International Journal of Computer Networks and Communications Security

VOL. 5, NO. 11, NOVEMBER 2017, 239–247
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

A Systematic and Generic Method for Choosing A SDN Controller

Omayma BELKADI[1] and Yassin LAAZIZ[2]

[1, 2] LabTIC, National School of Applied Sciences in Tangier, Department of Communication systems and Computer science, Morocco

[1]belkadi.omayma@gmail.com, [2]ylaaziz@uae.ma

ABSTRACT

Controllers are fundamental elements for the implementation of a Software-Defined Network (SDN) application. The present work focusses on the problem of choosing the appropriate controller, for a specific use case, among the multiple solutions proposed by academia and industry. We here present a new method for controllers' benchmarking, which allows finding the right controller for each use case in a systematic and rational manner. The Method is based on two steps: the first consists on classifying the controllers using a Multi-Criteria Decision Making method according to the relevance of their features for the use case. This step enables to limit considerably the number of potential controllers for the SDN application. In the second step, a performance evaluation is carried out using the 'CBENCH' tool, where only the first three ranked controllers at the first step are considered. The method has been tested in the case of user traffic classification over SDN, where ten most known controllers were considered. The results showed that 'OpenDaylight' (ODL) is the controller that responds the better to the constraints and requirements of this specific application. The method is generic and can be applied regardless of the SDN application.

Keywords: *Software Defined Networks, Multi-Criteria Decision Making, Analytic Hierarchy Process, CBENCH, SDN Controllers.*

1 INTRODUCTION

Service providers want their networks to be agile, efficient and meet the growing demand for bandwidth, in addition to yield innovative services and new business models. But, it is certain nowadays, that this goal cannot be achieved by means of traditional networking, for many reasons that we detailed elsewhere [1]. Recently, a new networking concept, named Software-Defined Networking (SDN) emerged to fix this issue; and according to the community of network scientists [2], it is the solution for new generation IT networks. Its strength lies in the fact of separating the control plane from the data plane, allowing programmability of the network. Indeed, in SDN, a network administrator can manage the network through the central control console without going by each switch manually. This is done by means of a controller, which is a strategic entity situated between network devices and applications, that acts as a 'brain' of the network. The controller is therefore considered as the key element for the implementation of any SDN-based application.

Nevertheless, due to the lack of standardization for SDN Networks [3], many controllers were proposed by academia and industry. These controllers are built on different technologies, have varied features and most of them undergo periodic enhancements and upgrading.

Given this diversity of controllers, the following question arises for any developer of SDN applications: What is the appropriate controller for a given SDN application? The answer to this issue is of great importance, especially for novice users of this concept who should dispose, in principle, of a systematic approach to make their choice of controller. Nevertheless, according to a selected bibliography of recent publications [4]–[13], there is no work proposing a global and generic solution to this issue. So, here we are suggesting a new approach which enables definitely to find the right controller for each use case.

In what follows, we begin by a bibliographic survey on the controllers' benchmarking methods (section 2). In section 3 we give in details our methodology for the feature-based selection and the performance tests that we applied to the specific case of user traffic classification. Section 4 present and

discuss the performance tests results we obtained with a limited number of controllers issued from the feature's comparison. And finally, we give a conclusion in section 5.

2 BIBLIOGRAPHIC SURVEY

Most articles on controller benchmarking for SDN have proceeded, by either analyzing their features or carrying out performance tests. The first tests were reported in [4]–[7], and they focused only on performance assessment - mainly testing the controllers' throughput and latency - which concerned only the few existing controllers back then. But, with the continuous release of new controllers and their subsequent versions, other researchers expanded the range of performance tests to cover mostly all available controllers [8]–[11]. However, in reality, choosing a controller is not only a matter of performance; for example, the existence of a specific feature for a given controller can make it more suitable than another powerful one, in a given use case. In addition, the performance tests are time-consuming since the test beds are not usually simple to implement, especially if the number of studied controllers is high.

