International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 3279~3285
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp3279-3285

Path based load balancing for data center networks using SDN

V. Deeban Chakravarthy, B.Amutha
Department of Computer Science and Engineering, SRM Institute of Science and Technology, India

Article Info

Article history:

Received Dec 16, 2018
Revised Mar 27, 2019
Accepted Apr 9, 2019

Keywords:

Load balancing
OpenFlow
SDN
Software-defined network

ABSTRACT

Due to the increase in the number of users on the internet and the number of applications that is available in the cloud makes Data Center Networking (DCN) has the backbone for computing. These data centre requires high operational cost and also experience the link failures and congestions often. Hence the solution is to use Software Defined Networking (SDN) based load balancer which improves the efficiency of the network by distributing the traffic across multiple paths to optimize the efficiency of the network. Traditional load balancers are very expensive and inflexible. These SDN load balancers do not require costly hardware and can be programmed, which it makes it easier to implement user-defined algorithms and load balancing strategies. In this paper, we have proposed an efficient load balancing technique by considering different parameters to maintain the load efficiently using Open FlowSwitches connected to ONOS controller.

Corresponding Author:

V. Deeban Chakravarthy,
Department of Computer Science and Engineering,
SRM Institute of Science and Technology,
SRM Nagar, Kattankulathur, Tamil Nadu 603203, India.
Email: vdeeban@gmail.com

1. INTRODUCTION

With increasingly transmitted data in data center networks, the traffic exchanged between data center network switches grew rapidly, which could lead to congestion and lead to a long delay and low performance in data center networks. This problem of congestion needs to be addressed urgently using an efficient path optimization algorithm to improve the performance of the data center networks.

Load balancing is widely implemented to distribute load on different resources, it is used to optimise the network and prevent congestion. Load balancing is one of the crucial issues, which divides the workload dynamically among the servers by improving the performance of the system [1]. There are two types of network load balancing techniques namely server and link. The server load balancer sends the load to the other servers which are just replica of main server.This approach is very costly and it has scalability issues too. Another approach is link load balancing in this we join multiple physical links with virtual links for transmitting the data.MPLS Traffic Engineering is an another approach of link load balancing, but it is very complex because every host and switch in the tunnel is configured first and every device directs the workload to the path .MPLS TE is not usually deployed because of it's complexity.Above mentioned models are useful but they are not proper solution for every problem.

A path connecting the source and the destination node includes numeruos switches and links to connect those nodes. The traditional path load balancing protocol transfers data by path first, sometimes this create congestion on single node.Path load balancing can be great overcome of this problem in which we can use SDN and distribute the workload by using different parameters.

SDN [2] is an approach to provide network management and to increase the efficiency of network through programs.SDN has logically centralized the intelligence in the network [3] by disconnecting the further process of packets, that intelligence is the algorithms used in the controller which controls the routing process of packets.

OpenFlow [4] is the main technology that is used in SDN, it is used to provide communication between SDN controller and and other forwarding layer, it maintains the flow of network traffic. SDN controller optimise the flow of network traffic by making decisions on the path links. Different SDN controllers such as ONOS, POX [5], RYU, NOX [6], Floodlight [7] etc.We are using ONOS [8] controller in our experiment.

Mininet [9] is used for creating virtual topology network and transfer packets to different switches,onos controller is used to control the network and to show the gui of the topology and how the packets transfers from one host to another. In order to solve the congestion problem on a data center network, network administrators can therefore install efficient path optimization algorithms with OpenFlow protocol to control paths of new flows and change paths of streams during their transmissions. SDN-based path optimization algorithms are able to route flows from a data center network according to the link load, because the centralized SDN controller is able to obtain load statistics for each data center switch.

SDN is the best approach for path load balancing, this paper presents an effective technique for path load balancing,in this experiment we uses controller to control the network by using different parameters. Then the controller transmits the decision made by controller to the forward switches.Experimental results shown that this technique has increased the efficiency of the network load balancing.

