Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

Dynamic Load Balancing Algorithm in SDN-based Data Center Networks

Gudu Bekama Haile
[1] Applied Computer Technology,
School of Information Technology and Engineering,
[2] Tianjin University of Technology and Education, Tianjin 300222, China

Jianxun Zhang
[1] Applied Computer Technology,
School of Information Technology and Engineering,
[2] Tianjin University of Technology and Education, Tianjin 300222, China

***Abstract*— Current networking architectures have many drawbacks that must be overcome to meet modern IT requirements. To overcome these limitations; Software Defined Networking (SDN) is taking place as the new networking approach. One of the major issues is that they use static switches that cause poor utilization of the network resources. Another issue is the slow response and delays are the main problems in current networking trend. This research proposes an implementation of a dynamic load balancing algorithm for SDN based three tier Data Center network to overcome these issues. A test has been implemented using Floodlight controller as SDN controller and Mininet software to emulate the network. Python programming language is used to define a network topology and to write the load balancing algorithm program. Finally, iPerf is used to test network performance. The network was tested before and after running the load balancing algorithm. The testing focused on Quality of Service (QoS) parameters such as throughput, bandwidth, and response time. The algorithm increased throughput with at least 50% at distribution level and with 33% at core level which also result in faster response time as well.**

Keywords*— *Dynamic Load balancing, Software Defined Networking, Data Center Network, Floodlight Controller,

1. INTRODUCTION

In the last 20 years networks requirements have been changing constantly, the amount of traffic has been increasing exponentially and more demanding end-to-end goals are needed. Data centers are the main hosting infrastructures of Internet applications and services (i.e., multimedia content, Internet banking, and social networks). Traditional load balancing methods in such data center networks use dedicated hardware devices to distribute the network traffic in different server replicas. Although this approach achieves high performance in general, it is expensive and lacks flexibility in its configuration, which cannot be dynamically adjusted based on real-time network state or other information. Now days a lot of data is being transferred in software defined networking. There is a lot of traffic that has increased to a great extent. There is an urgency to deal with this type of congestion problem using an efficient algorithm networks can be improved. Software defined networking helps in decoupling data and control plane. Thus the network administrators manage their network by using an external SDN controller. A special type of controller is used that has the capacity to change the forwarding behavior of the network. This behavior has to be changed directly. The network framework that is depicted by SDN is flexible to a great extent and the network administrator can take the profit of the programmability of SDN enabled switches.

Data Center Network (DCN) holds a pivotal role in a data center, as it interconnects all of the data center resources together. DCNs need to be scalable and efficient to connect tens or even hundreds of thousands of servers to handle the growing demands of Software defined network (SDN). Load balancing problem is one of the major issues in Data centers in their different shapes, whether they are physical Data Centers or virtual Data Centers. Data Centers usually allow multiple paths routing for the purpose of improving the tolerance to faults in addition to increasing network's throughput by means of sorting out the problem of congestion. A Software Defined Network based Openflow Data Center network architecture is used to obtain better performance parameters and implementing traffic load balancing function. In order to increase available bandwidth, maximize throughput, and add redundancy; network load balancing must be used. Network load balancing is the ability to balance traffic across multiple Internet connections. This capability balances network sessions like Web, email, etc. over multiple connections in order to spread out the amount of bandwidth used by each LAN user, thus increasing the total amount of bandwidth available.

In this Research, I proposed dynamic load balancing algorithms that are implemented in SDN controller based on datacenter network topology. I am using a Dijkstra's algorithm to compute the shortest paths of the same length and link cost between nodes based on the hop count. In load balancing algorithm, you can find the best path of each of the link present that can adapt to the topology changes. The performance of the algorithm is compared to the static load balancing algorithm in term of bandwidth, transmission rate and response time.

2. REVIEW OF RELATED WORKS

Several papers deal with the application of SDN in communication networks. The main technical features of SDN are described in [5]. Different models have been proposed to optimize network flow, resolve the network congestion problem by changing paths of flows during flow transmissions and achieve load balancing among different links. Most of load balancing classifications in the research literature are based on the functionality of load balancing algorithms. Based on such classifications, load balancing algorithms have been categorized into static or dynamic, centralized or decentralized, cooperative or non cooperative and etc.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

Static load balancing Algorithm

In the static load balancing algorithm, paths between switched are already allocated before sending of packets, and the paths cannot be changed during data transferring.

