IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-5, Issue-12 E-ISSN: 2347-2693

Load Balancing in Software Defined Network

P. Kumari[1*], D. Thakur[2]

[1*] Department of Computer Science, Banasthali Vidyapith, Rajasthan, India
[2] Department of Computer Science, Banasthali Vidyapith, Rajasthan, India

**Corresponding Author: kajal.mishra@gmail.com*

Available online at: www.ijcseonline.org

Received: 14/Nov/2017, Revised: 29/Nov/2017, Accepted: 17/Dec/2017, Published: 31/Dec/2017

Abstract— Legacy heterogeneous network paradigm have reached the extremity where its competence to accustom to dynamic situation has become a hindrance. These traditional network platforms are very complicated to manage due to various challenges like interoperability, upgrade which require protocols and management techniques hard-coded into the underlying network platform. Software Defined Network is an emerging network architecture which allows automation of infrastructure configuration enabling the network operators to adapt their network to meet the real time application requirements. Now-a-days with the frequent and rapid inflation of number of clients associated with the network, it is very important to distribute the incoming request equally among all the servers. So by taking advantage of the overall view of the network by the controller, various load balancing strategies can be used to distribute the request in order to increase the overall performance of the system. Every load balancing technique has some pros and cons. This paper gives the comprehensive critical survey on various load balancing strategies used in SDN technology.

Keywords— Software Defined Networking, Load Balancing, Openflow

I. INTRODUCTION

Dealing with the design and administration of traditional network seems quite difficult as they are equipped with various hardware devices such as routers, switches, and middle boxes like load balancer, network address translator, intrusion detection system, firewalls etc which are non programmable and vendor specific i.e. any alteration as per the requirement can be done only by the vendor which in case of complex network can be challenging and fallible. If there is a need to add or remove any functionality from the network, it cannot be done without tampering the network infrastructure and will directly affect its logic. Also setting up any new protocol has to pass through various testing and standardization to ensure interoperability provided by particular vendor.

Software defined network (SDN) [1] has emerged as the new network framework which is programmable and vendor neutral. It conquer the shortcomings of long-established network architecture by abstracting the main intelligence of network i.e. control plane from forwarding plane (data plane). SDN architecture consists of application plane, control plane and data plane as shown in figure 1.

The centralized controller is the brain/core of SDN network which is responsible for broadcasting information to the switches/routers residing below it via southbound interface and application/business logic residing above it via northbound interface. Any communication between controller and network elements are governed by Open flow protocol [2].

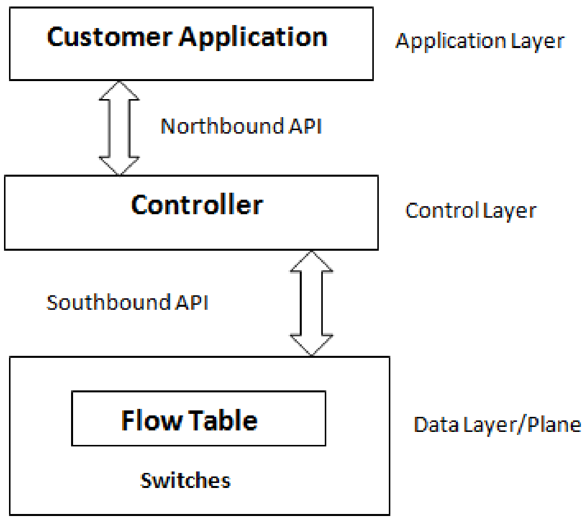

Figure 1: SDN architecture

Data plane consists of switches and these switches have flow table associated with it which is used for decision making.

Controller is responsible to control flow table. Lately the rapid growth of SDN has addressed various research challenges like there may be single point failure due to single centralized controller or sometime due to heavy traffic network may be overloaded which limits the scalability. SDN reduces the complexity involved in managing and developing the application by providing up to date status of the network to the programmer.

Technology has completely changed the way we live our life and web services like Internet is one such technology which has akin us in a way that was hardly plausible as it handles almost all facets of our life like business, entertainment, education, social network, communication etc. With this excellent evolution in computer technology, the demand for high speed, availability, scalability, hasty response has grown. Today Internet serves millions of customers due to which there is raise in web traffic which further leads to network congestion and loss of packets. So Load balancing techniques are used to distribute the incoming load among various servers to prevent the single server from getting overloaded in order to increase the efficiency of the network. The performance of a load balancing algorithm can be evaluated on the basis of various criterions such as resource utilization, response time, fault tolerance, latency. Load balancing decision can be taken either statically or dynamically. Static load balancing algorithm is not able to work according to the real time requirement, so it is more feasible to use the dynamic algorithm.

