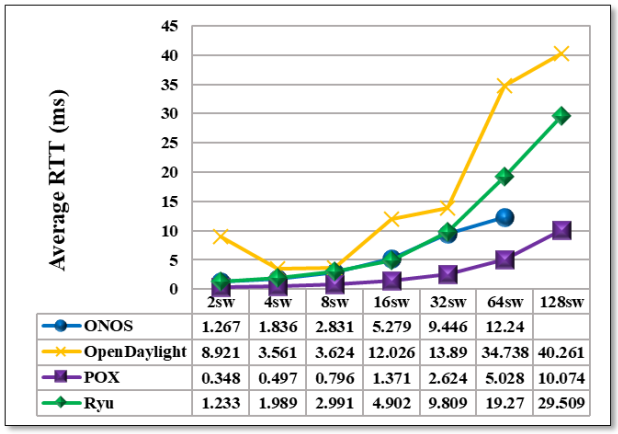
The results of measuring the average RTT, Throughput, and Jitter when connecting two controllers of each of (ONOS, OpenDaylight, POX, and Ryu) controllers to linear topology with different number of switches will be presented for comparing and evaluating the performance of these controllers to see the effect of using two controllers on these parameters.

Fig. 4 and Fig. 5, present the results of average RTT measurement and from these figures it can be observed that:



	2sw	4sw	8sw	16sw	32sw	64sw	128sw
ONOS	0.154	0.165	0.248	0.371	0.617	1.12	
OpenDaylight	0.601	0.515	0.505	0.671	0.815	1.128	2.201
POX	0.071	0.086	0.109	0.152	0.256	0.461	0.894
Ryu	0.115	0.133	0.229	0.349	0.573	1.088	2.04

Fig.4. Average RTT of first test



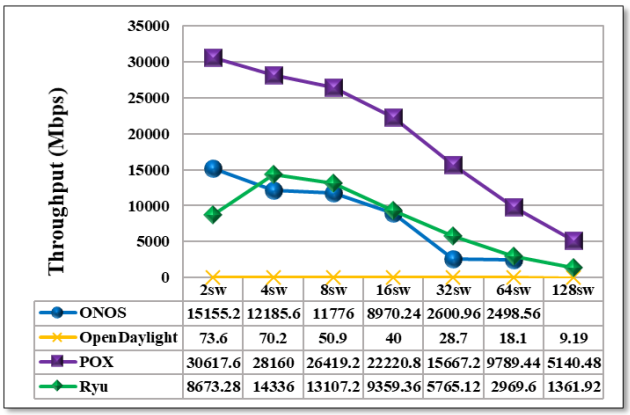
	2sw	4sw	8sw	16sw	32sw	64sw	128sw
ONOS	1.267	1.836	2.831	5.279	9.446	12.24	
OpenDaylight	8.921	3.561	3.624	12.026	13.89	34.738	40.261
POX	0.348	0.497	0.796	1.371	2.624	5.028	10.074
Ryu	1.233	1.989	2.991	4.902	9.809	19.27	29.509

Fig.5. Average RTT of first test with different packer size and time interval

- In general, the results of average RTT with different packer size and time interval shown in Fig. 5 is higher than the results of average RTT with default parameters shown in Fig. 4 specially when the number of switches increase.
- increasing the number of switches increases the delay due to increasing the load (number of switches) on the controllers and more processes are needed.
- OpenDaylight has the highest RTT values.
- POX has the lowest RTT values.
- ONOS and Ryu approximately have the same RTT delay.

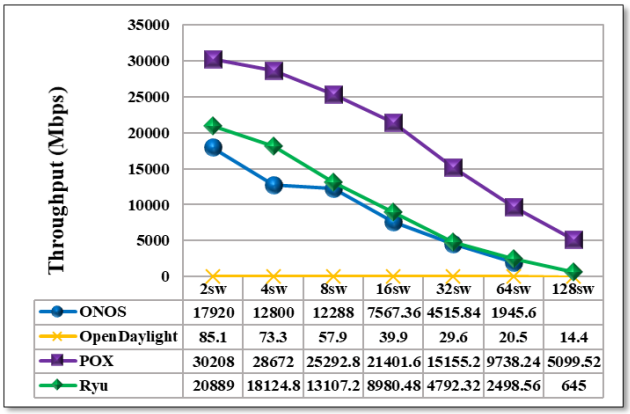
Fig. 6 and Fig. 7, present the results of Throughput measurement and from these figures it can be observed that:

- In general, for each controller the results of Throughput in both figures are nearly the same.
- increasing the number of switches decreases the Throughput due to increasing the load on the controllers.
- OpenDaylight has the lowest Throughput values.
- POX has the highest Throughput values.
- Except for (2) switches case in Fig. 6, ONOS throughput values is lower than Ryu.