Other researchers [12] preferred a different approach based on the controllers' features benchmarking. For this purpose, they used a multi criteria decision-making method (MCDM) [14]. Nevertheless, a selection of controller, even if performed using this rational and systematic procedure, is still insufficient to provide a definitive choice of controller, as performance tests are still needed to be sure that the chosen controller is powerful enough for the needs of the considered SDN application.

Hence, logically, an association of the two above approaches is a promising solution for an efficient choice of controller. Indeed, an upstream feature-based classification can help eliminate solutions having 'weak' features, before going ahead with performance tests, and this allows a considerable time saving. Nevertheless, to our knowledge, the idea of associating a feature-based classification and a performance evaluation for the choice of an SDN controller has not been proposed yet; at least, in the way we propose it in this paper. Only authors of [13] considered collecting controllers' features information, but their approach focused mainly on performance tests they executed on all these controllers, without exploiting the features' study results to optimize the problem.

So, in this work, we propose a complete process for benchmarking SDN controllers which can be applied regardless of the use case. The process is composed of two steps as illustrated in Figure 1. The first consists on classifying the controllers by prioritizing the most important features for the use case, and applying the Analytic Hierarchy Process (AHP) technique, which is a MCDM method. One can note that AHP results' change depending on the use case and this guarantees the conditions of a generic process. In the second step, we take only the top three ranked controllers in the previous step for performance evaluations using 'CBENCH' tool.

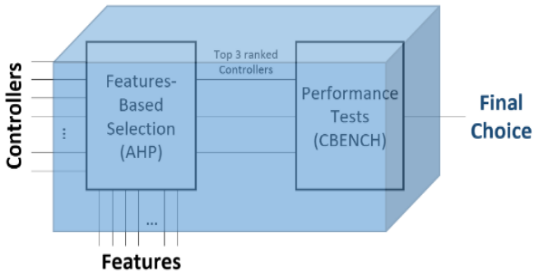

Fig. 1. The proposed method schema

3 METHODOLOGY/EXPERIMENTAL SETUP

3.1 Studied Use-Case, Controllers and Features

Our research deals with the problem of traffic classification as in [15], [16], but within SDNs. We are therefore looking for the most suitable controller for this use case. In addition, we have restricted our study to open-source controllers for obvious reasons.

Below, we give a list of the most known open-source controllers, their origin and some of their characteristics:

- ***NOX***: C-based and developed by Nicira, it is the first OpenFlow controller.
- ***Beacon***: is a Java-based OpenFlow controller, cross-platform, modular, that supports event-based and thread operations.
- ***Trema***: is a Ruby and C OpenFlow controller that is easy to use.
- ***POX***: written in Python, it is developed in Stanford and used mainly for research and education.
- ***OpenMUL***: its core is a C-based multithreaded infrastructure; it is an OpenFlow SDN controller platform.
- ***Ryu***: written in Python, it is a SDN controller with a well-defined API, which is designed to

help programmers create a new network management and control applications. It supports standard protocols, including OpenFlow, Netconf and OF-config.

- ***Flooflight***: written in Java, it is a Big Switch Networks' release using OpenFlow as a protocol and Apache.
- ***OpenDaylight***: written in Java, it is a Linux foundation cooperation project, it is developed to promote innovation and implementation of SDN.
- ***OpenContrail***: Sponsored by Juniper, it has an open architecture and includes a logically centralized view, virtual routing, analysis engine, etc.
- ***ONOS***: Written in Java, and mainly designed for carrier networks by the Open Networking Lab (ON. Lab). ONOS project aims to provide high speed and large-scale networks.

In Table 1, we have listed, for these different controllers, the most important features to consider in a benchmarking study. This table was filled collecting the latest information from various sources, like controllers' documentation and reviews [7], [17]–[26].