This paper gives a survey on various load balancing techniques in Software Defined Network (SDN). Some of the paper explains how load balancing will occur in case of single centralized controller and some in distributed controllers with and without Super Controller. Voellmy et al. [3] shows that multiple controller provide high fault tolerant network with reduced latency.

The organization of the papers is as follows, Section I contains the introduction of Software Defined Network , Section II contain the literature survey of load balancing algorithms in Software Defined Network, Section III contain the comparative analysis of algorithms along with the parameters considered, Section IV contain the conclusion and Section V concludes research work with references.

II. LITERATURE SURVEY

Although SDN is widely used network architecture there are various issues such as load balancing that needs to be taken care of. Load balancing is an intelligent congestion cognizant routing in Software Defined Network (SDN). In any SDN based network architecture, load balancing is an indispensable entity to promote the availability and scalability that further leads to attain the minimal response time of application as shown in figure 2.

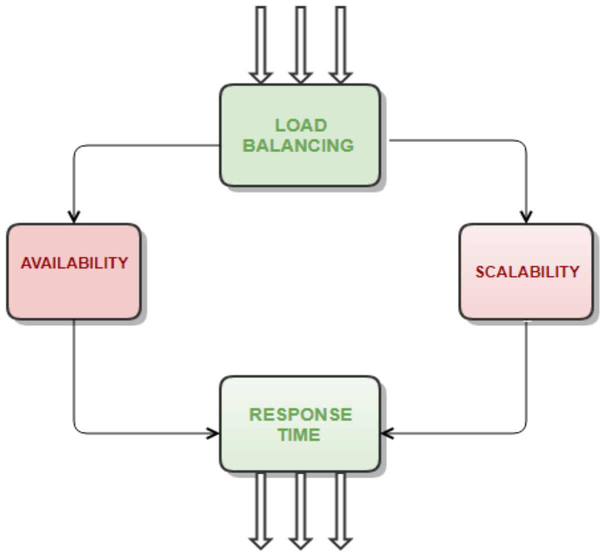

Figure 2: Load Balancing in SDN

The SDN architecture can be either distributed or centralized. In centralized controller architecture there will be single controller responsible for everything which indeed may suffer from single point of failure. Distributed architecture can be either flat or hierarchical. In Flat, all the controllers are on same layer. In hierarchical, controllers are placed on different layers. Compared to centralized architecture, distributed architecture has more communication overhead. The reason is because in order to have the view of complete network all the controller has to communicate with each other about their state and the information they contain. The load balances aims to disseminate the client request among available servers to prevent the single server from being overburdened. In case of server failure, load balancer provides fault tolerant capability by redirecting the request to one of the remaining server. Traditional load balancer uses dedicated hardware (F5 networks, Berracuda Networks, A10 Networks, Kemp technologies [4]) which was vendor locked and non-programmable, so the network administrator cannot change or write new load balancing algorithm which makes them inflexible and non-scalable. The outlay of purchasing any dedicated load balancer is very large. So large number of load balancer will be required if the network is large. This will increase the overall cost of the network. With the advent of Software Defined Network which is programmable in nature, allows to code and implement own load balancing algorithm converting the dumb physical switches into powerful load balancer using SDN Controller. It provides agility to quickly adapt to changes and flexibility to add new trait to existing network architecture as and when required.

The load balancer architecture consists of server pool connected to the load balancing device. The entire incoming request will be directed to virtual IP address of load balancer. Load balancer will overwrite the destination IP with one of the server IP. Different load balancing algorithm will be used to select one of the server. Once the server process the request, reply will be sent back to load balancer where it will change destination IP to Address of host. Different strategies to select the server are discussed below.

The most popular stateful algorithm for load balancing is **Hash-based** [5]. The algorithm works by first calculating the hash value of traffic flow using IP address of source and destination, port number of source and destination and URL. The request is then forwarded to the server with highest hash value. If any other request comes with same hash value, it will be forwarded to same server. For same server it creates unique hash value. If the request is coming from same host again and again, this algorithm cannot scatter the load across different links. The limitation of this algorithm is inability to generate different hash value for source and destination host.