	2sw	4sw	8sw	16sw	32sw	64sw	128sw
ONOS	15155.2	12185.6	11776	8970.24	2600.96	2498.56	
OpenDaylight	73.6	70.2	50.9	40	28.7	18.1	9.19
POX	30617.6	28160	26419.2	22220.8	15667.2	9789.44	5140.48
Ryu	8673.28	14336	13107.2	9359.36	5765.12	2969.6	1361.92

Fig.6. Throughput of first test



	2sw	4sw	8sw	16sw	32sw	64sw	128sw
ONOS	17920	12800	12288	7567.36	4515.84	1945.6	
OpenDaylight	85.1	73.3	57.9	39.9	29.6	20.5	14.4
POX	30208	28672	25292.8	21401.6	15155.2	9738.24	5099.52
Ryu	20889	18124.8	13107.2	8980.48	4792.32	2498.56	645

Fig.7. Throughput of first test with different TCP window size

Fig. 8 and Fig. 9, present the results of jitter measurement and from these figures it can be observed that:

- In general, for each controller except for OpenDaylight, the results of jitter in both figures are nearly the same and increasing the number of switches does not affect the jitter.
- OpenDaylight has the highest jitter values.
- POX has the lowest jitter values.
- ONOS, POX, and Ryu controllers keep close jitter values with some differences.

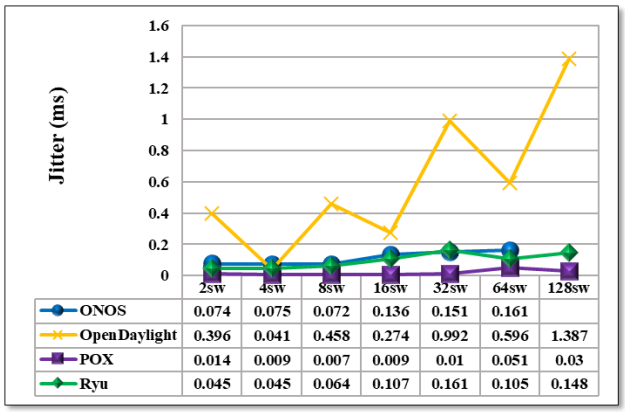


Fig.8. Jitter of first test

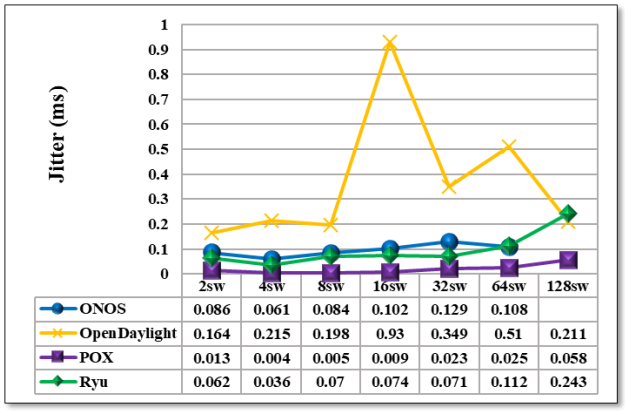


Fig.9. Jitter of first test with different UDP buffer size

2) Results of the Second Test

The results of measuring the average RTT, Throughput, and Jitter when connecting (1, 2, and 4) ONOS controller and (1, 2, 4, 8, and 16) controllers of each of (POX, and Ryu) to linear topology with fixed 64 switches will be presented for comparing and evaluating the performance of these controllers to show the effect of increasing the number of controllers on the network performance.

Fig. 10 and Fig. 11, present the results of average RTT measurement and from these figures it can be observed that:

- In general, the results of average RTT with different packer size and time interval shown in Fig. 11 is higher than the results of average RTT with default parameters shown in Fig. 10.
- increasing the number of controllers does not affect the average RTT of POX and Ryu controllers.
- Except for the case of one controller, POX has the lowest RTT values, and except for the case of two controller in Fig. 10, Ryu has the highest RTT values.