2. RELATED WORK

In SDN, there has been many studies done in path load balancing. For example, Plug-n-serve [10] system, it minimizes the response time taken by the client to server. When a request comes, the decisions are made by controller and create forwarding rules that handle the request of a client from a server. Plug-n-serve system has two main advantages flexibility and reduced response time but it has the biggest disadvantage of scalability, it has scalability issues. The more scalable solution has been provided by Wang et. al [10]. This algorithm computes the rules of wildcard for reducing the dataload on SDN controller. It has two algorithms “partitioning algorithm” and “transitioning algorithm”, partitioning algorithm is used to determine the wildcard rules and “transitioning algorithm” is used to change the rules of wildcard with respect to change of the policy of load balancing. These algorithms are for single network when the network gets distributed, we have to use distributed SDN approach which has given by Koener and Kao [11]. They developed a load balancing technique to handle multiple services by using multiple working OpenFlow. For example, one flow is handling internet browsing and another flow is handling e-mail transferring. This method has increased the efficiency of the network load balancing by distributing the workload to multiple OpenFlows. Hardeep Uppal and Dane Brandon [12] have developed an OpenFlow technology-based load balancer architecture that reduces costs and gives flexibility. It contains three random load balance policies to randomly select the registered servers, a round robin load balance policy to rotate the registered servers for the purposes of serving the load-based load balance algorithm, to select the server with the lowest possible load. This method can have scalability issues due to constant change and increased services. Chou et. al [13]. developed a genetic algorithm load balancing system which pre-decide the rules to transmit the data to the switches in advance.

In the context of inter-DC WAN, B4 and SWAN [14, 15] are using SDN. By coordinating the sending rate of services and centrally configuring the network data plane, these enable effective interDC wANs to be transmitted using a max-min-method. Although B4 develops custom switches and processes to integrate current routing protocols within the SDN environment. Long et al. [16] propose a loadbalance routing algorithm for OpenFlow-enabled datacenter networks (LABERIO) to solve the static routing path problem. The centralized controller will be notified and help schedule the greatest flow in the busiest hop with the maximum remainder capacity strategy if the link load detector exceeds the LABERIO trigger threshold. However, all the schemes we have described above rely on traffic demand information, but for many networks traffic demand can not be known in advance. All the approaches we discussed earlier is rely on demand of traffic but it is not sufficient to get information about network in before. In this research we don’t use traffic demand instead of that we use different parameters to know the network in advance

3. PATH BASED LOAD BALANCING ALGORITHM

To improve the cluster performance, the load balancing algorithm which will run on load balancer plays a significant role. Distributed System is important to distributing the work load on the servers [17]. Load Balancer places a vital role in fulfilling the request of the clients through servers and for this work load balancer routes requests to those servers, which has the capability of doing its job in an effective way that is maximization of speed, maximum utilization of capacity and can fulfill the client’s requests [18]. Our solution follows the layer architecture of SDN, in which there are two planes, control plane and data

plane, Figure 1 is showing the architecture of SDN. The paths which transfer workload from source to destination should be optimal, this is done by ONOS controller. The controller makes decision on the basis on path evaluator which selects the path which are optimal for path load balancing. It contains data collector method and path evaluator method.

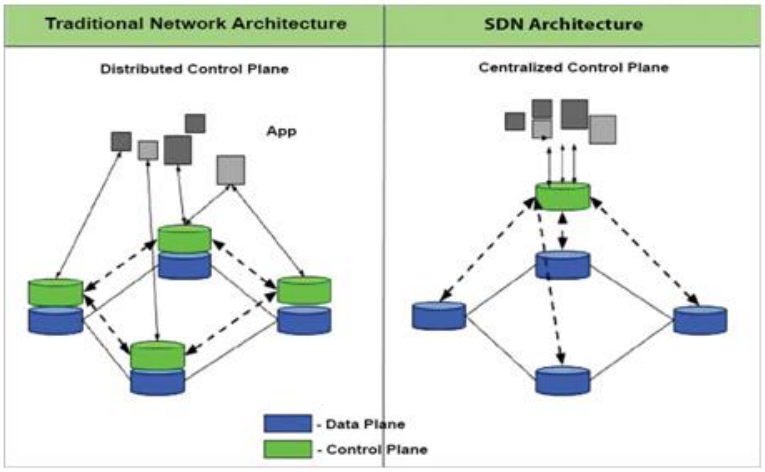

Figure 1. SDN Load balancing architecture

3.1. Data collector method

– Topology link collector

Topology link collector gets the information between the links of the topology, it is done by transmitting LLDP data packets to every node.

– Traffic data collector

Since we are using different parameters for making decisions for path load balancing, we need to collect the data about those parameters. We are using byte count, packet count forwarding rate and duration of transmitting between source node and destination node.

– Path evaluator method

This method contains Top-K path algorithm and evaluator method.

– K Path algorithm

There are many paths in the network for transmitting packets from source to destination. Those paths are used to put in the Algorithm 2, that has been provided below. This is done in two different variants one is start and another is interval. In the start variant, we use the Flyod's algorithm [19] to find the top-k paths. In the interval variant, we use the data collected by traffic data collector to get optimal path for transmitting the data. Algorithm 1 shows the k path algorithm.