In 2017 M. S. G. a. J. P. Nithin Das K.C [20] , a hybrid algorithm of weight round robin (WRR) and honeybee inspired load balancing approach was introduced. The algorithm finds and examines the overloaded and under loaded virtual servers' capacity. Weights are entered based on its capacity, if the load of the server is less than the capacity entered no more checks are done, otherwise check the overloaded servers using the honeybee inspired load balancing algorithm to allocate high priority requests to the under-loaded servers (virtual machines) respectively. As a result, this algorithm presents a better response time as well as data center processing time.

Dynamic load balancing Algorithm

In dynamic load balancing algorithm, the paths can be changed instantaneously based on the network observation due to the load balance of the network.

In 2016 Y. L. Lan, K. Wang, and Y. H. Hsu [4] In SDN-based data center networks for dynamic load-balanced path optimization (DLPO) a multi-link and a single-link algorithm are proposed for data center networks. The multi-link DLPO algorithm balances link loads in the whole network while the single-link algorithm performs hop to hop flow rerouting to avoid highly loaded links.

In 2019, Sikandar Ejaz, Zeshan IQBAL,Peer Azmat Shah, Bilal Haider,Bukhari,Armughan Ali, Farhan Aadil [16] has proposed traffic load balancing mechanism using SDN controller as VNF in SDN–enabled networks. The experiments using Fat-Tree topology as representative data center network infrastructure with OpenDaylight as SDN controller on mininet emulator for load balancing. The proposed system allows the provisioning of a vSDN controller which is act as a VNF service with exact same configuration as original can be added in the same network to balancing load on both controller. They found accurate working of two controllers and a rises in average pinging of hosts, transfer rate and link capacity after load balancing was witnessed to improve in network performance.

Centralized Load balancing algorithm

In centralized load balancing technique all the allocation and scheduling decision are made by a single node. This node is responsible for storing knowledge base of entire Software define network and can apply static or dynamic approach for load balancing.

In 2014, Gulsan Soni et al. [21] have proposed a 'central load balancer (CLB)' a load balancing algorithm to balance the load among virtual machines having different hardware configurations and states in cloud data centre. Every request form user arrives at Data centre controller. Data centre controller queries the CLB for, allocation of requests. CLB maintains a table that consists of id, states and priority of virtual machines. CLB find out highest priority virtual machine, then checks its states and if its state available then return that VMid to data centre controller. If the states of virtual machine is busy then it chooses next high priority virtual machine. Finally data centre controller assigns the request to that VMid that is provided by CLB. They considered two cases in first load are kept constant and the number of virtual machines varied. In second case, number of virtual machines are kept constant and the load is increased through alter data size per request. In both cases they get less response time.

3. ALGORITHM APPLIED TO SDN

In order to describe the algorithm, first it is needed to disclose the different data structures involved on it. Such structures characterize the different elements that have been taken in account in order to achieve an efficient load balancing, at the same time that to reduce as much as possible the computational cost and time . Since the algorithm provides load balancing based on flows and path. The algorithm that performs load balancing in a Data Center network topology depending on the minimum transmission cost of links at the given time is proposed for balancing the load in this network. The REST API is used to collect operational information of the topology and devices as well as instantaneous traffic and port statistics. Since Data center network the topology presents a high multipath capability, the Dijkstra's algorithm is employed to find multiple paths of the same length and reduce the search to a small region of the topology. Among selected paths, the path with least load is selected and traffic flow is forwarded on that route. The new flows rules are therefore pushed to Open Virtual switches switch (OVSs) in order to update switch forwarding tables. The first step of the algorithm is to collect operational information of the topology and its devices. Such as IPs, MAC addresses, Ports, Connections.

Next step is to find route information based on Dijkstra's algorithm, the goal here is to narrow the search into a small segment of the topology and to find the shortest paths from source host to destination host. And then find total link cost for all these paths between the source and destination hosts. Once the transmission costs of the links are calculated, the flows are created depending on the minimum transmission cost of the links at the given time.

Based on the cost, the best path is selected and static flows are pushed into each switch in the current best path with that, every switch within the selected path will have the necessary flow entries to carry out the communication between the two end points. Finally, the program continues to update this information every minute there by making it dynamic. The performance of the algorithm is evaluated in a fat tree datacenter topology by collecting operational data of the links and switches.