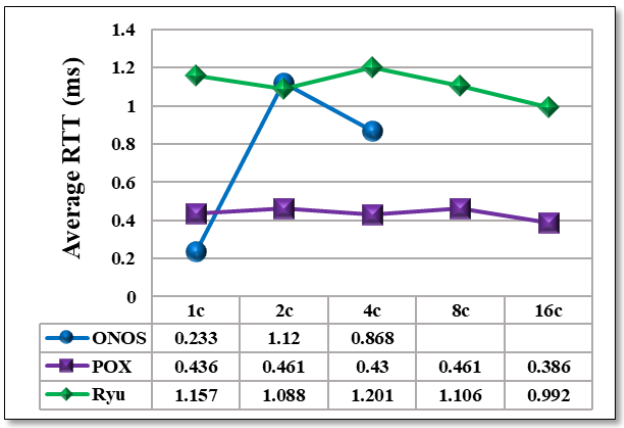


Fig.10. Average RTT of Second Test

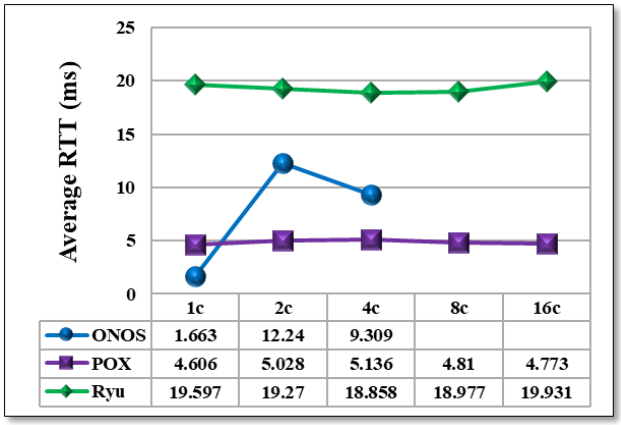
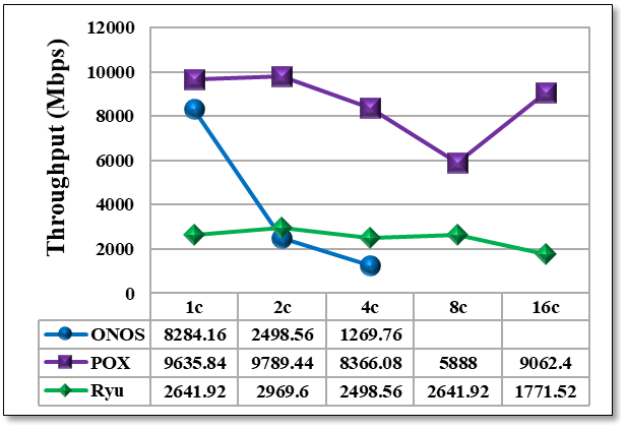


Fig.11. Average RTT of Second Test with different packer size and time interval

Fig. 12 and Fig. 13, present the results of Throughput measurement and from these figures it can be observed that:

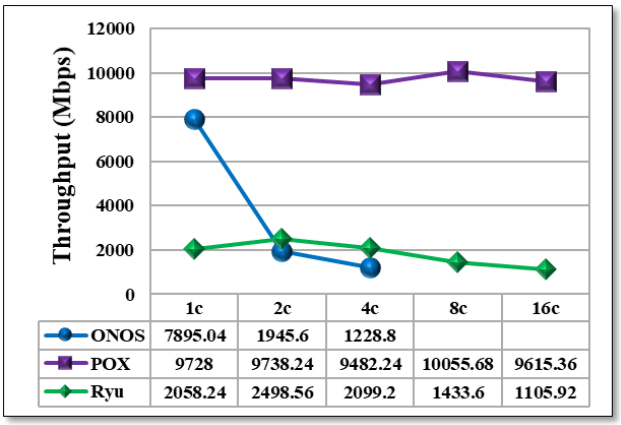
- In general, for each controller the results of Throughput in both figures are nearly the same.
- POX has higher throughput values than ONOS and Ryu.
- ONOS has lower throughput values than Ryu in case of two and four controllers connected.

Copyright © 2019 MECS



	1c	2c	4c	8c	16c
ONOS	8284.16	2498.56	1269.76		
POX	9635.84	9789.44	8366.08	5888	9062.4
Ryu	2641.92	2969.6	2498.56	2641.92	1771.52