Table 1: Controller's features

	Open source	**First release**	**Language support**	**Platform support**	**Activity**	**Rest API**	**Documentation**	**GUI**	**Clustered Deployment**	**Open Flow supported version**	**OpenStack networking**
NOX	Yes	2008	C/C++	Linux	Low	No	Low	No	Yes	1.0	No
Beacon	Yes	2010	Java	Linux, Mac, Win	Low	No	Medium	Yes	Yes	1.0	No
Trema	Yes	2011	C/C++, Ruby	Linux	Medium	No	Medium	No	Yes	1.0	Yes
POX	Yes	2012	Python	Linux, Mac, Win	Low	No	Low	Yes	No	1.0	No
OpenMUL	Yes	2012	C/C++	Linux	Low	No	High	Yes	Yes	1.0, 1.3, 1.4	Yes
RYU	Yes	2012	Python	Linux	Medium	No	Medium	Yes	Yes	1.0, 1.2, 1.3, 1.4, 1.5	Yes
Floodlight	Yes	2012	Java	Linux, Mac, Win	Medium	Yes	High	Yes	Yes	1.0	Yes
OpenDaylight	Yes	2013	Java	Linux, Mac, Win	High	Yes	High	Yes	Yes	1.0, 1.2, 1.3	Yes

OpenContrail	Yes	2013	C/C++, Java, Python	Linux	High	Yes	High	Yes	Yes	Not supported	Yes
ONOS	Yes	2014	Java	Linux, Mac	High	Yes	Medium	Yes	Yes	1.0, 1.2, 1.3	Yes

Step 1: Feature-based selection

The process we consider here is based on the constraints of the case of user traffic classification, in which we search for a controller that responds to SDN standards, like, being open-source, supporting OpenFlow, supporting Openstack, etc.

As SDN controllers have different properties, so, the choice of a controller must undergo a multi-criteria decision making process, depending on what features have priority for a specific application. Many methods are available for this aim [27], here we use the Analytic Hierarchy Process (AHP) which is perfectly adapted to our analysis. The AHP method is laborious, but its execution becomes simple by using an online free calculator [28].

To apply AHP on the set of features in Table 1, we begin by building the comparison matrix of features, by assigning to a given feature a level in a scale from 1 to 9 as follow:

1: Equally important
2: Quietly equally important
3: Moderately important
5: Strongly important
7: Very strongly important
9: Extremely important

Briefly, the user has an interface to provide manually the ‘Alternatives’ and the ‘Criteria’, which are respectively, the names of the given controllers and their features. Then, the prioritization conditions are fixed to the desired levels we mentioned before (from 1 to 9). The final results are presented in form of tables (generated comparison matrixes) and graphics (ranking the features and controllers).

For our case study, ‘Open source’, ‘OpenFlow support’, and ‘Openstack support’ are the features that we favor, followed by ‘Activity’, which means the controller is still being developed and has an active community. Then, comes the ‘Documentation’ and disponibility of a ‘GUI’ (Graphical User Interface), which we find more important than ‘REST API’ and ‘Clustered deployment’, which is the situation where multiple controllers are used. Furthermore, we consider the ‘First release’ of a controller the least important feature after ‘Platform support’ and ‘Language support’.

Table 2 presents the pairwise prioritization of controllers' features according to the rules above. In the first line, for example, we ranked the 'Open source' feature ‘Extremely important’ (level 9) compared to the 'First release' one, and ‘Very strongly important’ (level 7) compared to 'Platform' and 'Language support'. It is found ‘Moderately important’ (level 3) compared to 'Activity', and ‘Strongly important’ (level 5) when compared to 'REST API', 'Documentation', 'GUI', and 'Clustered deployment'. Finally, the 'Open source' feature is considered ‘Equally important’ (level 1) in comparison to 'OpenFlow' and 'Openstack' support features. Values for the other lines/columns follow the same logic.

