A Comparative Study on Load Balancing Algorithms in Software Defined Networking

Neha Joshi and Deepak Gupta(✉)

Computer Science and Engineering, NIT Arunachal Pradesh, Yupia, India
nehajoshi4321@gmail.com, deepakjnu85@gmail.com

Abstract. Advent of big data, cloud computing and IOTs resulted into significant increase in traffic on servers used in traditional networks as these networks are normally non-programmable, complex in management, highly expensive in nature, and have tightly coupled control plane with data plane. To overcome these traditional network-based issues a newly emerging technology software defined networking (SDN) has been introduced which decouples the data plane and control plane and makes the network fully programmable. SDN controllers are programmable so an efficient load balancing algorithms must ensure the effective management of resources as per client's request. Based on these parameters i.e. throughput, transaction rate, & response time the qualitative comparison between the load balancing algorithms of SDN is done to generate the best results.

Keywords: Software defined networking (SDN) · SDN controller · Mininet emulator tool · Sniper tool · Siege tool · Open flow

1 Introduction

SDN is rapidly emerging technology in the networking field, by using SDN architecture we can easily manage different network applications and services. SDN separates the network logic control plane and the forwarding element data plane (e.g. Router, switch). As a result, the network management information and the network logic are centralized together over the SDN controller (also known as control plane). The lead role in SDN architecture is played by the SDN controller, which controls all the functions of the network by the help of openflow protocol [1, 2].

With SDN, the network is fully programmable, more agile and scalable. It provides the flexibility to switch into the cloud environment, virtualization, private network and public network. Therefore, we can easily add or remove different routers, switches as per the requirement and implements different network application by the help of software based SDN controller in the system. The major applications of SDN are cloud integration, network monitoring, distributed system control, security services, automation, etc. However, in traditional network systems both planes i.e. data, and control plane are securely coupled with each other. Therefore, every network application needs an individual hardware and that hardware's are very expensive, inflexible and vendor-specific. The traditional load balancers are manufacturer based and they set the specific algorithms on it, which we cannot change according to our feasibility.

N. Kumar and R. Venkatesha Prasad (Eds.): UBICNET 2019, LNICST 276, pp. 142–150, 2019.
https://doi.org/10.1007/978-3-030-20615-4_11

To solve these issues in conventional load balancing method, we develop the load balancing algorithm and implement it over the control plane which convert the simple Openflow device into an effective load balancer [3, 4].

The enhancement in network technology introduces so many challenges. One of the major issues in network architecture is delay in providing the response to end users. Usually delay occurs when the system device is overloaded and creates a bottleneck situation in the system. To distribute the load among the different servers and to prevent the bottleneck situation we require the load balancers. In existing network system load balancer device uses various types of load balancing algorithms, so to distribute the large amount of client traffic into several servers it takes too much time to process and make the system inconsistent. One of the major applications of network is load balancer so that we implemented the load balancing application over the SDN controller and then controller is act as a load balancer. Thus, an SDN load balancer distributes traffic more easily in less amount of time and in efficient manner [5, 6].

To investigate the network load balancer performance of SDN, we compare different load balancing algorithms to evaluate the best performance among different scheduled algorithms. We implemented load balancing application over the SDN controller (i.e. POX Controller). To perform the task, POX controller is used which supports the Python language and mininet emulator tool is used to create the network topology, which provides the same virtual hardware setup as in realistic environment.

In this paper, we compare four different load balancing algorithms of SDN on the basis of following parameters as: response time, transaction rate, and throughput. The following four algorithms are as:

- Round-Robin Strategy [7, 8]
- Implementation of server load balancing algorithm [9]
- Flow statistics load balancing algorithm [10]
- Least time based weighted load balancing algorithm [11]

The main tasks performed in this experiment are:

- Tested the comparison on Mininet emulator tool [12, 13].
- Compared above mentioned algorithms by the help of various parameters namely throughput, response time, and transaction rate.
- Tested the result by the help of Load Balancer Sniper and Siege Tool.

This paper consists of five sections. Section 2 represents the background and related work of the load balancing and related algorithm using SDN and also displays the SDN architecture consisting of all the layer i.e. application layer, control layer, and infrastructure layer. Section 3 describes the load balancer architecture related to our topology used in the simulation process. Section 4 consists all experimental result, network setup, load testing tool, and emulator tool description. Overall this section shows the graphical representation of all the result. Section 5 represents the conclusion and future work of the paper.

2 Background and Related Work

The enhancement in network technology leads to increase in network traffic. Therefore, it is hard to handle the large amount of requests by the single server. The main aim of load balancer is to disburse the load among various servers and help us to increase the network performance by efficient use of all available resources in the network systems.