Proposed algorithm for dynamic load balancing operation is described as follows:

Algorithm 1: Heuristic for Traffic Load Balancing

Input: T (Traffic Matrix), Network Topology, Link Capacity

Output: Minimizing maximum link utilization path allocation of flows in T

For all possible flow f in T do

List all possible paths from f_{src} to f_{dst}

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

$list_p$ =Apply the Dijikstra's algorithm to find multiple paths of minimum length

for all path p in $list_p$ **do**

list $_{maximum\ link\ utilization}$[P] = maximum link utilization of p

end for

Pselected = listp[index of minimum in list maximum link utilization]

Assign f to $p_{selected}$

do for all link l in $p_{selected}$ update flow switch table

end for

end for

4. SIMULATION TOOLS PROCESS

The aim of the research is to apply load balancing and improve traffic management in the data center network when congestion occurs. To perform the load balancing experiment ubuntu operating system is used. SDN controller used is Floodlight controller and mininet is used to generate network topology. Mininet is connected to the floodlight controller through the ip address of controller and port 6653 and run our topology to generate the data center topology shown in figure 1, "Ping all" command is executed to check the reachability of all hosts in the defined network. Running network traffic is captured using wireshark and iperf is employed to test network performance. Python programming language implemented for developing Network topology and the algorithm. The simulated network topology consists of 2 core switches at core level, 4 distribution switches at distribution level, 4 access switch and 8 computers that can be viewed using REST API interface as shown in figure1. To perform load balancing test first, the network performance with normal packet transmission is observed first, and call this phase as 'before load balancing (BB)'. Then apply the load balancing algorithm and call this phase as 'after load balancing (AB)'. Again the network performance is observed to verify that the algorithm is capable of traffic management.

The experiment is conducted at Distribution level before load balancing and after load balancing. Also it is conducted at core level before load balancing and after load balancing. Finally the result obtained is analyzed. The result of the experiment is discussed in terms of Qos service measurement done.

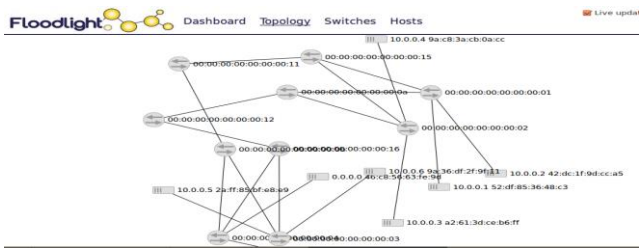


Figure 1: Simulated network topology

5. SIMULATION RESULT

5.1. Performing load balancing at Distribution level

In this case the load balancing is done for the communication between hosts at a distribution level. The load balancing can run between any hosts in the defined network. For this experiment pc1 is choose as source. pc3 and pc4 is chosen as destination hosts. For the experiment the ping command is transmitted from pc1 to pc3 and pc4.The path taken by switch s1 to forward traffic from pc1 to pc3 and pc4 is captured by wireshark and also throughput and response time is tested. The test is done before load balancing (BB) and after load balancing (AB)

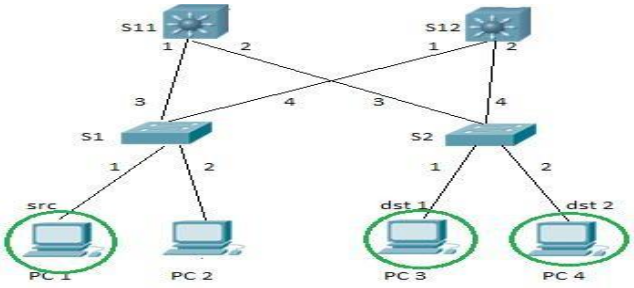


Figure 2: Distribution level test scenario

A. Distribution level testing before load balancing

Ping command is sent from pc1 to both pc3 and pc4 from the mininet terminal then after a wireshark is started to capture ICMP packets send from pc1 to pc3 and pc4. The result of wireshark capture show that both traffics is passing through s1-eth4 while there is no traffic on s1-eth3 which mean that there is no load balancing among the links.