Table 2: Comparison matrix of features

Criteria preferences	Open source	First release	Language support	Platform support	Activity	REST API	Documentation	GUI	Clustered deployment	OpenFlow support	Openstack support
Open source	1	9	7	7	3	5	5	5	5	1	1
First release	1/9	1	1/3	1/3	1/9	1/7	1/7	1/7	1/7	1/9	1/9
Language support	1/7	3	1	1	1/7	1/5	1/5	1/5	1/5	1/7	1/7
Platform support	1/7	3	1	1	1/7	1/5	1/5	1/5	1/5	1/7	1/7
Activity	1/3	9	7	7	1	5	5	5	5	1/5	1/5
REST API	1/5	7	5	5	1/5	1	1/2	1/2	1	1/5	1/5
Documentation	1/5	7	5	5	1/5	2	1	1	2	1/5	1/5

GUI	1/5	7	5	5	1/5	2	1	1	2	1/5	1/5
Clustered deployment	1/5	7	5	5	1/5	1	1/2	1/2	1	1	1
OpenFlow support	1	9	7	7	3	5	5	5	5	1	1
Openstack support	1	9	7	7	3	5	5	5	5	1	1

Figure 2 shows the ranking of the controllers' features, generated by the online AHP tool, as a result treatment of the comparison matrix in Table 2.

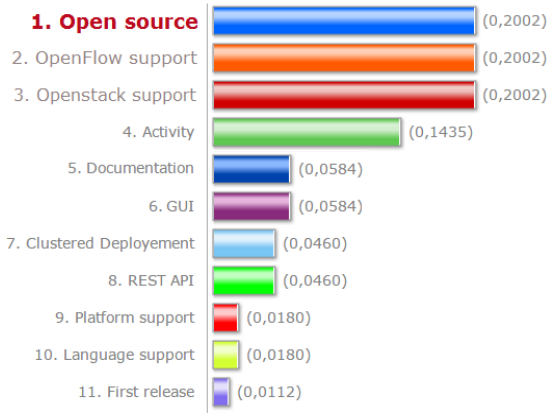

Fig. 2. Features ranking in our use case

The next step in applying the AHP consists in filling other matrixes, comparing each controller to another with their features' ranking, using the information we collected in Table 1. The complete description is not reported here with all the details, but is available as a tutorial in [29].

Step 2: Performance tests

At the end of the first step, we take the top three controllers we get, to conduct performance tests. These tests measure a controller Latency, which can be defined as the amount of response that can be given by the controller per second (responses/sec), and its Throughput, which is the number of streams that can be, handled (flows/sec). To do this, we used a tool named 'CBENCH', which is a collection of open-source programs with multiple datasets assembled by the community to allow comparison and architecture optimization of SDN controllers.

CBENCH was installed on an Ubuntu machine, and run successively against each one of the three selected controllers, that were implemented in another Ubuntu machine acting as server. We chose to execute CBENCH on a different machine to minimize the interference. The characteristics of the two machines are as follows:

Ubuntu (14:04) as the local machine, with a 1.8 GHz Quad Core processor (Core i5), 8 GB of RAM and 100 GB hard-drive

Ubuntu (12:01) as the Server, having 16 threads, 16 GB of RAM and 100 GB hard-drive.

To associate a specific number of threads to a controller, we used taskset line command. We run a set of tests for each controller in both the Throughput and Latency mode.

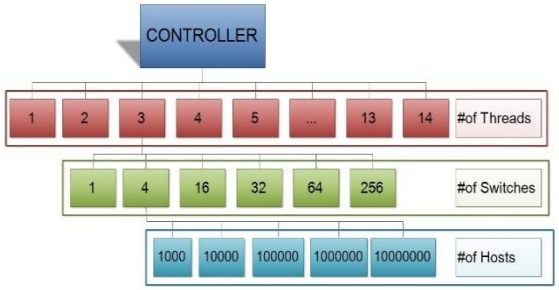

Fig. 3. Throughput test loop

For the Throughput test, we increase the number of threads each time then we run a number of switches (whether 1, 4, 16, 32, 64, or 256) and for each number of switches we keep increasing the number of hosts as shown in Figure 3. On the other hand, for Latency, we work with all the threads without using taskset, and focus more in increasing the number of switches and hosts, as presented in Figure 4, to see how the controller is reacting and how much time it takes to respond.

