Performance of Load Balancing Algorithms for SDN Controlled Data Center Networks with Leaf-Spine Topology

A. Furkan Okuyucu, Cem Karataş, Albert Levi, Özgür Gürbüz

Sabanci University
Faculty of Engineering and Natural Sciences
Orhanli-Tuzla, 34956, İstanbul, Turkey

okuyucu@sabanciuniv.edu, cemkaratas@sabanciuniv.edu, levi@sabanciuniv.edu, ogurbuz@sabanciuniv.edu

Abstract – Traffic in Data Center Networks (DCN) has increased drastically over the years. With the emergence of Software Defined Networking (SDN), the information gathered about the network has increased significantly. Apart from SDN, a solution proposed in DCNs to decrease the latency and complexity is using the leaf spine topology. This topology has taken its place in DCNs that experience east-west traffic rather than the north-south traffic, and the need for high availability for information in DCNs has led the research for accurate and efficient load balancing methods. In this paper, the performance of six different load balancing algorithms, namely random, Round Robin, Least Flow, Least Utilization, Proactive and Laberio, are evaluated and compared for DCN with leaf spine topology, using ONOS as the SDN controller for the first time in the literature. The performance metrics utilized include Round Trip Time (RTT) standard deviation of RTT, instantaneous and average throughput. It has been observed that Least Flow has the smallest average RTT and smallest standard deviation of RTT among the load balancing algorithms. In terms of throughput, Proactive and Laberio give the best results, and among all, Laberio is the only method that responds to the changes which occur after the installation of flow rules to the switches.

Keywords: load balancing, leaf-spine topology, Software Defined Networking (SDN), ONOS, Data Center Networks (DCN)

I. Introduction

In traditional Data Center Networks (DCNs), managing the network, the control protocol complexity, vendor dependency and security are the major problems. The main cause of these problems is that the switches used in these networks are black box switches. Black box switches come as a single package of hardware and software. This restricts the control over the network. However, with the emergence of Software Defined Networking (SDN), the control plane and the data plane are separated [1]. This separation makes it possible to use white box switches in the networks. White box switches come as a standard hardware and users install applications according to their needs without depending on a certain vendor. This solves the vendor dependency and security problems. The separation also makes it possible to manage the network in a better way, since there is a centralized controller, which can control all the switches in the network and gather any kind of information about the network, such as link utilization(s), loads on switches, number of packets passing through each different port etc. Accessing this information from a central node requires new solutions to another major problem observed in DCN, which is load balancing.

Load balancing is a crucial problem because when the underlying infrastructure of the Internet faces imbalanced traffic load, the delays and congestion on links increase and this causes a significant decrease in end user experience. Load balancing methods solve this problem by distributing the traffic equally among the links.

Various load balancing algorithms have been proposed for SDN-enabled DCN, such as round robin, least traffic and Laberio. In the literature, there exists comparative, but limited performance evaluation of these algorithms: In [2], dynamic load balancing algorithms on SDN-enabled networks are discussed, and throughput, delay and packet loss performance of the algorithms are compared, using Open Daylight as SDN controller. Topology is based on traditional tree-tier network design. In [4], the authors propose a method to increase performance on Web Servers using SDN-based load balancing algorithms. RTT, throughput and resource utilization are evaluated and compared. Simulations performed on a topology which consists of one controller, one switch and three servers. The controller used in this setup is POX. The authors of [7] compare the Round Robin load balancing algorithm with random load balancing. The topology and the controller used in [7] is the same as the ones used in [4]. Results are based on the average response time and the total transactions per second. Load balancing tests on software define multi-radio mesh networks are performed in [5], using ONOS as the SDN controller, while proposing a proactive load balancing algorithm.

Apart from the highly used traditional three-tier network in DCN, which is also used in [2], leaf-spine topology has emerged as an important and commonly used alternative DCN network topology [13]. Some of the advantages of this topology over fat-tree topology are link redundancy, decreased and fixed hop count, bandwidth enhancement and decreased delay. Having two layers in leaf-spine decreases the deployment and maintenance cost, since enterprises deploy

fewer switches. Leaf-spine is more agile and modern than the traditional network topology, and it is preferred for networks experiencing more east-west traffic than the north-south traffic.