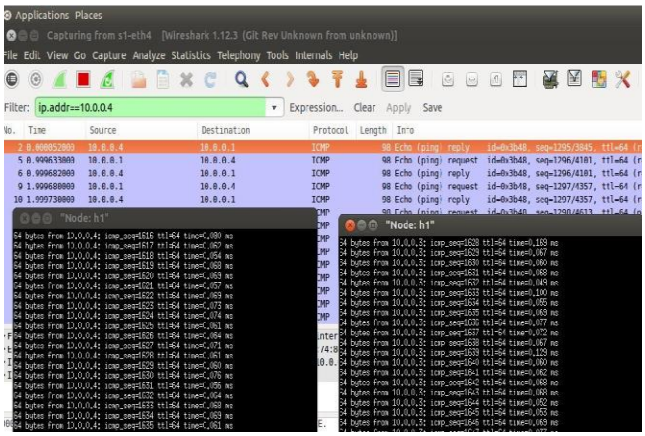
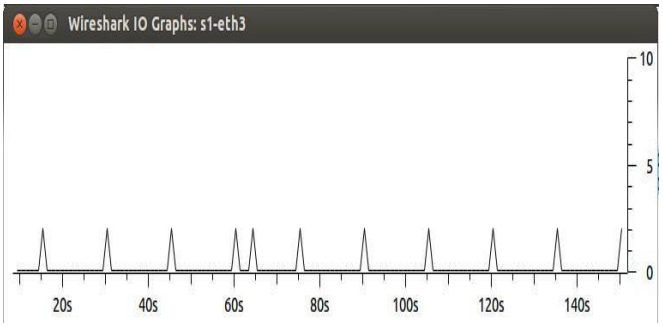


Figure 3: Wireshark capture traffic on s1-eth4 that go to pc4

Wireshark IO Graph output shows the throughput from pc1 to pc3 and pc4 are routed through s1-eth4 only, while no flow is found in the s1-eth3 as shown in the figure 4. The small traffic pics shown in s1-eth3 correspond to the LLDP (link layer discovery protocol) packets sent frequently from the rest API to collect statistics.



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

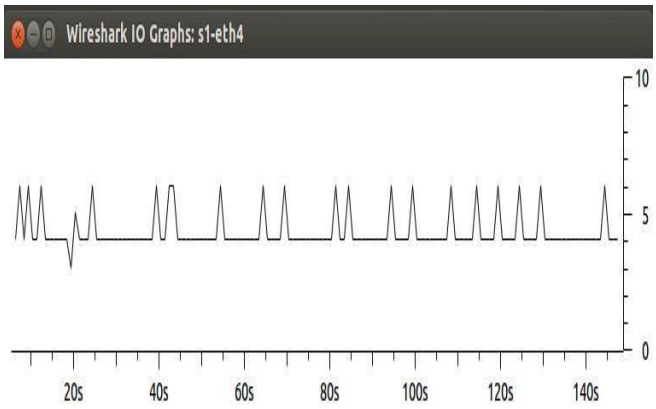


Figure 4: Wireshark io graph showing traffic on s1-eth4 and no traffic on s1-eth3

I. Quality of Service Testing at Distribution level before load balancing

The network was tested before running the load balancing algorithm. The testing focused on some of Quality of service (QoS) parameters such as Bandwidth, Transfer and response time. This is tested by using iperf and maintaining ping command between hosts.

Result obtained from testing Quality of service before load balancing at distribution level are described in the table below.

No.	Transfer(GBytes)	Bandwidth(GBits/sec)	Response time
1	10.5	9.0	0.32
2	12.01	11.0	0.474
3	12.5	10.95	0.323
4	11.35	9.51	0.465
5	11.55	10.01	0.67
6	12.93	11.23	0.487
7	11.1	9.93	0.526
8	10.3	9.1	0.323
Average	**11.53**	**10.09125**	**0.449**

Table 1: Result at distribution level before load balancing

B. Distribution level testing after load balancing

This time the load balancing algorithm written in python is run in the terminal where we run the floodlight controller. The test is to see whether it is able to find alternate best paths when the initial path from source to destinations got congested.

Ping command is done from pc1 to both pc3 and pc4 from the mininet terminal then after a wireshark is started to capture ICMP packets send from pc1 to pc3 and pc4. The result of wireshark capture show that traffic from pc1 to pc3 is passing through s1-eth3 while traffic from pc1 to pc4 is passing through s1-eth4 which mean that there is load balancing among the links.

This time Wireshark IO Graph output shows there is traffic on both interfaces of switch one which mean that load is balancing is done.

