Journal of Information Sciences and Computing Technologies(JISCT)
ISSN: 2394-9066

SCITECH
RESEARCH ORGANISATION|

Volume 7, Issue 2
April 27, 2018

Journal of Information Sciences and Computing Technologies
www.scitecresearch.com/journals

Impact of Controller Placement in an OpenFlow WAN

Ka Lung Ng, Hiu Ying Pun, Aastha Sharma, Stefan Weber, Eamonn O Nuallain
School of Computer Science and Statistics
Trinity College of Dublin,
Ireland
Email: ngka@tcd.ie, punh@tcd.ie, sharmaaa@tcd.ie, sweber@tcd.ie, eon@cs.tcd.ie

Abstract

The aim of this paper is to evaluate how the number and the position of controllers in an OpenFlow-enabled Wide Area Network would affect the time to completion of the flows. The experimental setting consisted of 12 switches with 3 hosts each and the number of controllers were varied from 0 to 12. 3 sets of hosts were considered for the experiment which consisted of 10 iterations between the same set of hosts to decrease the probability of individual error. Based on the experiment conducted, we found that increasing the number of controllers does not necessarily mean that the performance of the network will improve. For example, we found that when there was no controller, the mean time to completion of the flow was 5.1773 seconds. Whereas, when each switch was connected to a separate controller, the mean time increased to 7.2815 seconds for the same set of restrictions. It was also observed that the network performed well in some of the multiple controller placement scenarios and this is explained in more detail in the Results Section. Software Defined Networking decouples network architecture from infrastructure to achieve better flexibility, and it can be implemented using the OpenFlow protocol [1] among other approaches. Many experiments have been conducted using this protocol and some of them are reviewed in this paper.

Keywords: Software Defined Networking; OpenFlow; Wide Area Network.

1. Introduction

Current internet architecture faces problems related to Quality-of-Service (QoS), security and scalability. It was never designed to cater to the scale of users and services present today [2]. Moreover, modern Wide Area Network (WAN) links are expensive to be built and maintained. They should be widely available and all bits of information are treated in the same manner. Moreover, a failure means that all the applications would be affected equally, and this is undesirable as different applications have different sensitivities. To overcome these shortcomings, the concept of programmable networks came into existence wherein all the aspects of a network can be handled by a software which is independent of the network hardware. This is unlike the network architecture which is deployed currently wherein equipment from different vendors has its own software and each equipment needs to be tweaked separately to get the desired results [3]. Also, traditional networks have combined the control and the data plane together, i.e., the network hardware makes decision on traffic routing and also handles the traffic. The routers decide the route for the data packets and forward these data packets.

SDN borrowed from the above two concepts, i.e., programmable networks and control-data plane separation. Thus, a new network architecture came into existence wherein there was a centralized point of control which made all the decisions regarding how the traffic is forwarded, who is allowed access, etc. Moreover, the network hardware forwarded the flows according to the policies set by the centralized point of control. The main characteristics of a Software Defined Network are:

1) Centrally managed - SDNs are based on the concept of the centralized point of control, namely the controller.

2) Directly programmable - Since, the control plane is decoupled from the forwarding plane, the network functions and control can directly be programmed from the controller.

3) Agile - As it is directly configured, the network policies and control can be adjusted in real-time to meet the changing needs of the network-wide traffic flow. Our contributions in this paper are to (1) motivate the controller placement problem and (2) quantify the impacts of controller placement on real topologies by considering the number and the positions of controllers deployed with regard to different sizes of packet being transferred.
The essential benefit of SDNs is that the network traffic can be managed from a single point without the need to tweak individual switches, whereas this is required in the traditional network. SDN offers a very granular level of control. The main aim of SDN is to make the network respond swiftly to changing requirements through a centralized control. The most notable example of SDN being implemented is using the OpenFlow protocol which was first proposed to be deployed at Stanford University [3]. Another example of the concept of SDN being implemented is that of Ethane [4] but we will look at OpenFlow in detail as our experiment is based on it. The OpenFlow protocol is widely used to implement SDN because it has widespread support. There is a dedicated community, the Open Networking Foundation (ONF) [5], which focuses on using this protocol to implement SDN. OpenFlow provides a communication channel between controllers and network elements and allows flows to be directed from a source to destinations based on global knowledge. The protocol works as follow: Whenever a new traffic flow starts, the first packet of the flow will be sent to the controller which then decides what to do with the flow. If the controller decides to forward the packet, it will calculate a route for the packets and populate the flow table of the switches along that route with the information about the flow [3].