To sum up, we can say that SDN-based DCN traffic management techniques balance the load using the information gathered from SDN controller. As discussed above, there are works on the performance evaluation of load balancing in SDN-enabled networks [2,4,5,7]. However, none of these studies perform their test on the newly emerged DCN leaf-spine topology and they do not compare the proactive, reactive and dynamic load balancing algorithms under the same conditions, simulation environment or testbed. Apart from that, the SDN controller used in [4] and [7], namely POX, is described as not suitable for production level environment due to lack of performance [11].

In this paper, we have implemented and comparatively analyzed the performances of four reactive, namely, random, round robin, least flow, least utilization methods, a proactive and a dynamic load balancing method, namely Laberio on a DCN with leaf-spine network topology. The SDN controller of our setup is ONOS (Open Network Operating System), which is developed and maintained by Open Network Foundation (ONF) [10]. The load balancing algorithms are implemented as an ONOS application. Using these applications, we have measured Round Trip Time (RTT), standard deviation of RTT, instantaneous and average throughput. We have observed that on leaf-spine topology using ONOS, least flow method gives the smallest RTT and also the smallest standard deviation of RTT. Another outcome is that the proactive method gives the highest throughput and Laberio is the only method that responds to load imbalances emerging after flow rule installation.

The rest of this paper is organized as follows. Section II gives background information about the load balancing problem and leaf-spine network topology. Section III discusses the details of load balancing algorithms utilized in our work. In Section IV the simulation setup and the simulation results are provided, and finally Section V concludes the paper.

II. Load Balancing Problem and Leaf-spine Topology

Imbalanced traffic loads cause increase in congestion and delay in a network; some links remains underutilized, while others face loss and high delay due to high traffic. These problems can be solved by distributing the load equally. The distribution mechanism may differ from algorithm to algorithm, but the desired outcome is the same. These mechanisms rely on the network data to make decisions in appropriate manner. With the arrival of SDN, gathering and accessing network data becomes an easy process. With these data, load balancing becomes more advanced in terms of accuracy; therefore, it has better performance. As a result, it can be said that two main purposes of load balancing are (i) to maximize throughput, and (ii) to equally distribute the traffic in the network. In addition to this fact, we can also state that SDN-based load balancing is highly applicable in DCN. Load balancing has many algorithms that manages to find the shortest path by considering different network data such as link usage, flow count, hop count.

Moreover, in data centers, it can be observed that due to the changes in traffic patterns, traditional network topology with three layers has left its place to more modern and agile two layers network architecture, which is leaf-spine. It is composed of leaf switches and spine switches as can be seen in Figure 1 [13]. Data center networks experience more east-west traffic than the north-south traffic. For this type of traffic leaf-spine provides high performance and low latency since having 2 layers provides maximum of two hops to get to any other device. Another outcome of the leaf-spine network topology is that it is scalable and redundant given that if more capacity is wanted, the solution would be simply to add one more leaf and on the other hand adding more spine would increase the bandwidth and resiliency. Furthermore, it may be said that one of the most beneficial advantages of leaf-spine network topology is to be able to use several network paths at the same time in other words it uses all interconnection links and this fact can lead to reduction of congested links.

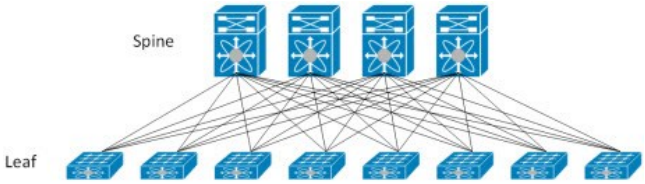

Figure 1: Leaf-Spine Architecture [13]

III. Load Balancing Methods

We classify the load balancing methods in three groups: *reactive*, *proactive* and *dynamic*. Reactive performs path selection on-demand. Proactive performs path selection and calculation periodically even when there is no demand. These two applications install the flow rules using per flow logic. In other words, they do not change the path of the flow after setting it. Dynamic Application differs at this point. It may act before the flow finished as opposed to reactive and proactive applications examined in this paper.