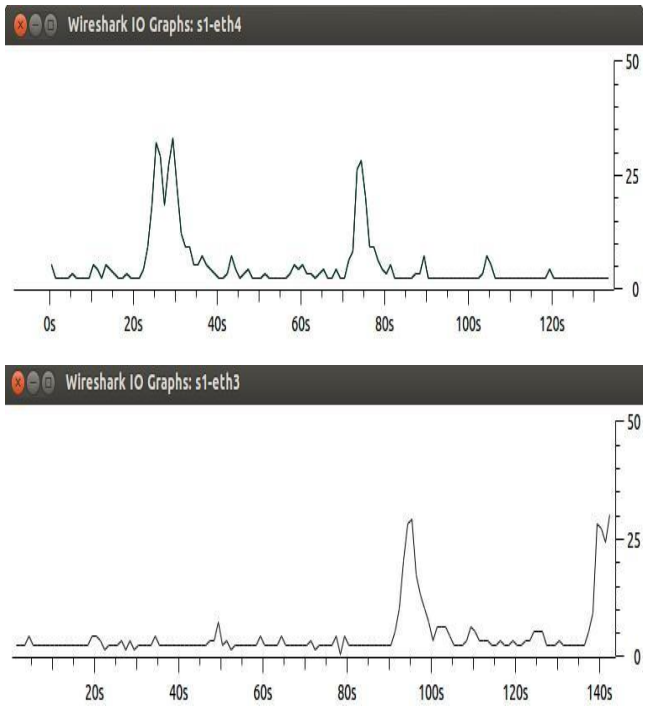


Figure 5: Wireshark io graph showing traffic on s1-eth4 and s1-eth3

II. Quality of Service Testing at distribute level After Load balancing

The network was tested again after running the load balancing algorithm. Bandwidth, Transfer and response time is tested using the same procedure done before load balancing.

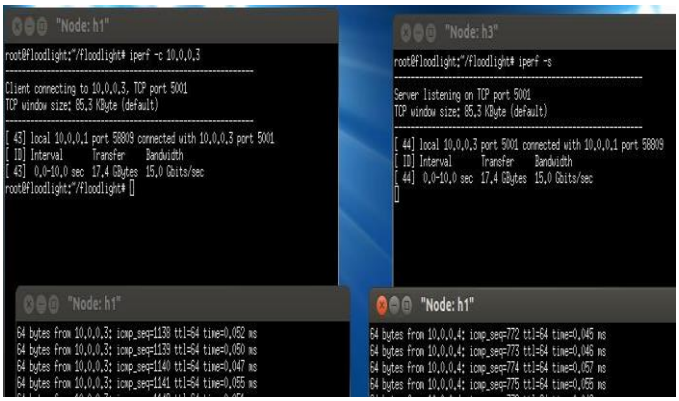


Figure 6: QoS test after load balancing at distribution level

Result obtained from testing Quality of service after load balancing at distribution level are described in the table below.

No.	Transfer(GBytes)	Bandwidth(GBits/sec)	Response time
1	17.4	15.0	0.131
2	18.5	16.2	0.165
3	17.0	15.1	0.09
4	16.4	15.0	0.10
5	18.1	16.01	0.147
6	18.2	16.3	0.14
7	16.5	14.0	0.121
8	16.8	14.2	0.122
Average	**17.3625**	**15.22625**	**0.145**

Table 2: Result at distribution level after load balancing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

C. Discussion of distribution level test result

The network performance test results before load balancing and after load balancing have the difference in terms of response time and throughput. Average of transfer rate, bandwidth and response time before load balancing is 11.53, 10.09 and 0.449 respectively. While average of transfer rate, bandwidth and response time after load balancing is 17.36, 15.23 and 0.144 respectively. The network performance is improved after load balancing which result in transfer rate improved with 50% and bandwidth improved with 50 %. Response time also improved by average from 0.449 to 0.144.