Silva et al. [6] explained that SDN load balancers are real, flexible, agile, and cost-effective over the conventional load balancers. They evaluated the performance of SDN load balancers with different scheduled algorithms. Kaur et al. [8] executed the Round Robin strategy. The demerit of this paper is that it does not include the load of the server & time delay. This method supposed that all the servers present in that particular network system have equal number of request and every link possess same speed. However, in real world the scenario is quite different. Practically all the link has different bandwidth and speed. Kaur et al. [11] implemented the least time based load Balancing strategy. In that case, the load balancer sends the client request to that server which has least time delay instead of any other servers having more delay. Koerner et al. [14] discussed one or more load balancing concepts in which one of the load balancer is taken care of balancing web servers whereas another load balancer is needed for balancing e-mail servers.

2.1 SDN Architecture

The SDN Architecture as shown in Fig. 1 consists mainly three components:

- **Application Layer**: It is the topmost layer of the SDN architecture. The SDN application layer consists of many network applications which create an abstract view from the internal network and to build the communication with SDN controller the API (Application Program Interface) used by the programmer is called Northbound API.
- **Control Layer:** It is also known as the control plane of the SDN architecture. All the routing decisions, management of the network is done by control layer. It is also called as the network operating system (NOS) that control all the operations of the SDN. To communicate with various network devices like routers, switches, etc. the SDN controller uses southbound API. The load balancer application is run on top of the SDN controller and the load balancing algorithms is installed on load balancer application.
- **Infrastructure Layer:** The bottom layer of SDN architecture also known as data layer. This layer helps to forward the packets by some set of rules given by the SDN controller. The infrastructure layer is the connection of various physical devices or virtual devices such as routers, switches, etc. The SDN controller defines and installs rules on the flow tables of Openflow switches.

The decoupled data plane and control plane are communicated by the help of Openflow protocol. This protocol helps us to exchange the information between these two planes. A secure channel is used to carry the information from the Openflow switch and the control plane by the help of Openflow protocol [2].

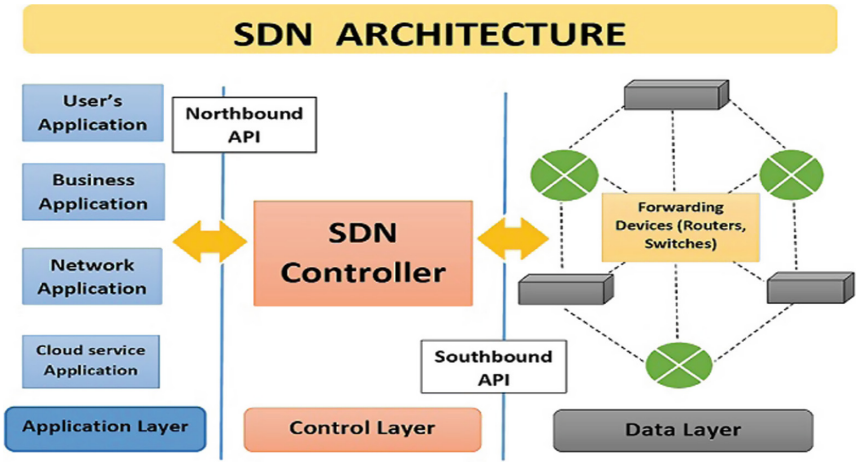

Fig. 1. SDN architecture

This architecture of SDN supports all the legacy network applications and provides the more enhanced features of the network system such as scalability, feasibility, adaptability, flexibility etc. So, this is only being happened due to its programmable nature. As we all know SDN controller is programmable so now it is easy to implement any kind of application over it and make the network more agile and programmer dependent [15].

3 Load Balancing Architecture

It consists the SDN controller, which is fully programmable and behaves like a Load Balancer after installing the load balancing algorithms over it and represents number of servers where load is distributed according to the load balancing algorithms as shown in Fig. 2.