The paper is organised as follows - Section II contains background information about OpenFlow and research work which is relevant to our experiment. In section III, we explain the motivation and methodology which provides an overview of how we implemented and conducted the experiment. In section IV, the results of the implementation are provided with analysis, followed by the conclusion.

2. Background

OpenFlow is a communication protocol which was first proposed to be implemented at Stanford University [3]. It is possible to remotely control network hardware from different vendors using this protocol. The rationale is that there is a common set of functions which run on the network hardware while different vendors have their own flow tables. OpenFlow exploits these functions to make the entire network programmable [3]. A lot of research work and experimentation have been done on OpenFlow. Some of the work which is relevant to our project is discussed in this section.

Cai et al. [6], addresses the performance issue of a controller. It proposes to exploit parallelism to improve the performance of a controller. A multi-threaded system called Maestro is proposed which improves the throughput of the system by combining parallelism and through optimization techniques.
Min et al. [7], proposed an enhanced version of FlowVisor so that the bandwidth requirements of multiple users can be met in a network of many users without being affected by the traffic from other users in the same network. The setting used multiple controllers but only because the number of users were more, the impact of

multiple controllers was not studied. Hu et al. [8] proposed a controller load balancing architecture called BalanceFlow for OpenFlow WANs. It works on OpenFlow WAN and uses multiple controllers but it uses a 'Super Controller' to balance controller traffic whenever there is controller load unbalancing but the impact of the placement of those controllers was not studied.

Jarschel et al. [9] proposed a tool to evaluate the performance of the OpenFlow controller at a granular level. Many implementations of the OpenFlow controller have been developed and each of them is suited for a different scenario. This tool helps in deciding which OpenFlow controller to be used for a specific scenario. Turull et al. [10] evaluated the network performance using different types of controllers by emulating a fixed OpenFlow network on Mininet, but only one controller was present at a time. There are significant differences between controller performances but the topology used was quite small.

Heller et al. [11] is so far the only paper which investigated controller placement problem to discover how many controllers are needed for a given topology. Based on their experiments, they concluded that one controller is usually adequate but it also obviously depends on the topology.

The papers discussed in this section are in no way exhaustive and these are the papers which are the most related to this work. All the papers mentioned here focus on reducing flow completion time but using different approaches.
Our aim is, given a certain OpenFlow-enabled WAN, how is the time to completion of flows affected as the number of OpenFlow controllers in that WAN increase and their placement is changed.

3. Methodology

As mentioned, a lot of work has been done on the OpenFlow protocol but very limited amount of work considers the deployability of OpenFlow in a Wide Area Network (WAN). Most of the work only focuses on small networks. Therefore, we are going to evaluate the performance of the OpenFlow Protocol in a WAN.

To evaluate the impact of controller placement on the flow completion time in an OpenFlow WAN, an OpenFlow WAN is firstly established. Then different parameters such as number of controllers, position of controllers and file size transferred are varied to see how the performance of the network varies with the variation in the aforementioned parameters.

To implement a large scale SDN deployment, Google's B4 SDN WAN [12] is used and a topology similar to B4 is emulated in Mininet. We have used this topology because Google's B4 SDN is the most reliable SDN-WAN topology. The bandwidth for all the links is kept the same but the delay between the links is different. This is done to implement minimum delay experienced in the real world between the switches which are located at a significant distance from each other. The Mininet emulator has been used to implement the topology and study the network performance. We have used Mininet because it provides theoretically expected results for the restrictions set by us [13]. Also, it is possible to recreate networks which supports several protocols and topologies [14].