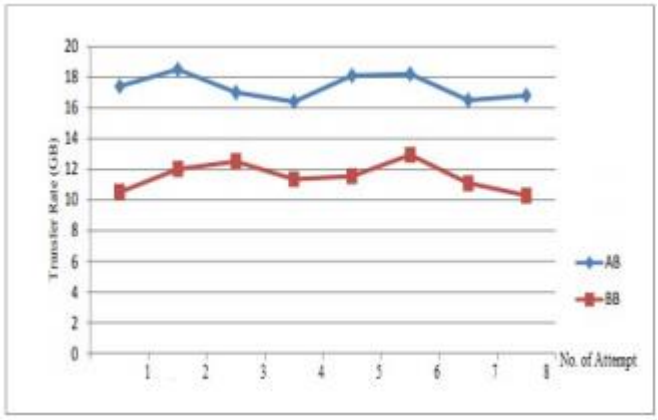


Figure 7: Transfer rate comparison at distribution level

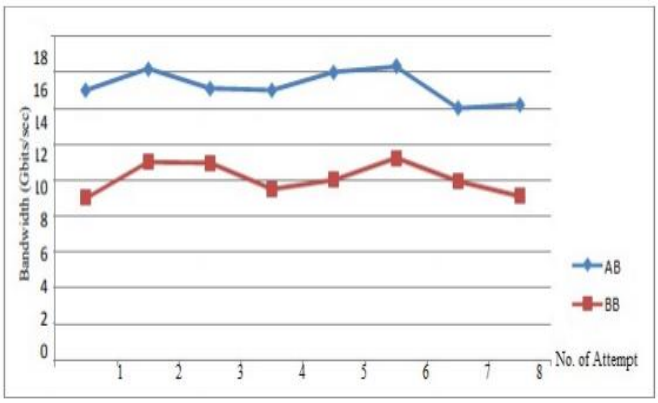


Figure 8: Bandwidth comparison at distribution level

5.2 Performing Load balancing at core level

In this case the load balancing is done for the communication between hosts at a core level. For this experiment pc1 is choose as source. Pc5 and pc6 is chosen as destination hosts. For the experiment the ping command is transmitted from pc1 to pc5 and pc6. The test is done before load balancing (BB) and after load balancing (AB).

A. Core level testing before load balancing

Ping command is done from pc1 to both pc5 and pc6 from the mininet terminal then after a wireshark is started to capture ICMP packets send from pc1 to pc5 and pc6. The result of wireshark capture show that both traffics is passing through the same interface while there is idle interface which show that there is no load balancing among the links.

I. Quality of service testing at core level before load balance

The testing focused on some of Quality of service (QoS) parameters such as Bandwidth, Transfer and response time. This is tested by using iperf and maintaining ping command between hosts.

Result obtained from testing Quality of service before load balancing at core level are described in the table below

No.	Transfer(GBytes)	Bandwidth(GBits/sec)	Response time
1	9.80	8.41	0.642
2	8.79	7.2	0.367
3	9.32	8.1	0.834
4	8.3	7.0	0.365
5	8.5	7.1	0.323
6	9.54	8.32	0.552
7	9.12	8.20	0.456
8	8.23	7.2	0.587
Average	**8.95**	**7.69**	**0.516**

Table 3: Result at core level before load balancing

B. Core level testing after load balancing

Ping command is done from pc1 to both pc5 and pc6 from the mininet terminal then after a wireshark is started to capture ICMP packets send from pc1 to pc5 and pc6. The result of wireshark capture show that traffics is passing through different interface as shown in the figure below.

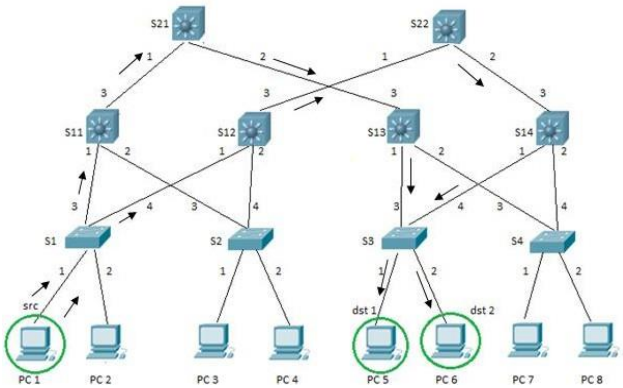


Figure: 9 Traffic paths after load balancing

II. Quality of service testing at core level after load balancing

The network was tested again after running the load balancing algorithm. Bandwidth, Transfer and response time obtained from the test is described below.

