I J C T A, 9(14) 2016, pp. 6647-6652
© International Science Press

A Mediator based Dynamic Server Load Balancing Approach using SDN

Ashwati Nair[1], Binya mol M. G.[2] and Nima S. Nair[3]

ABSTRACT

In the modern era, Network Management is becoming so complex with several networking devices like switches, routers and middle boxes for manipulating network traffic for different purposes. Software Defined Networks is a new archetype that meets the ossifying needs of the network. It separates the control plane from the data plane of forwarding devices by introducing a controller that brings the idea of programmable network. This paper suggests a dynamic load balancing scheme for a server farm using the concepts from SDN. The proposed work introduces two mediators' named Load mediator and Channel mediator, which will control the server load by updating the Load status table in the controller systematically. The proposed method has been implemented and compared with popular load balancing schemes like Round Robin and Random algorithms.

Keywords: SDN, Load Mediator, Channel Mediator, Load Status Table, Flow Table.

1. INTRODUCTION

Software defined networking is an expedient network technology which serve network operators more control of the network infrastructure. SDN technology splits the control plane logic from the data plane, by moving networking control functions from forwarding devices to a logically centralized controller, so that the networking function can be performed by software. The SDN centralized controller presents enormous opportunity for network operators to re-factor the control plane and to enhance the performance of applications [8]. In SDN architecture Openflow enables the controller to communicate with network devices [13]. It is a switching protocol based on SDN conception [9].

The OpenFlow switch divides the function into two; the switch consists of data portion, although routing decisions are shifted to a separate controller. The OpenFlow protocol helps to communicate via the switch and the controller. It contains a flow table contents that incorporates a set of packet field to match, and an action. If the packet received by the switch does not match with the flow entry are send to the controller. The controller then makes a decision to handle this packet. Either it can drop the packet or it can add a flow entry which helps the switch to forward similar packets in the future [11].

Beyond with the advance of network, servers have to serve more and more services for the increasing number of visits, the load has an intense development. It requires a lot of money to prove the ability of servers, and it does not achieve well [3]. Load balancing can be considered as a preferred way to solve this problem. This method distributes the work load equally among all the present nodes in the given environment, such that it assures no node in the system is over loaded or sits idle for any instant of time [5]. The server cannot manage so many visits. One way to elucidate this problem is by improving server performance. Another way to solve this problem is Load balancing. Load balancing uses multiple servers to proffer different services to the users. The server shares the work load and works better than the high performance

[1] PG Student, [2] PG Student, [3] Assistant Professor
Department of Computer Science & IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham (Amrita University), India
E-mails: [1]aashwatinair3@gmail.com, [2]mgbinnyamol@gmail.com, [3]nimasnair@gmail.com

one with heavy load. So load balancing technology has become a major technology to built high load websites [3]. Different load balancing schemes are used such as static load balancing and dynamic load balancing. In static load balancing the node performance is determined during the execution time [7]. Where as in dynamic load balancing it makes changes to the distribution of work among work stations at run-time [2].

2. RELATED WORK

Load balancing is one of the most challenging problems in computer networks. So far many research works have been introduced and solved server latency problem, scalability, throughput, response time with efficient server CPU and memory utilization. A dynamic load balancing approach for cloud infrastructure proposed an intense load calculation of virtual machine in a data center. Whenever virtual machine load reaches its threshold value, an agent named Load agent initiate search for a candidate virtual machine from other data centers. This method reduces service time by keeping information of candidate beforehand [10].

Another approach suggests an improvement of load balancing in software defined networks through load variance-based synchronization. The experiment explored the controller state synchronization problem in SDN based load balancing. They suggested two LVS based schemes to overcome the forwarding-loop problem and decreases controller synchronization overhead as compared with the traditional PS-based load balancing [8]. A modified round robin algorithm for DNS load balancing provides best services with large number of distributed servers connected uniformly. Drawback of this approach is that it works better only in distributed environment [4]. A server cluster dynamic load balancing method based on OpenFlow technology has been developed to solve load balancing problem based on network virtualization in data center [1]. Another method suggests a Dynamic server load balancing algorithm using OpenFlow and sflow effectively distribute traffic among server clusters. Wildcard rules are installed in the switch to assemble traffic of server replica, and decisions are based on real time traffic statics via the sflow protocol [6].