According to B4, there are 12 data centers and in our experiment these data centres are simulated by a network of 12 switches in Mininet and each switch is connected to three hosts. These switches are labelled from S1 to S12. The switches are also connected to each other. Finally, the dotted lines indicate the connection between controller and switches. An example topology has been shown in Fig. 2. In this example, all switches are connected to the same controller.

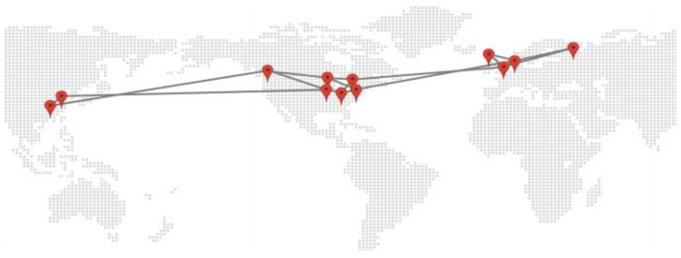
Fig. 1. Google B4 architecture

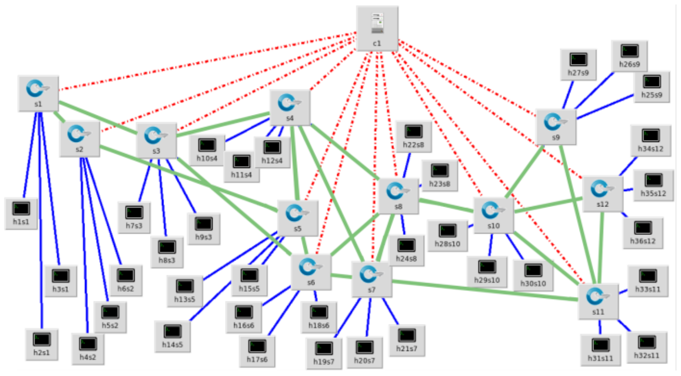

Fig. 2. Topology with 12 switches with the use of one controller

To evaluate the impact of controller placement on the flow completion time in an OpenFlow WAN, the above configuration is varied by increasing the number of controllers and changing the position of controllers. Afterwards, latency values are changed to simulate the distance between data centers. In general, the greater distance will cause greater latency in transmission by nature. Therefore, latency is used to define the distance in the virtual topology and the value of latency was taken from Wonder Network [15]. It has hourly measurements by doing 30 ping requests between locations. To avoid individual error, the values are obtained at 9 a.m. and 9 p.m. and the average delay value is obtained.

The next step is to generate the traffic and transfer it between different hosts. The ‘fallocate’ [16] command was used to generate files of 50KB and 200MB. This command is used because it is the most efficient command to allocate space on disk to create a file with the desired size. It is much faster to create a file using this method. Netcat is used for the purpose of traffic transfer because it is simple and supported on all platforms, such as Linux, Windows and MAC OS X. The packets transferred in this experiment are 200MB so that the file size is large enough to simulate real-world file transfer scenario. File size of 50KB is also tested to determine whether file size has impact on controller placement.

Initially, all the switches are connected to a POX controller and any traffic between the switches passes through this controller. After one iteration, stepwise, the number of controllers in the topology are increased and the position of the controllers is changed.

When the number of controllers increases, there are much more combinations than using one controller. For example, if two controllers are deployed, one possible combination is that switch 1 is connected to controller 1 while other switches are connected to controller 2. Another possible combination is that switch 1 and 2 are connected to controller 1 while other switches are connected to controller 2. Therefore, if 6 controllers are deployed, there are at least 12C6+12C2*11C6+12C3*10C6 = 77616 possible scenarios. In order to reduce the time for computing similar scenarios, the switches are grouped according to distance to reduce the number of possible combinations, labeled as Group A to E in Fig. 3.