No.	Transfer(GBytes)	Bandwidth(GBits/sec)	Response time
1	11.85	10.81	0.435
2	12.91	9.8	0.23
3	11.76	9.7	0.461
4	11.95	9.2	0.411
5	10.96	11.83	0.552
6	12.75	10.45	0.514
7	12.55	10.30	0.487
8	10.90	9.75	0.423
Average	**11.95**	**10.23**	**0.439**

Table 4: Result at core level after load balancing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

C. Discussion of core level test result

The network performance test result before load balancing and after load balancing has the difference in terms of response time and throughput. Average of transfer rate, bandwidth and response time before load balancing show on table is 8.95, 7.69 and 0.516 respectively. While average of transfer rate, bandwidth and response time after load balancing is 11.95, 10.23 and 0.439 respectively.

The network performance is improved after load balancing which result in transfer rate improved by with 33% and bandwidth improved with 33 %. The average response time improved from 0.516 to 0.439

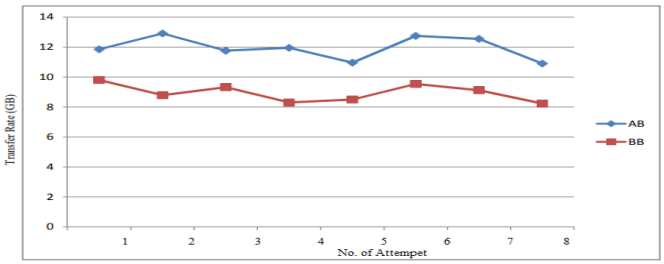


Figure 10: Transfer rate comparison at core level

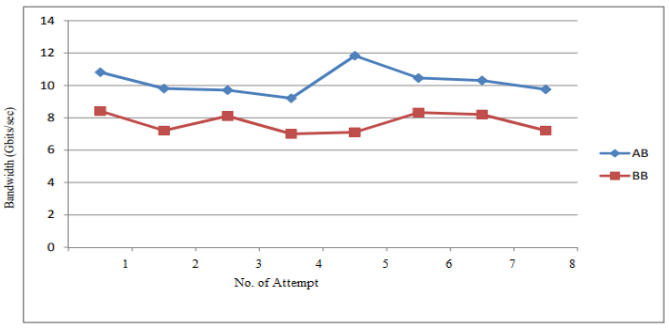


Figure 11: Bandwidth comparisons at core level

6. CONCLUSION

Floodlight controller based dynamic load balancing is implemented in data center network which has two alternative paths from source to destination. The network performance is tested before and after implementing load balancing to test quality of service in selected type of data center network.

The result of performance test done at distribution and core level before and after implementing load balancing is analyzed. The network performance shows improvement after implementing load balancing in terms of quality of service test done.

This thesis describes the implementation of a dynamic load balancing algorithm to distribute the different traffic flows carried by a network through the different parallel paths between source and destination. The implementation has been conducted over a Software Defined Network, trying to explore the capabilities that this new paradigm of networking brings to us.

SDN provides a view of each of the elements of the network as well as control over them. Thus, allows a dynamic control of the actions to be done in each possible situation. Regarding the optimization of resources, such control over the network adds a bunch of new functionalities, such as the dynamic routing proposed in this project.

Nowadays, OpenFlow is the most popular protocol for the southbound interface (i.e. to communicate the controller with the network elements), nonetheless, it presents some limitations. One of them is the monitoring of the network, which is involved in this project. OpenFlow, being a control-plane protocol has great difficult accurately determining data plane measurements. The way to do this with OpenFlow entails making periodic flow stats requests to the switches on the network. The problem is that stats are never truly accurate.

By the time they have been processed on the switch, sent over the network, and then processed by the controller they will be out of date. This is one of the problems showed in the emulation section. In addition to that, the stats requests generate a large amount of traffic (even though the traffic between controller and the switches is carried by a dedicated channel, it should be reduces as much as possible), and increase the computational cost in the controller to process these messages. It is because of that in the implementation the stats are updated once per second, which leads to a delay in the rerouting process. As a conclusion about this point, data-plane measurements are best handled by third party applications, thus improving the results of the implementation.

In relation to the load balancing, the proposed algorithm works well for non-priority traffic, with the objective of reroute the traffic flows which are carrying less traffic. Based on that principle, it has been shown how it is possible to reduce congestion and simultaneously enhance network resource utilization.

7. FUTURE WORK

Even though the implementation has been designed to be adapted to hierarchical topology, a further analysis with different topologies of different sizes should be made so could be guaranteed that it has no limitations on that aspect. A deepest analysis with a larger number of flows, with different kinds of traffic patterns is also needed.