3. PROPOSED SYSTEM

In this paper we proposes a method named “A Mediator Based Dynamic Server Load balancing approach Using SDN” by introducing two mediators called Load Mediator and Channel Mediator. Since load balancing in SDN requires searching for under loaded servers. Here the two mediators suit the purpose and fulfill it appropriately, without putting additional burden on the network. These two mediators are, programmed as two modules. One part is installed inside the controller and the other part is coded in the servers of the server farm. The role of Load mediator is to update the Load Status Table periodically according to the current server status. By using the help of these two mediators, server load balancing is achieved with minimum response time and maximum efficiency. Description of various components used in this algorithm is as follows.

3.1. Load Mediator (LM)

It maintains server status in Load Status Table inside the controller. The main role of this mediator is to calculate each server status using the information provided by the channel mediator.

3.2. Channel mediator (CM)

It is a part of a code installed in the controller and each servers of the server farm. This will determine each server status. Whenever the controller invokes a request to the server farm, the Channel mediator code inside the servers will remit the load status to the controller in every 30 seconds. The Channel mediator code has been set up in two parts. One part resides in the controller and the other part in the server as a memory resident program.

The proposed method introduces a component named Load mediator (LM) which resides in the controller that specifies the server status in the load status table. Load status table contains server id (Sid), Memory utilization (α), CPU utilization (β), Health value (γ), and load status as normal or overloaded. This table will be periodically updated by the channel mediator in every 30 seconds to congregate information from the server. The Channel mediator performs two functions. One function performs inside the controller and the other one resides in the server farm.

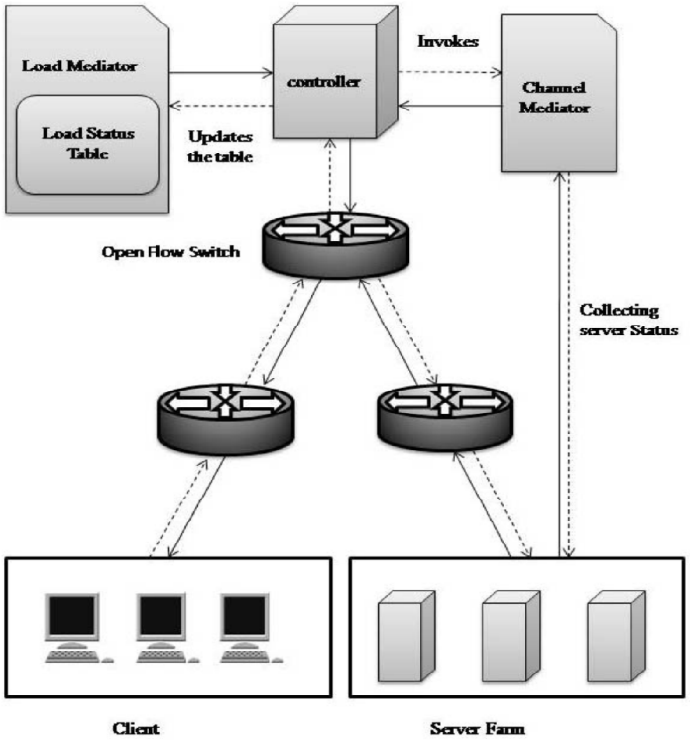

Figure 1: Detailed view of the Architecture

When the channel mediator code in the controller invokes an internal channel mediator function in the server, it will gather server status information from each server and send them back to the controller. There after the load mediator will update the load status table according to the server status collected by the channel mediator code. Load mediator sorts the load status table and identifies the least loaded server. Then the controller updates the flow table entry in the OpenFlow switch, so that any request arrives at the switch will be forwarded to the least loaded server. Thus the server load balancing is performed. The detailed view of proposed mechanism is shown in figure 1. Load mediator maintains the load status of various servers available in the server farm. The major task of channel mediator is to gather current load status of each server in terms of available memory (α), CPU utilization (β), and finally it estimates the health value (γ) of each server, which is directly proportional to the RAM size of the server. The calculated values are stored in the load status table using the following equation.

$$\alpha_{available} = \alpha_{total} - \alpha_{used} \tag{1}$$

$$\gamma = \frac{\alpha_{available}}{\alpha_{total}} \times 100 \tag{2}$$

The health value (γ) % will give each server status in the server farm. In this context 30 % is taken as a threshold value, and if this health value is greater than 30%, the server status is said to be normal. And if the health value (γ) is less than or equal to 30 % then the server is overloaded, and load balancing has to be performed.