A. Reactive Methods

These methods compute and select path for flows on-demand. When the first packet of the flow comes, switch checks whether flow rules are installed for this flow or not. If the flow rules are not installed, it sends the packet to SDN controller. Then SDN controller processes the packet and gets source and destination from the packet. With this information applications perform path calculation and flow rule installation. Reactive load balancing methods get into path selection at this application. At the end of the path selection SDN controller installs flow rules to the OpenFlow switches on the selected path. The same process happens for each new flow. The implemented methods are as follows; random, round robin, least flow and least utilization.

i. Random (Reactive Forwarding Application)

This method computes all shortest paths using k-shortest path algorithm for each flow when the first packet of the flow reaches to controller. Then it selects the path randomly from the paths whose hop count is among the smallest hop count in the set. Since the inputs of this method do not change over time unless a link failure or switch failure happens, its path computation and selection time remains still for different link utilization in the network. Thus, this method gives a stable latency. ONOS's Reactive Forwarding Application is used to test this method. [6]

ii. Round Robin

The round-robin load balancing method performs path selection using round robin fashion on spine switches while selecting the path for the incoming flow. [7] With this way, it distributes flows equally among the spine switches. We implemented this method as an ONOS application. This application stores a global integer which increases as the new flows coming to the network. It selects path such that it passes through the i^{th} spine switch. When i exceeds the spine count it is set to 1. This method balances the flows among spine switches, but it does not necessarily balance the load since it does not take flow's load into account.

iii. Least Flow

This method gets shortest paths between source and destination. Then it gets the number of flows passing through each path in the set as an input. With this input, it selects the path whose flow count is the smallest. This selection distributes the flows on links equally which reduces the number of underutilized links. From this point, it can be said that this algorithm considers fairness between links while selecting the paths for incoming flows. We implemented this method based on [5].

iv. Least Utilization

This method differs from the least flow method in a way that it does take the maximum minimum available link utilization on the paths. Then selects the path whose max-min is greatest. After selection it installs the flow rule to switches. We implemented this method based on [5].

B. Proactive Methods

Proactive methods perform path selection before the flows come to network. The paths for each host pair are calculated and installed on switches. These flow rules are renewed in a certain interval. The renewal of the flow rules causes an overhead in the processing. The implementation of this method adapted from the ONOS application implemented by [5], which balances the load proactively for SDN-enabled wireless mesh networks to use in leaf-spine topology. This application first gets the host list from SDN controller. Then it computes shortest path for each host pair. The selection is performed based on link utilization. If the link utilizations are equal, then the selection is based on flow counts of paths. If the flow counts are equal too, then the hop counts compared.

C. Dynamic Methods

Dynamic methods differ from the previously mentioned algorithms in a way that it breaks one path selection per flow logic. These methods take actions on the active flows if a certain condition is reached. With these actions, these methods make sure that the load kept balanced between links. The condition is checked in a time interval and it causes an overhead in the processing of the packets [3]. There are different t algorithms proposed for dynamic load balancing in SDN as discussed in [2]. From the discussed algorithms we select Laberio to compare with other load balancing algorithms since it can be referred to the first work to introduce an SDN-based dynamic path optimization algorithm.

We implemented Laberio as an ONOS application. It aims to minimize the latency and maximize the throughput and keep the load invariance between spine switches below a threshold. When a flow comes, Laberio selects the path whose available bandwidth is maximum. It takes the load passing through spine switches as input. With this input, it computes the load invariance between spine switches. It checks whether this value exceeds the threshold at every 10 seconds. When the threshold exceeds, it selects the flow whose load is the greatest among the flows passing through the overutilized spine switch and it moves this flow to the underutilized spine switch [5]. This decreases the load invariance between spine switches. Since the input of this algorithm is very fluctuating, the processing time may change for different conditions in the network. The threshold checking also creates an overhead on the processing time of the packets.