Testing this floodlight controller based dynamic load balancing for different type of network structure to test performance of this proposed method to trouble if it has some drawback. Comparing floodlight controller based dynamic load balancing with other software defined controller based load balancing to test performance.

8. REFERENCES

[1] Martí Boada Navarro. “Dynamic Load Balancing in Software-Defined Networks”. Aalborg University, Department of Electronic Systems, Fredrik BajersVej 7B, DK-9220 Aalborg. June 2014.

[2] H. Long, Y. Shen, M. Guo and F. Tang, “LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks,” in*Proc. Inter. Conf. on Advanced Information Networking and Applications*, Barcelona, Spain, Mar 2013, pp. 290-297.

[3] Y. L. Lan, K. Wang, and Y. H. Hsu, “Dynamic load-balanced path optimization in SDN-based data center networks,” in *Proc. IEEE Inter. Conf. on Comm. Sys, on Networks and Digital Signal Processing*, Prague, Czech Republic, July 2016, pp. 1-6.

[4] Y. L. Lan, K. Wang, and Y. H. Hsu, “Dynamic load-balanced path optimization in SDN-based data center networks,” in *Proc. IEEE Inter. Conf. on Comm. Sys, on Networks and Digital Signal Processing*, Prague, Czech Republic, July 2016, pp. 1-6.

[5] R. Masoudi and A. Ghaffari, “Software defined networks: A survey,” *J. Network and Computer Applications*, vol. 67, pp. 1-25, May 2016..

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

[6] Senthil Ganesh N and Ranjani S. “Dynamic Load Balancing using Software Defined Networks”. International Journal of Computer Applications (0975 –8887), International Conference on Current Trends in Advanced Computing (ICCTAC-2015), May 2015.
[7] Project Floodlight, Floodlight. (2012). from http://floodlight.openflowhub.org/
[8] Mininet: An Instant Virtual Network on Your Laptop (or Other PC) [online]. Available: http://www.mininet.org/.
[9] iperf - The TCP, UDP and SCTP network bandwidth measurement tool [online]. Available: https://www. iperf.fr/.
[10] Guido van Rossum. “An Introduction to Python for UNIX/C Programmers”. Proceedings of the NLUUG najaarsconferentie. Amsterdam, Netherlands, 1993.
[11] http://mininet.org/overview/
[12] (2012). Inter-datacenter wan with centralized the using sdn and openflow. White paper, Google, Inc.
[13] Richard Wang, J. R., Dana Butnariu. (2011). Openflow-based server load balancing gone wild. Princeton University.
[14] M.I. Hamed, B.M. ElHalawany, M.M. Fouda, A.S. Tag Eldien, "A New Approach for Server-based Load Balancing Using Software-Defined Networking", Proceedings of the 2017 IEEE International Conference on Intelligent Computing and Information Systems (ICICIS 2017), December 5–7, 2017.
[15] Floodlight OpenFlow Controller [online]. Available:
[16] Sikandar Ejaz, Zeshan IQBAL,Peer Azmat Shah, Bilal Haider,Bukhari,Armughan Ali, Farhan Aadil, “Taraffic load Balancing using Software Defined Networking(SDN) controller as virtualized network function” IEEE, Vol 7, 2019.
[17] Mavjeen and Adrashbi ‘ Load Balancing Utilization in Data Center Networks’ Student Symposium 2015, San Jose State University, California, Spring 2015.
[18] Shavan K. Askar ‘Adaptive Load Balancing Scheme For Data Center Networks Using Software Defined Network’ Electrical and Computer Engineering Department, Collage of Engineering, University of Duhok, Kurdistan Region-Iraq. Vol. 4 , No.2, Pp 275-286, 2016.
[19] J. Saisagar, Prashant Kothari D., ‘ Sdn Enabled Packet Based Load-Balancing (Plb) Technique In Data Center Networks’ Department Of Computer Science Engineering, Srm University, Vol. 12, No. 16, August 2017.
[20] M. S. G. a. J. P. Nithin Das K.C, "Incorporating Weighted Round Robin in Honeybee Algorithm for Enhanced Load Balancing in Cloud Environment," International Conference on Communication and Signal Processing, April 6-8, 2017, India, 2017.
[21] Gulshan Soni, Mala Kalara, “A novel approach for load balancing in cloud data centre”, 2014, International Advance Computing Conference , IEEE
[22] Pooja Samal, Pranati Mishra, ”Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing”, International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 416-419