Koerner et al. [6] discusses multiple service load balancing strategy having multiple controllers to service load of different type of servers. For example, to handle web server and email server, there will be different load balancer. FlowVisior [7] is used to slice the network resource and assign it to different controllers. The major limitation with this paper was large overhead due to multiple controllers. The link delay parameter was not considered which can greatly affect the performance.

Other load balancing model was based on PyResonance(Resonance implemented with Pyretic) controller[8]. Resonance is SDN control platform that promote event-driven network control. To define network policy it preserves Finite State Machine (FSM) model. Different states shows action depending on network condition. Transition between states shows reaction to dynamic network event. The controller control and balance the network flow by mapping the host (containing FSM) to state it is in. This module however, was just a prototype. In actual SDN environment it is more complex to balance the traffic.

Sukhveer et al. [9] discusses a load balancing technique for handling client request at line rate (the data transmission speed of a communication line or network) enhancing the network performance by proper utilization of available resources. The main aim of this strategy is to increase throughput and minimize latency and response time by scattering the incoming packets in round robin manner [10].The round robin algorithm uses circular queue to determine the server to whom the request will be forwarded. The load balancer comprises of IP address of service and server. The entire client request will go to service IP (address of load balancer) and load balancer will redirect the request to server in round robin manner. So the method is actually balancing service IP and to check whether the server is live or not, ARP probes are sent to the servers. The experiment shows that round robin strategy outperforms random load balancing strategy. The limitations of this paper are:-

- The code was not tested on real hardware.
- The code was tested using single controller whereas the performance of SDN application depend upon the performance of controller.
- Assumes all servers are homogeneous.
- Round robin load balancing method does not take into consideration the current load on the server which is the major limitation of this paper.

For dynamic load balancing of server, it is very important to make the flow table dynamically to reflect the current load. **Qilin et al.** [11] presents an algorithm to dynamically design the flow table and classify different client processing. It was based on “single flow table (for single client) “and “group flow table (for multiple client request)”. Single flow table is used for traffic inspection of each client while multiple flow tables are used to perform traffic forwarding function. It reduces the overhead of maintaining large number of flow tables for each client.

Sabiya et al. [12] proposed yet another load balancing policy named as weighted round robin. The architecture and implementation was same as in case of round robin but the only difference here was that a static weight was attached to each server which was directly proportional to server’s actual capacity i.e. if the capacity of server A is 5 times the capacity of server B the weight assigned to server A will be 5 and that to server B will be 1. This load balancing method is very beneficial in case when there are 2 servers with equal capacity. We want one to get less connection because it is accomplishing some critical task and should not be easily overloaded. Other perk of using weighted round robin algorithm is in case of heterogeneous server i.e. one server running i5 processor and other running i3 processor then we assign static weight to the servers so that server with lower capacity will get less number of request as compared to the one with higher capacity. Although weighted round robin give more prominent result as compared to round robin but it only solves the problem of server being homogeneous. Other limitations of round robin still exits.

The load balancer responsibility is to disseminate the load among servers in transparent way without involving client to directly interact with web server. Instead client send request to load balancer which is redirected to particular web server depending upon the various load balancing strategy. Web server in turn will send the response to load balancer and load balancer will then forward the reply to client whosoever has first requested. In some cases there may be delay in processing request as both request and reply has to pass through the load balancer.

Karamjeet et al. [13] proposed a load balancing strategy in which unnecessary delay is eliminated by not involving load balancer in return traffic i.e. the response is directly send to the client. This scheduling algorithm works by selecting the server with lowest flow connection. A flow statistics message is send to switch by load balancer after every 5 second. Flow Statistics Received Event occurs when the switch send its flow statistics information to the controller. Number of flows sent to the server by load balancer is counted by load balancing application and packet is send to server with least active connections.