IV. Performance Evaluation

A. Simulation Setup

For the simulations, the leaf-spine topology depicted in Figure 2 has been created in Mininet, an emulator tool for SDN-based networks [8]. The topology consists of two layers; two spine switches on first layer and four-leaf switches on second layer. In the simulations with the Mininet tool, the switches are implemented as OpenFlow switches and the SDN controller is running ONOS controller. The load balancing algorithms to be compared are implemented in this tool as an ONOS application.

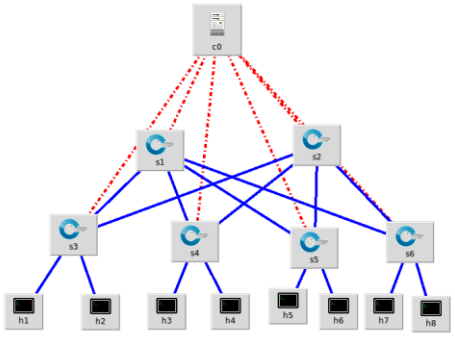

Figure 2: Leaf-spine Topology used in Simulation

As the traffic scenario for evaluating the load balancing algorithms, we have considered two hosts for each leaf spine switch and one of the hosts (h1), acting as a TCP server, while others are acting as TCP clients. The simulation scenario involves the events listed on Table 1. At the beginning, h1 is the TCP server. Hosts h8, h6, h7 start to send TCP packets of 2962 bytes to server consecutively, with 30 seconds in between. Then, at the 90th second, h6 stops its flow to create an imbalance in the network. 30 seconds after that, h5, h4, h3 starts to send TCP packets of 2962 bytes, in order, with 30 second intervals. At the end of 240 seconds the simulation is ended. The packets are sent and received via Iperf, which is a tool used to create TCP clients and server at hosts [9]. With the help of Wireshark network analyzer tool, we have captured packets at the server h1 and TCP client h8. Then from the captured packets we have observed the RTT and throughput measurements for each load balancing algorithm.

Table 1: Timeline of events occurring in the simulation

Time(sec)	Event
0	h8 starts sending TCP packets to h1
30	h6 starts sending TCP packets to h1
60	h7 starts sending TCP packets to h1
90	h6 stops sending TCP packets to h1
120	h6 starts sending TCP packets to h1
150	h5 starts sending TCP packets to h1
180	h4 starts sending TCP packets to h1
210	h3 starts sending TCP packets to h1
240	The end of Simulation

B. Simulation Results

Considering a DCN with described leaf-spine topology and applying the described traffic scenario, the performance of load balancing algorithms is evaluated and compared in terms of the metrics below.

1) Round Trip Time (RTT)

Instantaneous RTT values are obtained for host h8 via wireshark for different load balancing methods. Host h8 is selected for examining RTT values because it is the only host that stays active throughout the simulation, providing the most reliable results among the hosts. RTT values gives us the idea about the reaction and changes by the algorithms with respect to certain network conditions and the effects of these changes.

Figure 3a shows the instantaneous RTT values for reactive load balancing methods. In this figure, ONOS's Reactive Application [6] is labeled as random. For this method, it can be observed that the RTT value increases when h7 is added as a TCP client to network at the 60th second. This increase indicates that random application gives h7 and h8 the same links to send packets to h1. Using the same links for the hosts at same leaf switch (h7, h8) shows that there are links which stay idle throughout the simulation. This proves that the (random) reactive forwarding does not consider fairness between links in terms of link utilization and this decreases the total number of packets processed. This indicates that no other host's flow is added to the spine switch used by h8. From this point, it can be said that (random) reactive forwarding does not balance the load and flows among spine switches.