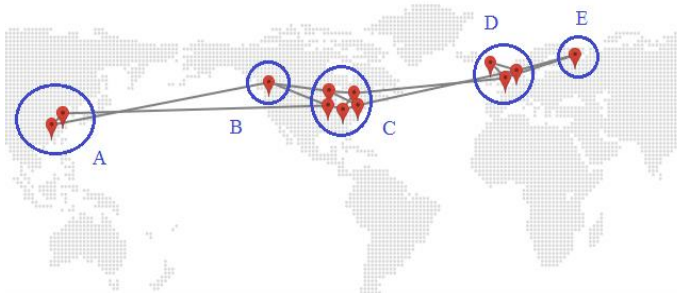

Fig. 3. Topology with 12 switches with the use of one controller

This is based on the assumption that the best network performance is given when the controller controls a group of switches which are closest to it. Therefore, for instance, when a controller is allocated to Group A, S1 and S2 are connected to that controller. For example, in the previous case, the scenario of switch 1 connecting to controller 1 while other switches connecting to controller 2, and the scenario of switch 1 and 2 connecting to controller 1 while other switches connecting to controller 2 are similar scenarios. Hence, testing these similar cases will be avoided. To further prove the assumption mentioned above, an experiment was also carried out and more details are explained in Section 4. After testing the multiple controller placement configurations, the final scenario is that no controller is deployed, i.e., traditional routing of data takes place. This serves as the control setup to show the effect of SDN.

4. Results

To evaluate the performance, communication in the OpenFlow network was monitored and the time for the packet to be transferred from its source to its destination was measured. When the time of flow completion is shorter, this indicates that network performance is better and, controller placement is better as well.

For each controller placement scenario, the time measured included file transfer from a host in S1 to a host in S2 (both were always under the same controller), from a host in S1 to a host in S12 (they had the largest distance and were under different controllers when we increased the number of controllers) and from host in S3 to S9 (a random combination with multiple hops). Ten trials were run and the average value was taken to avoid bias and individual error so that the results are more accurate.

The number of controllers are gradually increased from 0 to 12 and, the placement of multiple controllers is changed to see the effect of the controller placement on the topology. We have kept maximum of 12 controllers because we want to see the performance of the network when each switch is connected to a separate controller. For combinations which utilized more than 1 controller but fewer than 12 controllers, 6 different placements, including 3 random controller placements according to the grouping in Fig. 3 and 3 totally random controller placements, were implemented and then the average of the data obtained for these 6 placements is taken for plotting the graph.

To evaluate the performance, maximum, mean and minimum values of 10 trials for each combination of switches were plotted on the graphs. The upper bound and lower bound represent the maximum and minimum time taken among all trials for that configuration respectively, and the curve between the bounds indicates the mean time averaged over the number of different placements implemented. One hop corresponds to data transfer between host in S1 and host in S2, largest number of hops corresponds to data transfer between S1 and S12 and, random number of hops corresponds to data transfer between S3 and S9.

4.1 Effect of different file sizes on flow completion time when deploying different number of controllers

From Fig. 4 we see that in all scenarios, i.e. one hop, multiple hops and largest number of hops, the difference in flow completion time is less than 0.5 seconds, regardless of how many controllers were deployed. One possible reason is that the file size was very small to give an obvious difference in the results. It is also notable that the time required for the setup without controller to finish file transfer was similar to the setup of deploying controllers, indicating that they are having similar packet transmission rate. Therefore, the network performance without controller is similar to the scenario of deploying controllers.

However, from Fig. 5 we see that the required time for flow completion on deploying different number of controllers had greater discrepancies when comparing with the scenario of smaller size file transfer, particularly in the scenario of largest number of hops. The average flow completion time ranges from 82 seconds to 100 seconds. The large difference in completion time indicates large difference in the network performance. This also shows that controller placement has more effect on network performance when the packet size is larger and the transmission route is longer which contains numerous hops. Details about the effect of the number of controllers on flow completion time is explained in the following part based on this scenario.