Algorithm 1: K path collector

Start mode: Getting K paths
We assume that E be the source and G be the destination node, in between these nodes we have to find k paths.
M[r] is the array in which the paths will be stored and the first path is M[0]
1.M[0] represents path in nodes E to G
2.For n in {1,2,…,N-1}
3. For a in {0,1,2,…,length(M[n-1])-2}
4. *nodeP = P[n-1].node(a)*
5. For b in {0,1,2,…,n-1}
6. *nodeQ = P[b].node(i*)
7. If nodeP = nodeQ
8. The distance between nodeP to M[b].node(a+1) = infinite
9. EndIF
10. EndFor
11. E[a] represents the path in nodes nodeP and G

12. R[a] will be found in M[k-1] from E to nodeP
13. *Q[a] = R[a] + E[a]*
14. EndFor
15. The minimum path in Q[a] is M[r]
16. Again store the initial value of distance between nodeP to nodeQ
17. EndFor

3.2. Interval mode

Take k paths from the above algorithm and use them in algorithm 2 and after every 5 minutes take another new k paths.

– Evaluator Algorithm

The evaluator algorithm is used to evaluate the k path which has been found by Algorithm 1. The parameters of evaluation are byte rate, packet count, port forward rate and duration between transmission of packet from one host terminal to another. Algorithm 2 shows the algorithm for evaluation of top-k path.

Algorithm 2: Evaluation of Paths
Input: The output paths found in algorithm 1 is K
Output: the best optimized path is OPTIMAL_PATH

1. Define function factor F (byte_count, packet_count, Flow_rate, duration) and define the rank function R[N] for path.
2. For i in {0, 1, 2, ..., K-1}
3. *byte_count[i] = Byte count in K[i]*
4. *packet_count[i] = Packet count in K[i]*
5. *Flow_rate[i] = Flow rate in K[i]*
6. *Duration[i] = Duration in K[i]*
7. EndFor
8. For each j in {0, 1, 2, ..., K-1}
9. *byte_count[j] = 1.0 / log(byte_count[j] + 0.1)*
10. *packet_count[j] = 1.0 / log(packet_count[j] + 0.1*)
11. *Flow_rate[j] = 1.0 / (1 + exp (Flow_rate[j]/100.0))*
12. *Duration[j] = 1/Duration[j]*
13. EndFor
14. Define weightmatrix W (byte_count_weight, packet_count_weight, Flow_rate_weight, duration_weight)
15. For k in {0, 1, 2, ..., K-1}
16. *t_byte_count[k]= byte_count_weight + byte_count[k]*
17. *t_packet_count[k]= packet_count_weight+packet_count[k*]
18. *t_Flow_rate[k] = Flow_rate_weight and Flow_rate[k]*
19. *t_Duration[k] = duration_weight + Duration[k]*
20. *value_score[k] = t_Duration[k] + t_bytes[k] + t_packet_count[k] + t_Flow_rate[k]*
21. EndFor
22. Whichever path gets the maximum value_score is the OPTIMAL_PATH

4. SIMULATION AND ANALYSIS

We have used Mininet [20, 21, 22] and the ONOS platform [23, 24] to simulate the network topology mentioned in the proposed technique. ONOS is a platform, which capitalizes on the resources of JAVA for the rapid configuration and prototyping of network controllers. Mininet assists users in quick formation of virtual networks, Connect connectors, switches, and local networks to a single server. Providing a lightweight evaluation platform for configuring OpenFlow applications, mininet is running on Virtual Box in which the host is linux. The performance of Mininet is depended on the capability and configuration of the version of linux. The specification of the host is listed down in Table 1.

Table 1. Specification of host

Parametets	Values
Physical Machine	Intel Core i7-2100 3.1GHz processor 8GB memory
Operating System	x86_64-with-Ubunu14.10
Virtual Machine	Virtual Box

In order to create useful results, we have tried to test the technique and the network for a proven and robust overall efficiency. For network verification capability and ability, the simulation environment has been designed with 6 OpenFlow switches topology and 8 host users. As shown in the Figure 2, the switch indicates an OpenFlow switch; it's switch ID. The host is a connected host simultaneously and its host ID is. In the meantime, the solution proposed to balance the load is also analyzed to solve the ways of understanding the differences.