The measurements for the round robin method in Figure 3a show that there are three jumps in h8's RTT value with round robin load balancing application. For each of two TCP clients there is a jump at this value. It means that for each of the two hosts there is a one host whose flow's path passes through the same spine switch as h8. These three jumps correspond to TCP client adding events for h7, h5 and h3. Therefore, at the end there are four hosts using the same switch. This clearly shows that round robin also does not necessarily balance the load between spine switches since it does not take recreated or stopped flows into account.

As can be seen in Figure 3a, least flow method is the only application which does not have an RTT jump at 60th second when h7 is added as a TCP client. This shows that least flow method distributes the load and flows fairly between links. The links which have remained idle in the previous methods are not idle in this method. From the jumps in Figure 3a, it can be said that h8, h5 and h3 use the same spine switch. It indicates that this method also distributes the flows equally among the spine switches. Since this method does not consider the bandwidth used by flows, it does not necessarily balance the load as well.

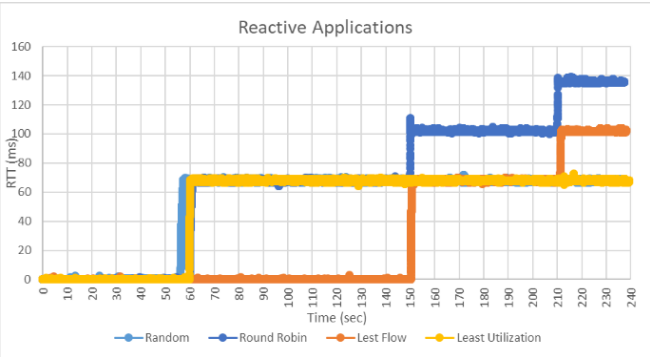

Figure 3a: RTT for Reactive Load Balancing Methods

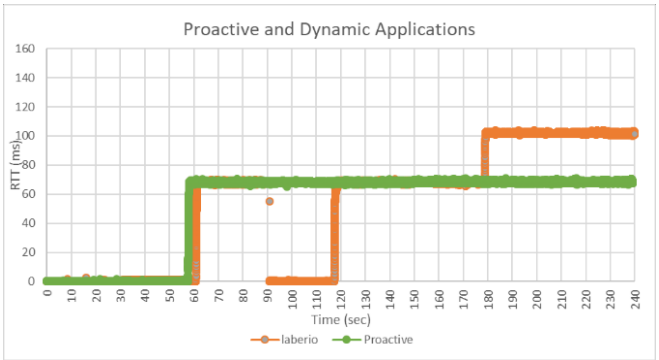

Figure 3b: RTT for Proactive and Dynamic Load Balancing Methods

RTT measurements with the least utilization method in Figure 3a indicate that it performs similarly as random, reactive forwarding. This behavior is caused by the fact that the links from both spine switches to host h1 is fully utilized before the host h3 starts its flow at the 210th second and the algorithm considers link utilization as 100% for both spines. Therefore, it can be said that this method does consider link fairness and balances the load when the links are not fully utilized. From the distribution of the flows at the end of the simulation it can be said that this method does not balance the flows between spine switches when links are fully utilized.

Figure 3b shows results for proactive and dynamic load balancing methods. RTT measurements indicate that proactive load balancing method does not consider link fairness. The utilizations of the spine switches are equal, but the number of flows is different at each spine switch. Since the period of optimizing the path between host is large (60 seconds), this method does not send flows considering the current situation of the network. This is one of the drawbacks of the proactive approach.

From Figure 3a and Figure 3b, it can be said that Laberio is the only load balancing method that adapts itself to the network event at 90th second. The h6 stops it's flows at 90th second. This creates a load imbalance between spine switches. Since h7 and h8's flows are passing through the same spine, the other spine stays idle between 90th and 120th seconds. When the load imbalance passes the threshold, this load balancing method selects the h7's flow and moves it to the idle switch. This movement decreases the RTT between 60th and 90th seconds, as it can be seen in Figure 3b. At the end, there are three flows on each switch. The equal distribution indicates that this method balances the load equally among spine switches.