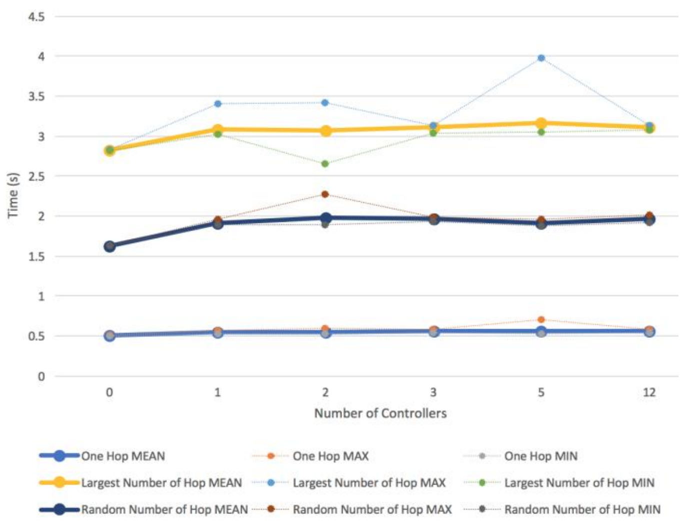

Fig. 4. Graph to show the relationship between number of controllers and 50KB packet transfer completion time

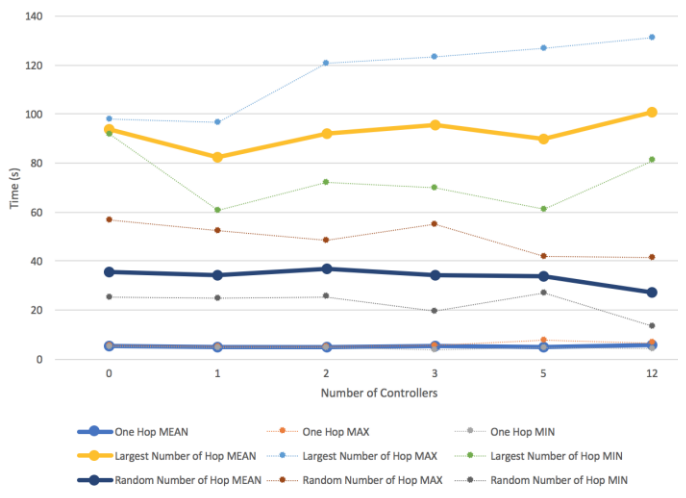

Fig. 5. Graph to show the relationship between number of controllers and 200MB packet transfer completion time

4.2 Effect of deploying different number of controllers on packet transfer completion time

As explained in the previous part, this part focuses on investigating the effect of deploying different number of controllers on 200 MB packet transfer between the hosts with the longest distance and largest number of hops in between.

Fig. 6 focuses on the relationship between the presence of 1 controller and completion time for 200MB file in largest number of hops in topology. From the result, the flow completion time reduced from 93 seconds when no controller was present to 82 seconds on average when one controller was deployed. Since the time of flow completion has reduced, this proves that the packet transmission rate is faster and the performance of network improves when there is one controller as compared to the scenario of no controller.

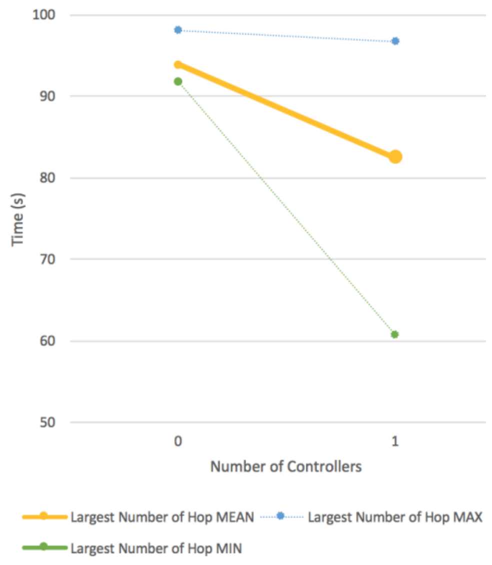

Fig. 6. Graph to show the relationship between the presence of controller and 200MB packet transfer completion time from host 1 (in Switch 1 in Group A) to host 36 (in switch 12 in Group E)