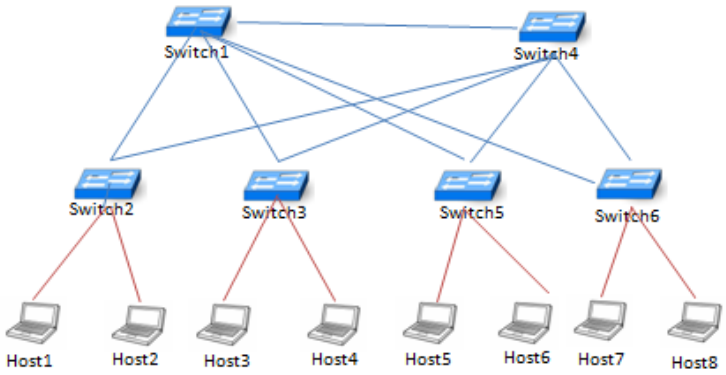

Figure 2. Topology of simulation

4.1. Effectiveness

To deduce the value of the evaluation module the efficiency test has been carried out under dynamic changes. The Host1 and Host8 are selected to be the entity under observation. Firstly, we iteratively figure out the selection of three iterative paths. Screening the dynamic status of the network, based on requirements, the three paths are then evaluated using the fuzzy evaluation model. Initially, the value to evaluate a path is found. To know the workload in Switch6 and Switch4 a packet will be sent by Host7. To get the resources of bandwidth in between Switch1 and Switch4, Host6 sends a packet to Host2. When the link status changes the evaluation value of three paths changes too. For first path, its one link and 2 switches are impacted and they have no obstacle on the, so its value of evaluation decreased maximum. For third path, Switch4 is implicated and caused obstruction, so the value of evaluation decreased but a less than the others. The results have been shown in Figure 3. Later on, we will get the optimal path to transmit the load for acquiring the load balancing in the network. Demonstrating effectiveness, the presented technique also promises to dynamically alter paths with the adequate traffic status the links and nodes.

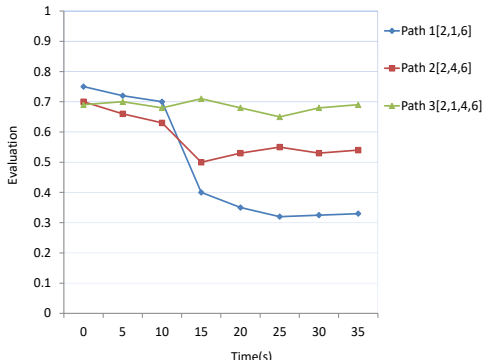

Figure 3. The evaluation of paths

4.2. Analysis of efficiency

To understand the proposed efficiency level, the k-paths algorithm is considered and further analysis is done. Host2 and Host9 are selected as the subject of observation in the 9 openFlow open network topology and 10 host terminals, that is represented in Figure 4. As experimental result the RTT has been tested, this has shown in Figure 5. It shows that RTT fluctuates violently, while the path becomes more filled with the path algorithm. However, RTT does not fluctuate with our solution, because the traffic is transmitted to the optimal route in time. The results ensure the proposed algorithm of load-balancing to be optimized enough to guarantee sanity in packet transmission and also dynamic management of the paths.

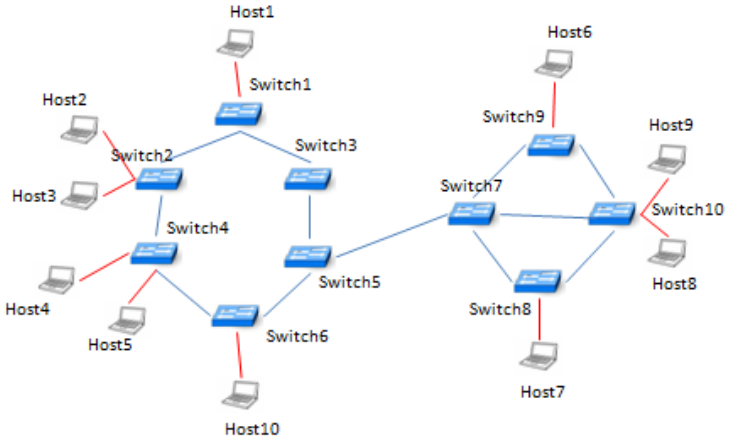

Figure 4. Topology for comparison

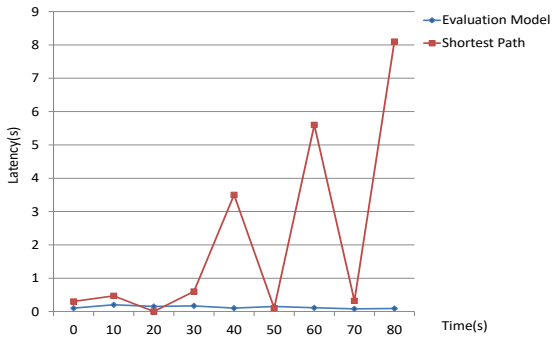

Figure 5. RTT of the proposed solutions

5. CONCLUSION

With an aim to innovatively propose an SDN-based path load-balancing technique, this paper presents how it improves the current rates of efficiency, reliability and utilization. The fuzzy synthetic evaluation model helps in the dynamic and optimized selection of a path, as a reflex to the variation in the network traffic. The implementation of the solution is supported using the ONOS platform, whereas the verification and proof of the results obtained is backed by Mininet. The graphical representations conclude that this technique can dynamically not only avoid failures and faults in the path but also adjust the transmitting path. The result seems more accurate with the overall performance of the network being increased.