2) Standard Deviation of RTT

Figure 4 shows the standard deviation of RTT measurements for all load balancing schemes. The figure shows that least flow has the smallest RTT standard deviation. Since the least flow distributes the flows fairly between links, the number of idle links is smaller than the other algorithms. Thus, the number of interfering network events with h8 are smaller for this algorithm. This leads to smaller fluctuation in the RTT values, hence smaller standard deviation.

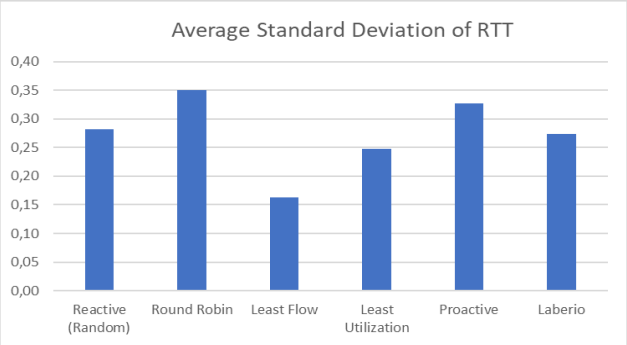

Figure 4: Average Standard Deviation of RTT values for: all Load Balancing Methods

3) *Throughput at Server*

Figure 5a and Figure 5b show the instantaneous throughput measured at the server host, h1, for the reactive and proactive load balancing methods.

As it can be seen from Figure 5, when there is just one flow in the network all algorithms forward traffic at the same rate as expected. Their performance starts to differ at 90th second, due to h6's flow stopping. Laberio's throughput is observed to be the highest between 60th and 90th seconds.

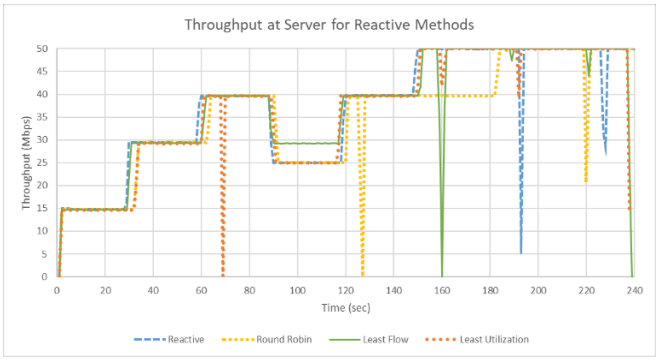

Figure 5a: Throughput at server for Reactive Load Balancing Methods

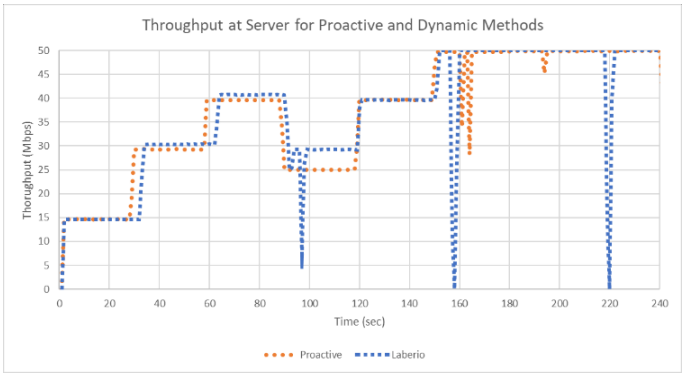

Figure 5b: Throughput for Proactive and Dynamic Load Balancing Methods

Figure 6 shows the average throughput of all load balancing algorithms. As depicted by the figure, the proactive method has highest throughput, since the flow rules are already installed on the switches when the first packet of the flow comes. The Laberio method performs the second best among the implemented algorithms. This is because Laberio is the only algorithm which balances the load dynamically when the h6 stops its flow at 90th second and gets higher throughput.