The load balancer application consists an algorithm by which it takes the decision to select one server from the pool of servers and distributed the load simultaneously as per client's request [4]. In this architecture of load balancing first client sent the request to the server which is controlled by the SDN controller (act as a load balancer) then load balancer sent their request to the one server according to their scheduled algorithm. Server processes the client requests and gives the response back to the client. In this scenario, the controller communicates with openflow switch via openflow protocol using southbound API. In our experiment, we use four types of load balancing algorithms, which are described below:

Round-Robin Strategy: This algorithm is defined as the requests are sent to each server available in the queue one by one in a circular manner. When any packet is arriving, the next chosen server is available on the queue of all present servers in the

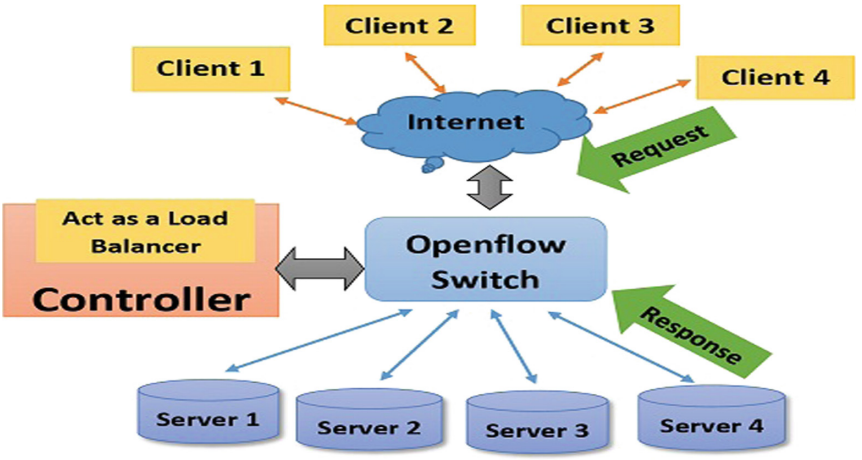

Fig. 2. Load balancing architecture of software defined networking.

network system. So, that all the servers in the list is in the same order and handles the equal number of load, excluding the load present on each server [8].

Server Load Balancing: This algorithm is explained as the load is served to that server where the server has the lowest CPU load value and the server, which has the minimum load value, is chosen. The server determines their current load value by the help of mpstat command [9].

Flow Statistics Load Balancing: It is defined as the server, which has minimum number of flow connection, is selected for processing the next request. After every 5 s the Openflow switch receives the flow-statistics request message from the load balancer. The total no. of requests that were sends to each server is counted by the load balancer. Then after the server with least active connection handles the next upcoming packet for processing which is send by the load balancer [10].

Least Time Based Weighted Load Balancing: It depends on the time delay of the server. Server with less delay can deal with the more no. of requests. At first, we assigned various delay on every link between the Openflow switch and the servers. Secondly, we assigned unique weight to the server based on delay on each server. Then we set more weight to that server which has least delay and that server is connected to Openflow switch [11].

4 Result and Discussion

There are various tools available by which we can test and compare our SDN load balancer application. We used the mininet emulator tool which helps us to create the network topology containing the number of hosts, forwarding element switches and the controllers [12, 13]. In our framework we implement the python based POX controller as load balancer and one device act as an Openflow switch (forwarding element) and

other systems act as a host and remaining systems works as a server where load is distributed by the help of load balancer [16]. We also use load balancer Sniper testing tool on host computer to generate the readings of different parameters to find the least active connection in the server, mpstat and netstat command is used.

For experimental estimation, we compare the above mentioned load balancing algorithm with each other by the help of the attributes like throughput, response time, and transaction rate.

Mathematically the throughput can be calculated as the number of bits processed in per unit time. It is denoted as:

$$throughput = \frac{\#bits}{second}$$

Response time is defined as the total processing time for all the users and is divided by the number of users. Response time usually gives the total time taken by the request response process. It is the amount of time to process the request by the servers when it is received the request by the client. We calculate the response time by the given formulae:

$$response\ time = \frac{total\ processing\ time}{total\ number\ of\ users}$$

Transaction rate can be calculated as the number of http request-response pair is processed in per unit time. It is usually an amount of information or request-response pair is exchanged from the server in a given amount of time. So, the maximum transaction rate shows the faster and better response. It can be denoted as:

$$transaction\ rate = \frac{\#\ http\ request\ response\ pair}{second}$$

We simply send the different number of requests as per clients to the load balancer tool (i.e. siege tool and sniper tool) according to the scheduled algorithms and it displayed the output readings of different attributes. Based on these above mathematical equations the parameters like throughput, response time and transaction rate gave their value. By running this whole setup by the help of mininet emulator tool we get the results and on the basis of those output readings graph is plotted.

The graphical representation of all the parameter results is shown as below:

Figure 3 represents the throughput result. Horizontal axis represents the concurrent users and vertical axis represents the throughput (mb/sec). On behalf of this parameter, server load balancing shows the better result than any other given algorithms. Throughput means the number of requests in bits is processed in a given amount of time.