However using single controller may cause scalability and availability problem. Using multiple controllers can alleviate this problem in wide area networks. There is still absence of any flexible load balancing mechanism for distributed controller. COLBAS [14], a load balancing scheme for hierarchical controller configuration was proposed by **Selvi et al**. This method relies on controller collaboration via cross-controller communication. It assigns one of the controllers as SuperController. The hosts connected to switches generate request according to poison process with interarrival time of packet exponentially distributed. The super controller collects the load metrics from the controller and check if there is any imbalance. If the request handled by any controller exceeds the upper threshold, SuperController continues to redistribute the request until the load of a controller reaches lower threshold value. In case if multiple overloaded controller and link/node failure, how the mechanism will work was not explained by the authors of this paper. The major limitation of this paper was that it considers packet size to be identical, which is not always possible.

Sufiev et al. [15] proposed Dynamic Cluster, a multicontroller load balancing method for SDN. The architecture consists of a SuperController (SC) and various clusters of Regular Controller (RC). Each cluster must contain equal number of RCs. Load balancing is performed at two levels:-low level load balancing occurs when any RC reaches its threshold value and high level load balancing occurs when on periodical checking, initiated by SC ,it is realized that some cluster have reached its maximum load threshold. To break the interdependency between SC and RC, Cluster Vector (CV) is defined that contain the address of all the RCs in a cluster. Whenever there is cluster imbalance, SC will run partition algorithm to rearrange the RCs in a cluster and return the new CV to the controllers. So two clusters cannot directly communicate with each other without the involvement of Super controller. There can be availability problem in case the SuperController fails. Also due to periodic collection of load information by SuperController can cause communication overhead.

Yuanhao et al. [16] put forward DALB algorithm for load balancing. This method was completely based on distributed decision i.e. there will be no centralized controller. Every controller will first measure its load to check if it surpasses the threshold value. If yes, then it will collect the load information of other controllers and calculate the value of load balancing rate and maximum load. If load is large then it will identify the switch with maximum load and migrate it to low load controller. To avoid the frequent collection of load information from controllers, dynamic and adaptive threshold is adopted. But the limitation with this mechanism is that whenever a controller becomes overloaded then only before making load decision it is collecting the load information of other controller, which reduces the time efficiency of load balancing.

Another approach for load balancing in SDN architecture with multiple controllers [17] was given by **Jinke et al.** [18] which was based on load information. The controller makes the load balancing decision at its own level. Every controller will repeatedly broadcast the information about its load to all the other controllers and save the load information of other controllers to make load migration attainable. So overloaded controller need not collect the load information of other controllers before making decision. Periodically informing the controller about load reduces the time to make decision but on other hand it can cause communication overhead. If present load on controller has not changed much as compared to the last value then notifying it to other controller will only lead to superfluous operation. So the author even proposed an algorithm to lessen the frequency of informing. Each load balancing module in controller consists of: (1) measurement of load to check if controller has reached its threshold value (2) informing about load to other controller (3) make load balancing decision and (4) shifting the switch to other controller to balance the load on local controller.

Sroya et al. [19] implemented an algorithm that assign load to server according to delay. It assigns delay to each link between server and switch according to the speed and dynamic weight is assigned according to delay. The server with minimum delay handles more traffic in contrast to server with more delay. Although this strategy consider parameters like link delay, link speed which is very important from performance point of view but due to single centralized controller there can be single point of failure.

Chen-Xiao et al. [20] proposed a load balancing algorithm that is based on artificial neural network. This algorithm works by measuring the four parameters of every transmission path i.e. packet loss rate, transmission hops, bandwidth utilization ratio and transmission latency.

Artificial neural network with back propagation is applied to anticipate the load on each transmission path and the path with minimum load is used for transmission of data. It reduces the network latency by upto 19.3%.

III. Comparative Analysis

Below mentioned table summarizes the algorithms surveyed along with the parameters considered for performance analysis.

Table 1: Algorithms along with parameters

	Algorithms		**Parameters**					
Algorithm	**Centralized**	**Distributed**	**Overhead**	**Delay**	**Throughput**	**Response Time**	**Flow Table Size**	**Packet Arrival Rate**
[6]		✓	✓					
[8]	✓			✓				
[9]	✓			✓	✓	✓		
[11]	✓						✓	
[12]	✓				✓	✓		
13]	✓				✓	✓		
[14]		✓		✓				✓
[15]		✓						
[16]		✓	✓		✓			✓
[18]		✓	✓		✓			✓
[19]	✓			✓	✓	✓		✓
[20]	✓			✓				

IV. CONCLUSION

In this paper a detailed survey of load balancing has been done. As Software Defined Network has been developed to manage large networks like cloud computing technology, wide area networks, data centre big data. Due to ossification of internet, enormous number of request is arriving at server per second. To increase the performance and efficiency of network, there is requirement of efficient algorithm to balance the load of server to avoid network degradation. The centralized controller of SDN has the global view of network which makes load balancing in SDN easy. The load balancing algorithm must consider the current load to reflect the real time change. Using single centralized controller can lead to single point of failure. So load balancing algorithm should be mainly based on distributed decision. Researchers should do more detailed study of distributed architecture to develop better load balancing algorithms taking advantage of SDN architecture. The algorithm should be designed in such a way that it minimizes the latency and response time and maximize the throughput.