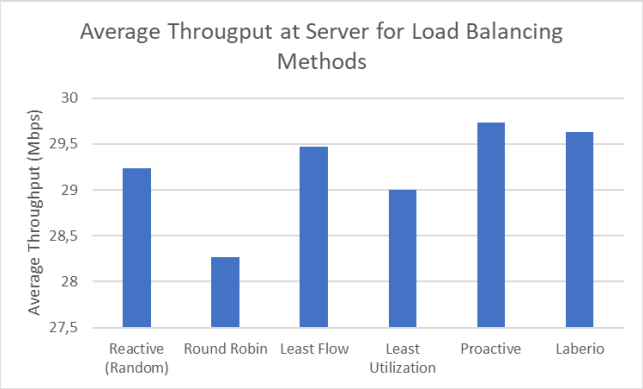

Figure 6: Average Throughput for all load balancing methods

V. CONCLUSIONS

In this paper, we have evaluated and compared four reactive a proactive and a dynamic load balancing method which are random, round robin, least flow, least utilization, proactive and Laberio algorithms for a leaf-spine DCN. Realistic simulations (emulations) are performed on the leaf-spine topology using ONOS as SDN controller. The RTT and throughput metrics are observed. The results indicate that Least Flow has the smallest standard deviation of RTT. It has been observed that Proactive and Laberio has the highest throughput and Laberio is the only method which responds to load imbalances emerged after the path selection.

In the light of these results it can be said that for networks which require stable and small RTT it is better to use Least Flow load balancing method. Another outcome is that Laberio and Proactive methods suits better to the networks which require higher throughput, at the cost of higher CPU usage. For networks which face highly varying flows, Laberio performs best because of its dynamic characteristics.

ACKNOWLEDGEMENTS

This work supported in part by ULAK HABERLESME A.S. in the form of industry focused graduation project.

REFERENCES

1. Open Networking Foundation. 2016. *ONF SDN Evolution*, 2016.
2. U. Zakia and H. B. Yedder, "Dynamic load balancing in SDN-based data center networks," *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2017.
3. H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: Dynamic load-balanced Routing in OpenFlow-enabled Networks," *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013.
4. Suwandika, I. P. A., Nugroho, M. A. & Abdurahman, M. (2018). Increasing SDN Network Performance Using Load Balancing Scheme on Web Server. *2018 6th International Conference on Information and Communication Technology (ICoICT)*. doi:10.1109/icoict.2018.8528803
5. Schlebusch, M. R. (2017). Topology Control and Routing inSoftware Defined Multi-Radio Mesh Networks. Retrieved January3,2019,from https://git.fslab.de/mmklab/ONOS_based_Load-Balancing_in_SDWMNs.
6. Opennetworkinglab. *Reactive Forwarding Application-ONOS.* Accessed from https://github.com/ opennetworkingl ab/onos/blob/master/apps/fwd/src/main/java/org/onosproject/fwd/R eactiveForwarding.java
7. Kaur S., Kumar K., Singh J. and Ghumman N.S. 2015, March. Round-robin based load balancing in Software Defined Networking. In Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on (pp. 2136-2139). IEEE.
8. Team, M. *Mininet*. Accessed from http://mininet.org/
9. GUEANT, V. *iPerf - The ultimate speed test tool for TCP, UDP and SCTPTest the limits of your network Internet neutrality test*. Accessed from https://iperf.fr/
10. Open Network Foundation. *ONOS Project*. Accessed from https://onosproject.org/
11. Stancu, Alexandru L., et al. "A Comparison between Several Software Defined Networking Controllers." *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, 2015, doi:10.1109/telsks.2015.7357774.
12. Betzler, August, et al. "On the Benefits of Wireless SDN in Networks of Constrained Edge Devices." *2016 European Conference on Networks and Communications (EuCNC)*, 2016, doi:10.1109/eucnc.2016.7561000.
13. *Figure 1.* Leaf-Spine Architecture Design. January 8, 2019, retrieved from https://blog.westmonroepartners.com/a-beginners-guide-to-understanding-the-leaf-spine-network-topology/
14. Alizadeh, M., & Edsall, T. (2013). On the Data Path Performance of Leaf-Spine Datacenter Fabrics. *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*. doi:10.1109/hoti.2013.23