Fig.12. Throughput of Second Test

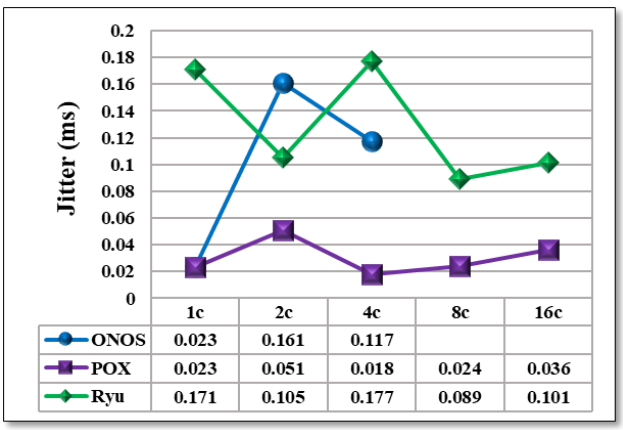


	1c	2c	4c	8c	16c
ONOS	7895.04	1945.6	1228.8		
POX	9728	9738.24	9482.24	10055.68	9615.36
Ryu	2058.24	2498.56	2099.2	1433.6	1105.92

Fig.13. Throughput of Second Test with different TCP window size

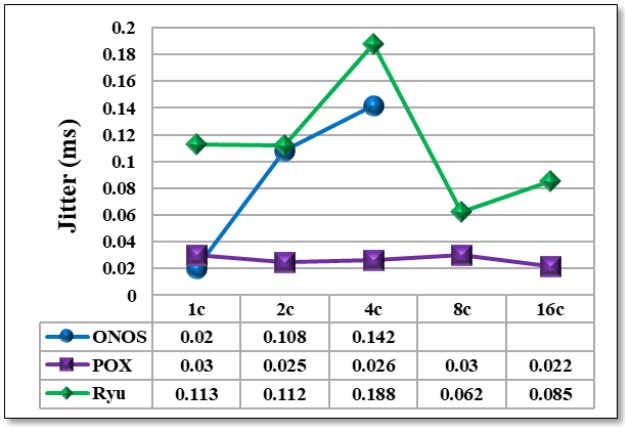
Fig. 14 and Fig. 15, present the results of jitter measurement and from these figures it can be observed that:

- In general, for each controller the results of jitter in both figures are nearly the same.
- OpenDaylight has the highest jitter values.
- POX has the lowest jitter values in both figures and Ryu has the highest jitter values.
- ONOS jitter value in case one controller connected is lower than 2 and 4 controllers.



	1c	2c	4c	8c	16c
ONOS	0.023	0.161	0.117		
POX	0.023	0.051	0.018	0.024	0.036
Ryu	0.171	0.105	0.177	0.089	0.101

Fig.14. Jitter of Second Test



	1c	2c	4c	8c	16c
ONOS	0.02	0.108	0.142		
POX	0.03	0.025	0.026	0.03	0.022
Ryu	0.113	0.112	0.188	0.062	0.085

Fig.15. Jitter of Second Test with different UDP buffer size

V. Conclusion

A performance evaluation and comparison of ONOS, OpenDaylight, POX, and Ryu controllers was carried out in two tests that include connecting linear topology to each controller with different number of switches, and different number of controllers.

In first test, connecting two controllers to the network shows that among controllers, POX gives better (average RTT, throughput, and jitter) results than other controllers and OpenDaylight gives worst results than other controllers. It is also concluded that increasing the number of switches increases the RTT delay and jitter and decreases the throughput values because more processing will be needed. In second test, connecting more controllers to the network shows that among controllers, POX has better results in most cases than ONOS and Ryu. It also shows that within each controller, POX and Ryu controllers keep nearly constant results and ONOS results of (average RTT, throughput, and jitter) of one controller is better than two and four controllers.

The two tests results show that repeating (ping, Iperf TCP connection, and Iperf UDP connection) commands with different (packer size and time interval, TCP window size, and UDP buffer size) parameters degrades the performance of the controllers due to load increasing. Finally, from the results of these tests, it can be concluded that POX controller shows better results in having constant low average RTT, high throughput, and low jitter values, in addition of having more durability and flexibility.

References

[1] J. Doherty, SDN and NFV Simplified Visual Guide to Understanding Software Defined Networks and Network Function Virtualization, USA: Pearson Education, Inc., 2016.

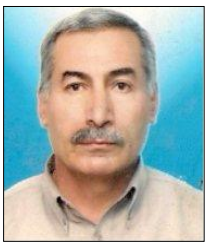
[2] D. K. Sharma, *Training Report on Software Defined Networking*, Florida : Florida International University, 2015.

[3] M. Liyanage, A. Gurtov and M. Ylianttila, Software Defined Mobile Networks (SDMN), UK: John Wiley & Sons, Ltd, 2015.