However, in Fig. 6, from the middle line in the topmost group of three lines which indicates the average flow completion time in largest number of hops in Fig. 5, it was observed that increasing the number of controllers did not decrease flow completion time. This implies that the performance of the network was not improved. Instead, as the number of controllers increased, the flow completion time increased gradually, and this implies that the performance of the network degraded. The network performed worst when there was a separate controller for every switch. One possible explanation is that no controller has a global view of the topology. It only knows the next hop and as such, flow table population cannot take place at a competitive rate. This leads to an increase in terms of flow completion time.

Another observation from Fig. 5 is that the flow completion time without using controller ranges from 91 second to 93 seconds among 10 trials, whereas the flow completion time with using one controller ranges from 60 seconds to 82 seconds among 10 trials, resulting in more than 22-second difference between the best and the worst trials. The transmission time further fluctuates when we increase the number of controllers.

We also found that although using multiple controllers yielded similar mean as without using controller, they indeed yielded a much wider range of result among trials. The best trial of using multiple controllers (less than 72 seconds except the scenario of deploying 12 controllers) is much better than that of without controller (91.8

seconds), whereas the worst trial of using multiple controllers (more than 120 seconds) is much worse than that of without controller (93.8 seconds). A possible reason for the great difference is that some combinations may deteriorate performance. This will be further discussed in the following part.

4.3 Effect of controller position on packet transfer completion time

One controller may sometimes raise other concerns such as fault tolerance and scalability issues. For instance, the controller may break down or it may not be able to handle very large amount of switches. Therefore, one controller may sometimes not be feasible, and deploying multiple controllers with optimal placement needs to be considered.

Besides, it is possible that some combinations of multiple controllers may deteriorate the performance. To validate this assumption, an experiment was conducted on the separate grouping of switches which are then controlled by separate controllers.

This experiment was similar to the previous part in measuring the flow completion time from host 1 in switch 1 in Group A to host 36 in switch 12 in Group E, but focusing on the performance of 9 different combinations of 2 controllers' deployment. The first 3 combinations, namely in Group (I), are that one of the controllers has the view of both Group A switches and Group E switches, while another 3 combinations, namely in Group (II), are that no controller has the view of Group A and Group E at the same time. The remaining 3 combinations, namely in Group (III), are random grouping. Fig. 6 has shown the grouping for controllers and the corresponding results in the experiment.

TABLE I. THE RELATIONSHIP BETWEEN CONTROLLER POSITION AND 200MB PACKET TRANSFER COMPLETION TIME FROM HOST 1 (IN SWITCH 1 IN GROUP A) TO HOST 36 (IN SWITCH 12 IN GROUP E)

Grouping			**Time (s)**		
Requirement	***Controller 1***	***Controller 2***	***MEAN***	***MAX***	***MIN***
(I) Connect Group A and Group E	A,C,E	B,D	88.9	99.1	68.0
	A,E	B,C,D	88.8	106.8	74.7
	A,B,C,E	D	88.2	97.6	76.4
(II) Separate Group A and Group E	A	B,C,D,E	88.2	100.7	72.2
	A,B,C	D,E	90.5	104.6	71.4
	A,B	C,D,E	90.5	108.6	70.6
(III) Random grouping	S1,S12	S2-S11	90.4	101.2	77.6
	S1-S6	S7-S12	89.0	100.7	71.7
	S1,S3,S5, S8,S10, S12	S2,S4,S6, S7,S9, S11	89.5	106.7	65.6

From Table I, the average flow completion time of 9 combinations is between 88.2 and 90.5 seconds, and the performance difference between combinations is very slight.