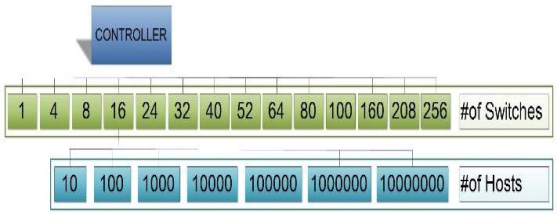

Fig. 4. Latency test loop

For each test, we launched three loops but we retained only the last loop's result, since the first two loops can be considered a warm-up for the controller. The complete procedure of the tests can be found in [29].

4 RESULTS

4.1 Feature-based selection

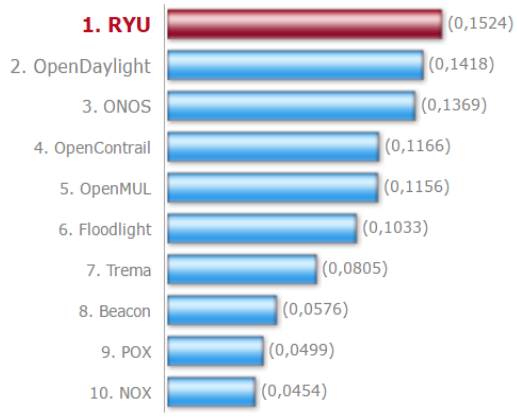

Fig. 5. Results of the Feature-based selection

The results of the feature-based selection (Figure 5) show three top-ranked controllers which are: RYU, OpenDaylight (ODL), and ONOS.

RYU was ranked first, indeed, in addition to having all the requirements of a standard SDN controller; it is supporting OpenFlow in all its versions, which was an extremely important criterion for our case study. For ODL and ONOS the notation is slightly lower than RYU; the small advantage of ONOS compared to ODL, comes from the fact that the latter has a richer documentation and is supporting three platforms, versus two platforms only for ONOS.

4.2 Performance tests

In the following we show and discuss the results of tests of Throughput and Latency, we conducted on the three top-ranked controllers: ODL, ONOS and RYU.

The tests are usually conducted with the default values of CBENCH, which are 100000 hosts, 16 switches on a machine with 4 threads [30]. Since we cannot present all the combinations of our tests and their results, we will use these common values in the following figures.

4.2.1. Throughput tests

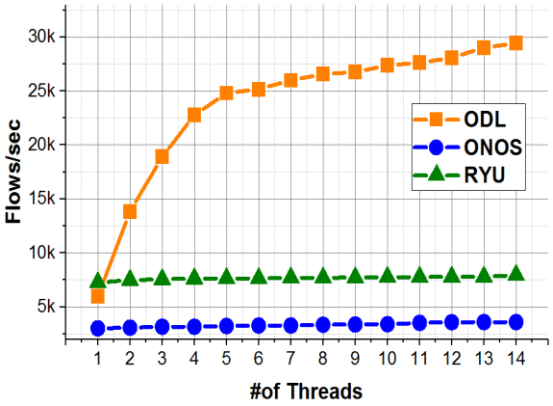

Fig. 6. Maximum throughput as function of number of threads

Figure 6 shows the throughput results as function of the number of threads for the three controllers under study. Even though the three controllers are multithreaded, we can see that ONOS and RYU throughputs show a very small variation with the increased number of threads, contrarily to ODL, for which the number of treated flows increases exponentially. Furthermore, ODL could achieve up to 30000 flows/second while RYU did not even achieve 8000, and ONOS has been limited to less than 4000 flows. These results show that ODL is 4-8 times more performant than RYU and ONOS respectively.