V. REFERENCES

[1] Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2015): 14-76.

[2] Vaughan-Nichols, Steven J. "OpenFlow: The next generation of the network?" Computer 44.8 (2011): 13-15.

[3] A. Voellmy and J. Wang, "Scalable software defined network controllers," SIGCOMM Comput. Commun. Rev., vol. 42, no. 4, pp. 289–290, Aug. 2012.

[4] Serverwatch.com, "5 Load Balancers You Need to Know,"2015.

[5] "Load-Balancing: Hash Methods," Calix, 2010.

[6] Koerner, Marc, and Odej Kao. "Multiple service load-balancing with OpenFlow." High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on.IEEE, 2012.

[7] Sherwood, Rob, et al. "Flowvisor: A network virtualization layer." OpenFlow Switch Consortium, Tech. Rep 1 (2009): 132.

[8] Zhou, Yuanhao, et al. "A method for load balancing based on software defined network." Advanced Science and Technology Letters 45 (2014): 43-48.

[9] Kaur, Sukhveer, et al. "Round-robin based load balancing in Software Defined Networking." Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on. IEEE, 2015.

[10] Ghaffarinejad, Ashkan, and Violet R. Syrotiuk. "Load balancing in a campus network using software defined networking." Research and Educational Experiment Workshop (GREE), 2014 Third GENI. IEEE, 2014.

[11] Qilin, Mao, and Shen Weikang. "A load balancing method based on SDN." Measuring Technology and Mechatronics Automation (ICMTMA), 2015 Seventh International Conference on. IEEE, 2015.

[12] Sabiya, and Jaspinder Singh. "Weighted Round-Robin Load Balancing Using Software Defined Networking." International Journal Of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 6, June 2016, pp. 621–625.

[13] Kaur, Karamjeet, Sukhveer Kaur, and Vipin Gupta. "Flow statistics based load balancing in OpenFlow." Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on. IEEE, 2016.

[14] Selvi, Hakan, Gürkan Gür, and Fatih Alagöz. "Cooperative load balancing for hierarchical SDN controllers." High Performance Switching and Routing (HPSR), 2016 IEEE 17th International Conference on. IEEE, 2016.

[15] Sufiev, Hadar, and Yoram Haddad. "A dynamic load balancing architecture for SDN." Science of Electrical Engineering (ICSEE), IEEE International Conference on the. IEEE, 2016.

[16] Zhou, Yuanhao, et al. "A load balancing strategy of sdn controller based on distributed decision." Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, 2014.

[17] Blial, Othmane, Mouad Ben Mamoun, and Redouane Benaini. "An overview on SDN architectures with multiple controllers." Journal of Computer Networks and Communications 2016 (2016).

[18] Yu, Jinke, et al. "A load balancing mechanism for multiple SDN controllers based on load informing strategy." Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific.IEEE, 2016.

[19] Sroya, Manamrit Singh, and Vikramjit Singh. "LDDWRR: Least Delay Dynamic Weighted Round-Robin Load Balancing in Software Defined Networking." International Journal 8.5 (2017).

[20] Chen-Xiao, Cui, and Xu Ya-Bin. "Research on load balance method in SDN." International Journal of Grid and Distributed Computing 9.1 (2016): 25-36.

Authors Profile

Priyanka Kumari completed B.tech in Computer Science from Banasthali Vidyapith in 2015 and pursuing M.Tech in Computer Science from Banasthali Vidyapith, Rajasthan.

Dipanwita Thakur received her M.Tech degree in computer science in 2007. She has worked for Banasthali Vidyapith(Deemed to be University), Rajasthan since 2008, and now she is an Assistant Professor. Her research interests include Software Defined Networking, Cellular learning Automata and Network Virtualization.