However, from Table I, it is notable that Group (I) has slightly better performance than Group (II) and Group (III), particularly in the grouping of Group A, B, C, E in one controller while Group D in another controller. Three factors may have contributed for the same: Firstly, a controller should have the view of as many switches as possible. This explains why the third scenario in Group (I) has better performance than other two scenarios in Group (I). The second factor is that controller should contain the source and destination. This explains scenarios in Group (I) are better than that in Group (II). The third factor is that a controller has the view of switches which are nearby. For example, some scenarios in Group (III) also contained switch 1 and 12 in one controller, but the performance was still relatively weaker than scenarios in Group (I). Also, these three factors are not in conflict with the conclusion of the last part that a controller with centralized view of all switches yields the best performance.

5. Discussion and Conclusion

In this paper, we propose to evaluate the impact of the controller placement on an OpenFlow WAN topology. Our focus was to measure the time for flow completion while varying the number of controllers and their placement. We observed that increasing the number of controllers does not necessarily lead to a performance boost in the network.

Based on the results we got from our experiment, we have made the following conclusions:

1) One controller is generally sufficient to handle the network needs.

2) Controller placement has more effect on network performance when the packet size is larger and the transmission route is longer which contains numerous hops.

3) A network in which a controller has a global view of the topology will perform better than the network which has multiple controllers and no controller with global view. However, if the controllers are statically configured, then time will not be spent on calculating the route and the transfer time would be reduced thereby improving the performance.

4) When deploying multiple controllers, some of the configurations will perform worse than other configurations.

We have provided a basic topology, taken from Google's B4 architecture which can be studied upon further to analyse how exactly the OpenFlow protocol can be improved upon so that the services can directly 'talk to' the controller(s) about the policies they want to implement. An interesting research area would be how an OpenFlow WAN would work if network slicing is implemented. Another one would be statically configuring the controllers so that they have the best route between different switches. This would decrease the response time of the network. As such, the controller would only need to compute the route if the switch(es) along the configured route fail. It would be interesting to see the performance of an OpenFlow-enabled network under these circumstances.

References

[1] *OpenFlow Switch Specification*, version 1.1.0, Open Networking Foundation, February 2011.

[2] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* April 2008, vol 38, no. 2, pp. 69-74.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," *ACM SIGCOMM Computer Communication Review*, August 2007, vol 37, no. 4, pp. 1-12.

[5] Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/ [Accessed: 9 March 2017]

[6] Z. Cai, A. L. Cox, and T. S. E. Ng, "Maestro: A System for Scalable OpenFlow Control," Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08, 2010, pp. 1-10.

[7] S. Min, S. Kim, J. Lee, B. Kim, W. Hong, and Jonguk Kong, "Implementation of an OpenFlow network virtualization for multi-controller environment," in *Proceedings of 14th International Conference on Advanced Communication Technology (ICACT)*, PyeongChang, South Korea, February 2012, pp. 589-592.

[8] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks," in *the Proceedings of 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, Hangzhou, China, October 2012, pp. 780-785.

[9] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," *2012 European Workshop on Software Defined Networking*, Darmstadt, Germany, October 2012, pp. 48-53.

[10] D. Turull, M. Hidell, and P. Sjödin, "Performance evaluation of openflow controllers for network virtualization," in *Proceedings of 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*, Vancouver, BC, July 2014, pp. 50-56.

[11] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, August 2012, Helsinki, Finland, pp.7-12.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, New York, USA, August 2013, pp. 3-14.

[13] J. Ortiz, J. Londoño, and F. Novillo, "Evaluation of Performance and Scalability of Mininet in Scenarios with Large Data Centers," *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*, Guayaquil, Ecuador, October 2016, pp. 1-6.

[14] B. Lantz, N. Handigol, B. Heller, and V. Jeyakumar, "Introduction to Mininet". [Online]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet [Accessed: 25 January 2017]

[15] WonderNetwork. Global Ping Statistics [Online]. Available: https://wondernetwork.com/pings [Accessed: 9 March 2017].

[16] Linux Programmer's Manual. [Online]. Available: http://man7.org/linux/man-pages/man2/fallocate.2.html [Accessed: 25 February 2017] .