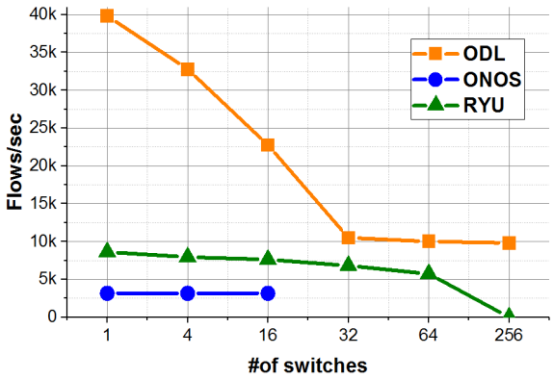

Fig. 7. Maximum throughput as function of number of switches

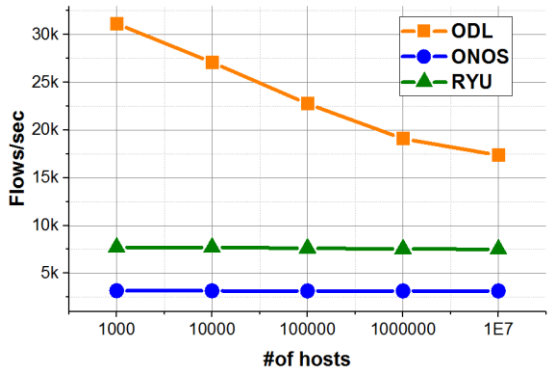

Fig. 8. Maximum throughput as function of number of hosts

Figures 7-8 show results of throughput versus number of switches and hosts respectively for the three controllers under study. Both Figures show a decreasing performance with higher number of switches or hosts. Moreover, in Figure 7, we find that the performances of RYU went down to zero with 256 switches, while with ONOS we could not have results above 16 switches.

4.2.2 Latency tests

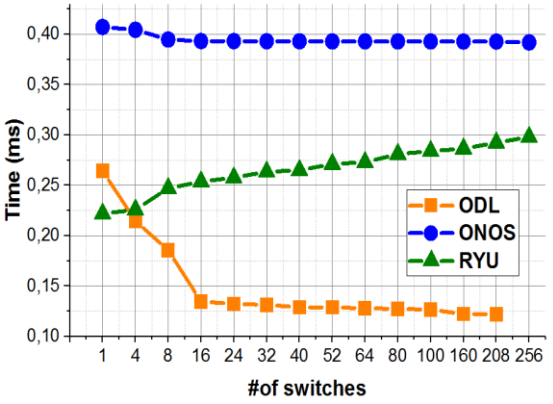

Fig. 9. Average latency as function of number of switches

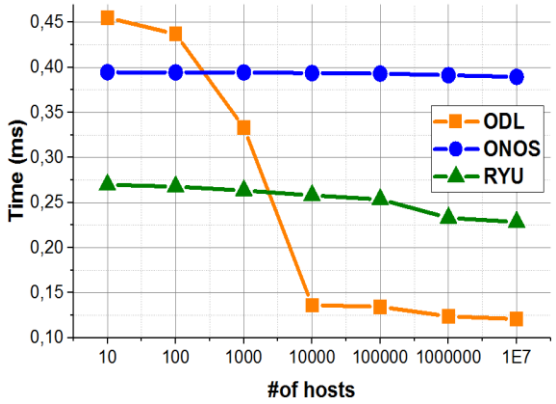

Fig. 10. Average latency as function of number of hosts

Figures 9-10 represent the average latency versus the number of switches and hosts for the three controllers. We can see that ODL average latency is somewhat high for low number of hosts, compared to the values for the other two controllers, but it decreases significantly with increasing number of hosts and switches. Despite the high latency at first, ODL proves to have the minimum latency compared to RYU and ONOS.

5 DISCUSSION

The results of the proposed two-step method show without ambiguity that ODL is the most suitable solution that responds the better to the constraints and requirements of our use case. Indeed, ODL is a well-featured controller presenting a high modularity and capability of supporting a wide range of applications. Moreover, it is supported by the most known communities, in the aim of leading the transformation to the networks of the future.

However, from a practical point of view, we must shed light on some difficulties in testing ODL and ONOS performance. We found that the tests conducted by ODL and ONOS communities using CBENCH in a more powerful testbed [31], [32] show that both controllers can achieve up to 1 million flows per second, which means that the hardware configuration affects directly and decreases these controllers' performance. In fact, ONOS throughput went dramatically from 1 million to less than 3000 flows/sec in our tests, while we could only achieve 87000 flows/sec with ODL.