REFERENCES

[1] T. Sasidhar, V. Havisha, S. Koushik, M. Deep, VK. Reddy, "Load Balancing Techniques for Efficient Traffic Management in Cloud Environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 3, pp. 963-973, 2016.

[2] "Software-Defined Networking: the new norm for networks," White Paper, Open Networking Foundation, 2012.

[3] Haleplidis, E., Denazis, S., Koufopavlou, O., Halpern, J. and Salim, J.H., "Software-defined networking: Experimenting with the control to forwarding plane interface," *In Software Defined Networking (EWSDN)*, 2012 European Workshop on (pp. 91-96). IEEE, October, 2012.

[4] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J., "OpenFlow: enabling innovation in campus networks". *ACM SIGCOMM Computer Communication Review*, 38(2), pp.69-74. 2008.

[5] POX,https://openflow.stanford.edu/display/ONL/POX+Wiki

[6] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N. and Shenker, S., "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, 38(3), pp.105-110. 2008.

[7] Floodlight, http://www.projectfloodlight.org/floodlight/

[8] ONOS, https://wiki.onosproject.org/display/ONOS/ONOS+from+Scratch

[9] Mininet, http://mininet.org/

[10] Koerner, M. and Kao, O., "Multiple service load-balancing with OpenFlow," *In High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference*, (pp. 210-214). IEEE, June, 2012.

[11] Wang, R., Butnariu, D. and Rexford, J., OpenFlow-Based Server Load Balancing Gone Wild. Hot-ICE, 11, pp.12-12, 2011.

[12] Uppal H, Brandon D, "OpenFlow based load balancing," *In Proceedings of CSE561: networking. project report. University of Washington, Spring*, 2010.

[13] Chou, L.D., Yang, Y.T., Hong, Y.M., Hu, J.K. and Jean, B., "A genetic-based load balancing algorithm in openflow network," In Advanced Technologies, Embedded and Multimedia for Human-centric Computing, Springer, Dordrecht, (pp. 411-417), 2014.

[14] S. Jain, et al, "B4: experience with a globally-deployed software defined WAN," SIGCOMM, 2013.

[15] Chi-Yao Hong, Srikanth Kandula,Ratul Mahajan,Ming Zhang,Vijay Gill, Mohan Nanduri,Roger Wattenhofer, "Achieving High Utilization with Software-Driven WAN," SIGCOMM, 2013.

[16] Hui Long, Yao Shen, Minyi Guo, Feilong Tang, "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks*,"* In *Proceeding of the 27th international conference on advanced information networking and applications*, pp 290-197, 25-28, March 2013.

[17] S. Potluri and K. Subba Rao, "Quality of Service based Task Scheduling Algorithms in Cloud Computing", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 2, p. 1088, 2017.

[18] Kadiyala Ramana and M. Ponnavaikko, "AWSQ: An Approximated Web Server Queuing Algorithm for Heterogeneous Web Server Cluster," *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 9, No.3, pp. 2083-2093, 2019.

[19] Floyd, R.W., "Algorithm 97: shortest path. Communications of the ACM," 5(6), p.345, 1962.

[20] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M. and Zolla, J., August. "B4: Experience with a globally-deployed software defined WAN," In *ACM SIGCOMM Computer Communication Review*, Vol. 43, No. 4, pp. 3-14, ACM, 2013.

[21] Hong, C.Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M. and Wattenhofer, R., August. "Achieving high utilization with software-driven WAN," In *ACM SIGCOMM Computer Communication Review*, Vol. 43, No. 4, pp. 15-26, ACM, 2013.

[22] Long, H., Shen, Y., Guo, M. and Tang, F., March. "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks," In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference, IEEE, (pp. 290-297).* 2013.

[23] Bellman, R.E. and Zadeh, L.A., "Decision-making in a fuzzy environment," *Management science*, 17(4), pp. B-141. 1970.

[24] Shier, D.R., "On algorithms for finding the k shortest paths in a network," *Networks*, 9(3), pp.195-214. 1979.