Table 1
Load Status Table

Server ID	*Memory utilization(α)*	*CPU utilization(β)*	*Health value(γ)*	*Load status*
s_1	α_1	β_1	γ_1	Normal
s_2	α_2	β_2	γ_2	Over loaded
..	..	..	..	..
s_n	α_n	β_n	γ_n	Normal

When the client sends a request for the server, the switch will check the matching content in the flow table. This flow table is already updated by the controller. If the flow entry matches with the incoming packet header, the request is directed to the least loaded server. Else the switch communicates with the controller, and the Channel mediator code in the controller invokes the channel mediator code in the server.

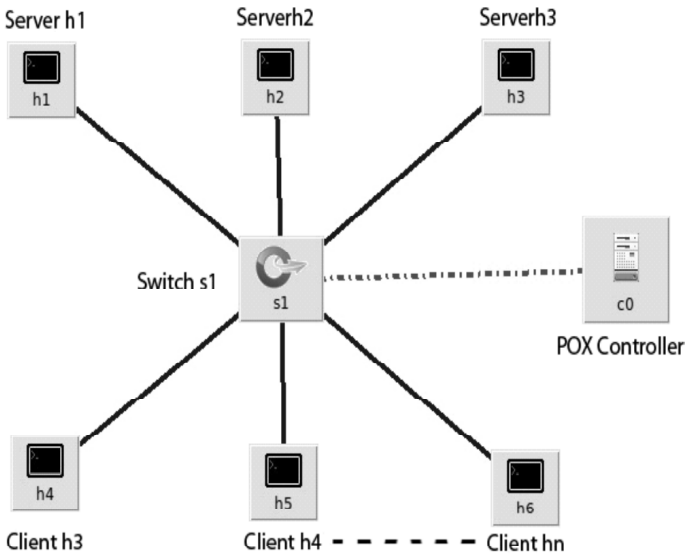

Figure 2: Network topology

4. IMPLEMENTATION

To experiment with the proposed system we used Mininet network emulator [11], and POX controller. Then we created a network topology using miniedit.py which introduces a controller, OpenFlow switch, servers and host as shown in the fig.2. POX controllers are programmed using Python language [12]. For the server setup, the given three hosts are converted to web servers by running HTTPd service in all these servers. We created a program module named Channel_Serv.py and installed servers. This module will be invoked by the Channel_Pox.py which is coded in POX controller. The Channel_Serv program will send the current status of the servers to the controller and update the Load Status Table. The other module called LoadMediator.py will sort the table and find out which is the least loaded server. It will also update the Flow Table so that the clients are automatically routed to the least loaded server. Using this scenario, the performance of the proposed algorithm has been observed in three cases. For testing and comparing the performance of our algorithm, we used the traditional Random and Round Robin algorithms. In Random algorithm, the incoming requests from the client to the switch are forwarded to the controller. The controller randomly selects a server from the server farm and responds to the client. The controller updates the flow entry and sends to the switch. Whereas in Round Robin algorithm, the controller sequentially select the server, and respond to the client till the end of one cycle and repeats the process in certain order [6]. The proposed algorithm reduces the response time and throughput as compared with the traditional load balancing

algorithms. Figure.3 depicts the graphical representation of the minimum response time and request arrival rate/sec of three load balancing schemes. The X-axis shows request arrival rate/sec and Y-axis represents minimum response time/sec. Here Random and Round Robin algorithm shows 25 seconds and 23 seconds of response time. At low rate, request can be served fast in Random and Round Robin algorithm. Whereas, when the request increases the proposed method gives 17 seconds of response time. Since the OpenFlow switch is updated periodically within every 30 seconds by the controller in advance, the response time lagging is significantly reduced in our method. By this way, the other packets followed by the process can be served quickly.

Figure 4 shows the graphical representation of Number of request and throughput of three load balancing algorithm. In Random it shows 230 KB/seconds throughput and in Round Robin the throughput is 250KB/ seconds. Mediator based algorithm shows 350 KB/seconds of throughput. This proves that the mediator based load balancing method results better throughput than Random and Round Robin method.