In addition, both controllers are Java based and suffer from inherent problems and bugs of the language. For instance, in both flow and latency modes, CBENCH failed sometimes to provide results, by showing a "disconnected controller" message, or by displaying only zeros in the output, or even by exiting the CBENCH loop without finishing the calculations. Moreover, ONOS was observed to have additional issues when increasing the number of switches and/or hosts, we therefore had to install / configure the controller several times from scratch, to continue the tests and get results. Thus, that was the reason why we decided to conduct manually the experiments on these two controllers, and we think that more efforts must be done to resolve these controllers' bugs and errors to allow automation of tests. Therefore, it must be mentioned that the scripts we share in [29] were mainly obtained with RYU, the Python-based controller, whose installation, configuration and running went with no errors nor issues.

Nevertheless, RYU showed less performance due to a less active community in the recent years [33] contrarily to ONOS and ODL.

5 CONCLUSION

We have proposed a new procedure for benchmarking SDN controllers, associating features based selection and performance assessments. The method comes in two-steps: the first consists of carrying out a feature-based comparison using the AHP method, which is a systematic and generic process. Hence, the controllers are classified in rational way according to the needs and constraints of the use case. Then, in the second step, we achieve a performance test using 'CBENCH' tool, but only for the best-ranked controllers according to the results of the first step.

The method was tested in the case of user traffic classification over SDN, where the ten most known controllers where considered, and up-to-date information concerning these controllers were used. The results confirmed without ambiguity that ODL is the controller that responds the best to the constraints and requirements of this use case. Further outputs of the benchmarking process can be found in a GitHub repository [29].

Finally, we have reported, in all fidelity, some difficulties we have encountered during the performance tests of ONOS and ODL controllers.

6 ACKNOWLEDGEMENT

We thank the PMECloud Company in its person BELKADI Rafik for providing us the server where we conducted our experiments and for the technical support.

7 REFERENCES

[1] B. Omayma and L. Yassin, “Software Defined Networks (SDN) : the new Era of Networking,” 2015.

[2] H. Kim and N. Feamster, “Improving network management with software defined networking,” IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, 2013.

[3] “Joint Coordination Activity on Software-Defined Networking (JCA-SDN).” [Online]. Available: http://www.itu.int/en/ITU-T/jca/sdn/Documents/deliverable/Free-download-sdn-roadmap.docx.

[4] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, “On controller performance in software-defined networks,” Proceeding Hot-ICE’12 Proc. 2nd USENIX Conf. Hot Top. Manag. Internet, Cloud, Enterp. Networks Serv., pp. 10–10, 2012.

[5] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, “A flexible OpenFlow-controller benchmark,” in 2012 European Workshop on Software Defined Networking, 2012, pp. 48–53.

[6] M. Fernandez, “Evaluating OpenFlow Controller Paradigms,” ICN 2013, Twelfth Int. Conf. …, no. c, pp. 151–157, 2013.

[7] D. Erickson, “The beacon openflow controller,” Proc. Second ACM SIGCOMM Work. …, pp. 13–18, 2013.

[8] A. Shalimov, D. Zimarina, and V. Pashkov, “Advanced Study of SDN / OpenFlow controllers,” Proceeding CEE-SECR ’13 Proc. 9th Cent. East. Eur. Softw. Eng. Conf. Russ., 2013.

[9] Y. Zhao, L. Iannone, and M. Riguidel, “On the performance of SDN controllers: A reality check,” 2015 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Network, NFV-SDN 2015, pp. 79–85, 2016.

[10] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, “A comparison between several Software Defined Networking controllers,” 2015 12th Int. Conf. Telecommun. Mod. Satell. Cable Broadcast. Serv., pp. 223–226, 2015.

[11] Z. K. Khattak, M. Awais, and A. Iqbal, “Performance evaluation of OpenDaylight SDN controller,” Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, vol. 2015–April, pp. 671–676, 2014.

[12] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, “Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers,” no. JANUARY 2014, 2015.