Figure 4 presents the response time result. Horizontal axis denotes the no. of users and vertical axis shows the response time in sec. Based on this parameter, flow-statistics algorithm has the least response time among all other given algorithms. By the way, both server load balancing and flow statistics load balancing algorithms shows the similar kinds of results. However, flow statistics based application gives the better response time.

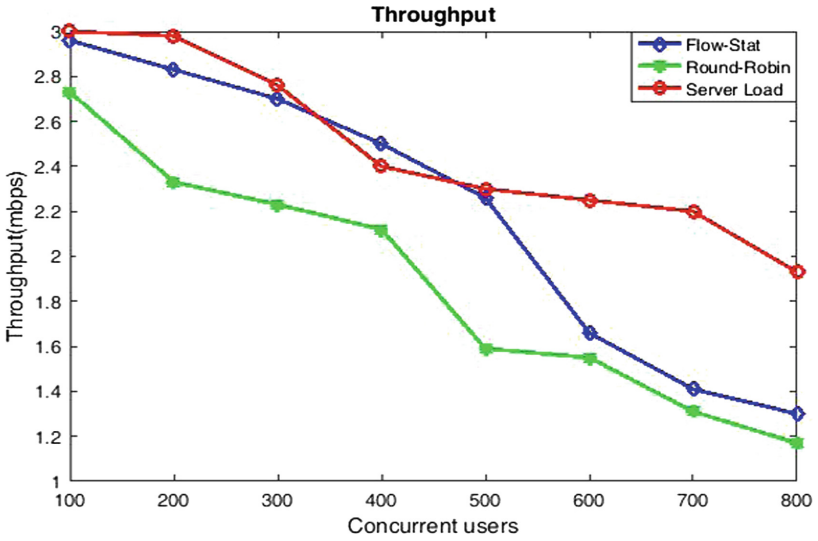

Fig. 3. The throughput comparison of three algorithms i.e. round-robin, flow statistics, server load algorithm.

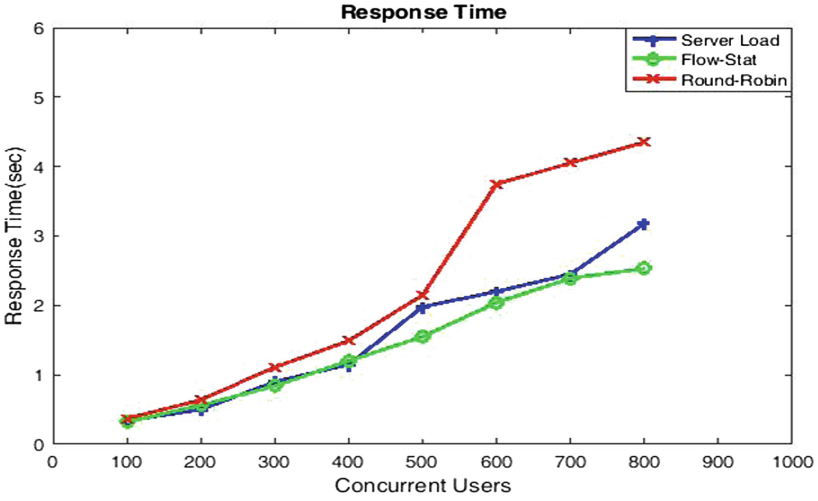

Fig. 4. The response time result of three algorithms as round-robin, flow statistics, server load

Figure 5 shows the transaction rate of the server. In that case server load balancing is compared to the round-robin strategy and it gives better results than round-robin method. Server load balancing algorithm has the higher transaction rate that means it processes the request faster than the round-robin algorithm. The X-axis of the graph shows the total number of users sends the requests and the Y-axis represents the transaction rate of the server.

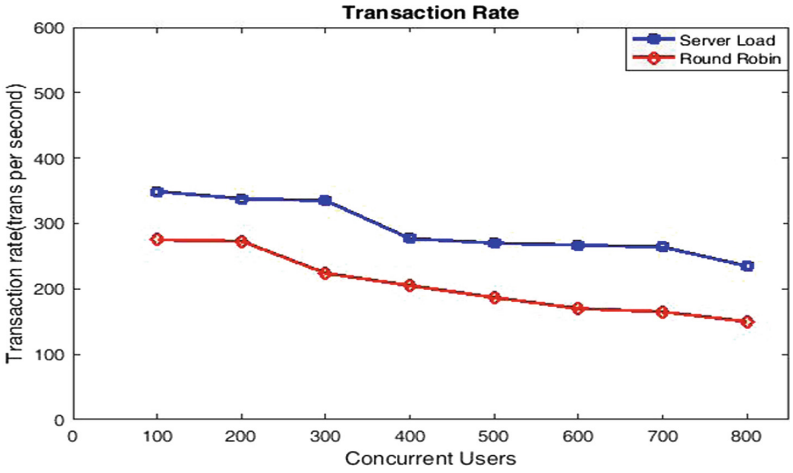

Fig. 5. The transaction rate of server load and round-robin algorithm.