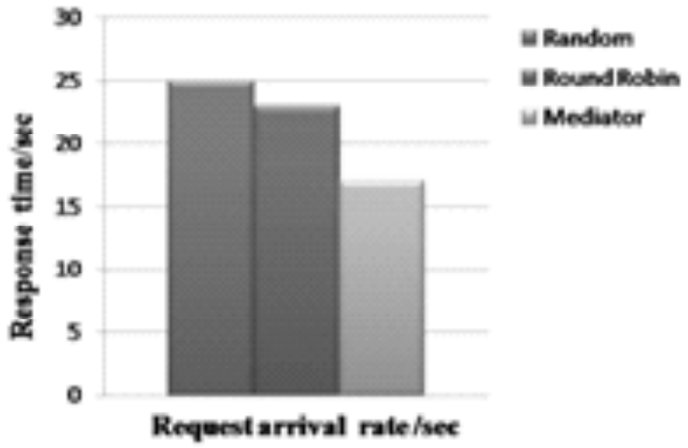

Figure 3: Comparison of response time

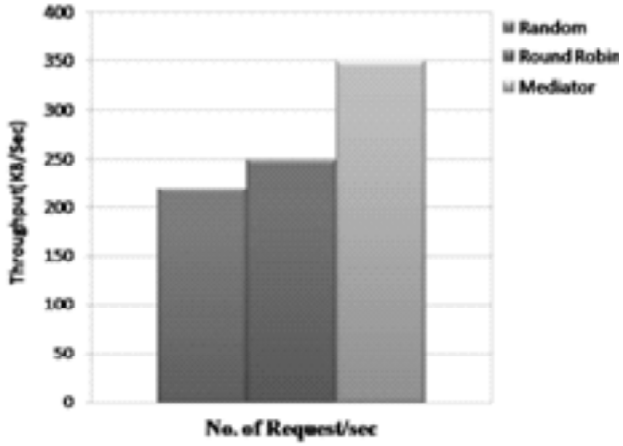

Figure 4: Comparison of Throughput

5. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed a dynamic load balancing method two mediators named Load mediator and Channel mediator. The two mediators will control the server load, by updating the Load status table in the controller systematically. Beyond that, the algorithm shows better response time and throughput as compared with the traditional Random and Round Robin algorithm. The proposed method has been implemented and has given satisfactory result. In the future work, we are planning to implement this method in a real time campus network. Also we are planning to integrate this module to a cloud computing environment.

REFERENCES

[1] WenboChen, ZhihaoShang, XinningTian , Hui Li, “Dynamic Server Cluster Load Balancing in Virtualization Environment with OpenFlow”, International Journal of Distributed Sensor Networks, Volume 2015, Issue 8, Jan 2015.

[2] Sameena Naaz,” To Develop and Analyze a Fuzzy Based Algorithm for Load balancing in a Distributed Environment”, Shodhganga : A Reservoir of Indian Theses @INFLIBNET” Oct 2014

[3] FanZhang, JinyaoYan, BoLiu, HaiyanMa, “SDN based load balancing and its performance”, Proceedings of the Fourth International Conference on Multimedia Technology (ICMT),March 2015.

[4] Senthil Ganesh N ,Ranjani S ,” Dynamic load balancing using software de fined networks”- International journal of Computer Applications (IJCA), International Conference on Current Trends in Advanced Computing (ICCTAC), Issue 2, 2015.

[5] Geethu Gopinath P P, Shriram K Vasudevan, “An in-depth analysis and study of Load balancing techniques in the cloud computing environment”, International Journal of Computer Science and Engineering, Volume 50, Issue 1, May 2015.

[6] Qingwei Du, Huaidong Zhuang, "OpenFlow-Based Dynamic Server Cluster Load Balancing with Measurement Support", Journal of Communications, Volume 10, Issue 8, 2015.

[7] Avneet 'Gutz' Bagga, Network Geek (*www.quora.com*) 2014.

[8] ZehuaGuo, MuSu, YangXu, ZheminDuan, LuoWang, ShufengHui, H. JonathanChao, "Improving the Performance of Load balancing in Software defined networks through Load Variance-based Synchronization", Elsevier, volume 68, August 2014.

[9] Srinivas Govindraj, Arun Kumar, Jayaraman, Nitin Khanna, Kaushik Ravi Prakash, "OpenFlow: Load balancing in Enterprise networks using Floodlight controller" , May2014.

[10] Aarti Singh, Dimple Juneja, Manisha Malhotra, "Autonomous Agent Based Load balancing Algorithm in Cloud Computing", Elsevier,Volume 45, March 2015.

[11] *Mininet.http://www.mininet.org/*

[12] *www.searchsdn.techtarget.com*

[13] A Lara, A. Kolasani, B. Ramamurthy, "Network Innovation using Open Flow: A Survey"-IEEE Communications surveys and Tutorials, Volume 16, Issue1, 2014.