[4] W. Stallings, Foundations of Modern Networking SDN, NFV, QoE, IoT, and Cloud, USA: Pearson Education, Inc., 2016.
[5] S. Azodolmolky, Software Defined Networking with OpenFlow, Birmingham, UK: Packt Publishing Ltd., 2013.
[6] F. Hu, Network Innovation through OpenFlow and SDN Principles and Design, NW, USA: Taylor & Francis Group, LLC, 2014.
[7] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, R. Smeliansky, ”Advanced study of SDN/OpenFlow controllers”, 9th Central & Eastern European Software Engineering Conference, Russia, 2013.
[8] Y. Zhao, L. Iannone and M. Riguidel, "On the Performance of SDN Controllers: A Reality Check," in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, San Francisco, CA, USA, 2015.
[9] D. Turull, M. Hidell and P. Sjödin, "Performance evaluation of OpenFlow controllers for network virtualization," in *IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*, Vancouver, BC, Canada, 2014.
[10] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu and E. C. Popovici, "A Comparison between Several Software Defined Networking Controllers," in *International Conference on Advanced Technologies, Systems and Services in Telecommunications - TELSIKS*, Niš, Serbia, 2015.
[11] S. Rowshanrad, V. Abdi and M. Keshtgari, "Performance Evaluation of SDN Controllers: Floodlight and OpenDaylight," *IIUM Engineering Journal*, vol. 17, pp. 47-57, 2016.
[12] O. Salman, I. Elhajj, A. Kayssi and A. Chehab, "SDN Controllers: A Comparative Study," in *18th IEEE Mediterranean Electrotechnical Conference – MELECON*, Limassol, Cyprus, 2016.
[13] A. D. Jasim and D. A. Hamid, "Enhancing the Performance of OpenFlow Network by Using QoS," *International Journal of Scientific & Engineering Research (IJSER)*, vol. 7, no. 5, pp. 950-955, 2016.
[14] P. Göransson, C. Black and T. Culver, Software Defined Networks A Comprehensive Approach, 2nd ed., MA, USA: Elsevier Inc., 2017.
[15] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking, HotSDN '14*, Chicago, IL, USA, 2014.
[16] S. N. A. Braojos, "The OpenDaylight Open Source Project," King Juan Carlos University, Móstoles, Madrid, Spain, 2014.
[17] *OpenDaylight Documentation Documentation Release Boron*, OpenDaylight Project, 2018.
[18] *ryu Documentation Release 4.28*, ryu development team, 2018.
[19] K. Kaur, J. Singh and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform," in *International Conference on Communication, Computing & Systems (ICCCS)*, USA, 2014.
[20] M. B. Al-Somaidai and E. B. Yahya, "Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation," *American Journal of Software Engineering and Applications*, vol. 3, no. 6, pp. 74-82, 2014.
[21] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and L. R. Prete, "Using Mininet for Emulation and Prototyping Software-Defined Networks," in *IEEE Colombian Conference on Communications and Computing (COLCOM)*, Bogota, Colombia, 2014.
[22] "Ping," 2011, [Online]. Available: http://openmaniak.com/ping.php.
[23] "Bandwidth, Throughput and Delay," [Online]. Available: http://networking.layer-x.com/p040300-1.html.
[24] "Iperf," 2010. [Online]. Available: http://openmaniak.com/iperf.php.

Authors’ Profiles

Mahmood Zaki Abdullah is an associate professor Dr. in the Computer Engineering Department at the College of Engineering of Al-Mustansiriyah University. He got the Ph.D. and M.Sc. degrees from the University of Technology at 2007, and 2000 and a B.Sc. degree from the University of Baghdad at 1991. His research interests include Information Technology, Software Engineering, and Computer Networks. He has served as a Technical Program Committee member for many international conferences; he published many books and papers in these fields.



Nasir Ahmed Al-awad was born in Iraq, 1957. He received B.Sc. degree in control and system engineering from Technological University, Iraq, in 1981. M.Sc. degree in control and instrumentation engineering from Technological University, Iraq, in 1984. He is currently Assist Prof. and work at Computer Engineering Department, Al-Mustansiriyah University, Iraq. His research interests include control theory, computer control and computer aided design of control system.

Fatima W. Hussein is a M.Sc. Student at Computer Engineering Department, AL-Mustansiriyah University, 2017. B. Sc. Degree from Computer Engineering Department, AL-Mustansiriyah University, 2016.

How to cite this paper: Mahmood Z. Abdullah, Nasir A. Al-awad, Fatima W. Hussein, " Evaluating and Comparing the Performance of Using Multiple Controllers in Software Defined Networks", International Journal of Modern Education and Computer Science(IJMECS), Vol.11, No.8, pp. 27-34, 2019.DOI: 10.5815/ijmecs.2019.08.03

Copyright © 2019 MECS