[13] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, “SDN controllers: A comparative study,” Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016, no. 978, pp. 18–20, 2016.

[14] T. L. Saaty, “What is the Analytic Hierarchy process?,” in Mathematical models for decision support, Springer, 1988, pp. 109–121.

[15] A. Vlăduţu, D. Comăneci, and C. Dobre, “Internet traffic classification based on flows’ statistical properties with machine learning,” Int. J. Netw. Manag., 2016.

[16] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapé, “Exploiting packet-sampling measurements for traffic characterization and classification,” Int. J. Netw. Manag., vol. 22, no. 6, pp. 451–476, Nov. 2012.

[17] D. Kreutz, F. M. V Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proc. IEEE, vol. 103, no. 1, pp. 14–76, 2015.
[18] N. Gude et al., "NOX: towards an operating system for networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, 2008.
[19] M. Mendon et al., "A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks To cite this version : A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks," 2014.
[20] "Trema controller." [Online]. Available: http://trema.github.io/trema/.
[21] "OpenMUL controller." [Online]. Available: http://www.openmul.org/.
[22] "Floodlight controller." [Online]. Available: http://www.projectfloodlight.org/.
[23] "OpenContrail controller." [Online]. Available: http://www.opencontrail.org/.
[24] "The OpenDaylight controller." [Online]. Available: https://www.opendaylight.org/.
[25] "ONOS controller." [Online]. Available: http://onosproject.org/.
[26] N. Communications, "RYU." [Online]. Available: https://osrg.github.io/ryu/.
[27] E. Triantaphyllou, Multi-criteria Decision Making Methods: A Comparative Study, vol. 44. Springer Science & Business Media, 2013.
[28] "AHP online Calculator." [Online]. Available: http://www.123ahp.com/izracun.aspx.
[29] "Two-step Method," GitHub repository, 2016. [Online]. Available: https://github.com/omaymega/2stepMethod.git.
[30] R. Sherwood and Y. A. P. KOK-KIONG, "Cbench: an open-flow controller benchmarker," 2013-05-13]. http://www, openflow, org/wk/index, php/Oflops. 2010.
[31] "Performance Tests - OpenDaylight Project." [Online]. Available: https://wiki.opendaylight.org/view/CrossProject:Integration_Group:Performance_Tests.
[32] "Performance Tests - ONOS." [Online]. Available: https://wiki.onosproject.org/display/ONOS/1.5%3A+Experiment+G+-+Single-node+ONOS+Cbench.
[33] "RYU controller stats." [Online]. Available: https://www.openhub.net/p/ryu.

AUTHOR PROFILES:

BELKADI Omayma was born in 1992, she is a PhD student since December 2014 in the National School of Applied Sciences in Tangier, belonging to Abdelmalek Essaâdi University, Morocco. She holds a Master degree in Communication systems and Informatics from the same school. She works on traffic classification in Software Defined networks, she has already worked on traffic classification during her Master thesis internship in INRIA's Paris-Rocquencourt research center, France, specifically within the LINCS laboratory, where her mission was about identifying features and patterns to detect and classify user's traffic.

LAAZIZ Yassin, Prof. Dr., was born in 1966, he received the "Doctorat de Troisième Cycle" degree from Abdelmalek Essaadi University (UAE) in Tetuan (Morocco) in 1992, and the "Doctorat d'Etat" degree in Solid State Physics from Cady Ayad University (UCA) in Marrakech (Morocco), in 1999. He has been a professor of Physics at the Faculty of Sciences & Technics of Marrakech (UCA) from 1992 to 1999. In 1999, he moved to ENSA of Tangier (UAE), where he has been the Head of the Department of Telecommunications & Electronics and the Coordinator of Graduate Engineering Program in Telecommunications & Networks. He is currently in charge of Master Degree Program in Telecommunication & Embedded Systems. His research interests focus mainly on Networks and Systems also on thin film devices for optoelectronics and energy conversion. He is the author and co-author of more than 30 papers, which appeared in refereed specialized journals and symposia.