5 Conclusion and Future Work

SDN load balancer deals with lots of issues of conventional load balancer in an efficient and cost-effective way. By the help of above experiment and comparison, the result is that among these four types of algorithms "Implementation of server load balancing" is the best load balancing algorithm in software defined networking with respect to these parameters as throughput, transaction rate, and response time.

However, the flow statistics based algorithms shows the better response time in comparison to the server load balancing algorithms. The main challenge of this experiment is that it is not tested in the real time hardware system, it is based on the mininet emulator tool, which provides the real time simulation of the experiment but not in the actual hardware.

To get the better results we can use RYU controller instead of using POX controller and can use more than one controllers in place of using single controller. So, that if any failure is occurring in the single controller we can easily recover it by using another controller [17].

Acknowledgment. We thank Mr. Sunit Kumar Nandi of NIT Arunachal Pradesh for helping us to understand the RYU controller functionalities and to learn the mininet emulator tool concepts.

References

1. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
2. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software-defined networking. IEEE Commun. Surv. Tutorials **17**(1), 27–51 (2015)

3. Kim, H., Feamster, N.: Improving network management with software defined networking. IEEE Commun. Mag. **51**(2), 114–119 (2013)
4. Neghabi, A.A., Navimipour, N.J., Hosseinzadeh, M., Rezaee, A.: Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature. IEEE Access **6**, 14159–14178 (2018)
5. Qilin, M., Weikang, S.: A load balancing method based on SDN. In: 2015 Seventh International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). IEEE (2015)
6. Silva, W.J.A., Dias, K.L., Sadok, D.F.H.: A performance evaluation of software defined networking load balancers implementations. In: 2017 International Conference on Information Networking (ICOIN). IEEE (2017)
7. Deep, G., Hong, J.: Round robin load balancer using software defined networking (SDN). Capstone Team Res. Proj. **5**, 1–9 (2016)
8. Kaur, S., Kumar, K., Singh, J., Ghumman, N.S.: Round-robin based load balancing in software defined networking. In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2136–2139. IEEE (2015)
9. Kaur, S., Singh, J.: Implementation of server load balancing in software defined networking. In: Satapathy, S.C., Mandal, J.K., Udgata, S.K., Bhateja, V. (eds.) Information Systems Design and Intelligent Applications. AISC, vol. 434, pp. 147–157. Springer, New Delhi (2016). https://doi.org/10.1007/978-81-322-2752-6_14
10. Kaur, K., Kaur, S., Gupta, V.: Flow statistics based load balancing in OpenFlow. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE (2016)
11. Kaur, K., Kaur, S., Gupta, V.: Least time based weighted load balancing using software defined networking. In: Singh, M., Gupta, P.K., Tyagi, V., Sharma, A., Ören, T., Grosky, W. (eds.) ICACDS 2016. CCIS, vol. 721, pp. 309–314. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5427-3_33
12. Mininet: "Mininet - An Instant Virtual Network on your Laptop (orother PC)" (2016). http://mininet.org/
13. De Oliveira, R.L.S., Schweitzer, C.M., Shinoda, A.A., Prete, L.R.: Using mininet for emulation and prototyping software-defined networks. In: 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), pp. 1–6. IEEE (2014)
14. Koerner, M., Kao, O.: Multiple service load-balancing with OpenFlow. In: 2012 IEEE 13th International Conference on High Performance Switching and Routing (HPSR). IEEE (2012)
15. Salman, O., Elhajj, I.H., Kayssi, A., Chehab, A.: SDN controllers: a comparative study. In: 2016 18th Mediterranean Electrotechnical Conference (MELECON), pp. 1–6. IEEE (2016)
16. Kaur, S., Singh, J., Ghumman, N.S.: Network programmability using POX controller. In: ICCCS International Conference on Communication, Computing & Systems, vol. 138. IEEE (2014)
17. De Oliveira, B.T., Gabriel, L.B., Margi, C.B.: TinySDN: enabling multiple controllers for software-defined wireless sensor networks. IEEE Latin Am. Trans. **13**(11), 3690–3696 (2015)