An Analysis of Load Balancing Algorithm Using Software- Defined Network

M. Sakthivel[a], N. Balakrishna[b], K.S. Kannan[c], and P. Devabalan[d]
a, b
Sree Vidyanikethan Engineering College, A.Rangampet, Tirupati-517102, India,
[c]CMR Engineering College, Medchal Road, Hyderabad-501401, India
[d]BVS Engineering College, Odalarevu, East Godhavari-533210, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 20 April 2021

Abstract: A data center is one of the secured space which serves as a house for many disks, switches, servers, routers, and several other hardware for a computer. It is a hardware solution for users who are present near the data center. Cloud Computing (CC) is the newer version of the data center which is capable of giving computer services to the users. Cloud service is the one which provides the users with data center set up when needed or preferred by the user themselves. In general cloud network service will be limited to a particular location or zone. In case if the target area of the user is near then the server will complete the action needed for the user. Only certain servers can provide service for the user in such cases some servers will remain idle. If the provided service by the servers is not used properly this may also lead to the issue of processing and managing a larger set of traffic which is one of the difficult tasks to perform. More network traffic may lead to giving more amount of pressure and complexity to the data center. In such a case Load Balancing (LB) is the preferred process for reducing the network failure and degradation process of the entire network. Software-defined networking is one the upcoming process which is capable of managing the entire network and also gives a general view about the network and its further configuration which needed to be upgraded. In this research paper, a software-defined network is developed for LB algorithms. In this paper, a Dynamic Server LB algorithm (Dserv- LB) is used for open-flow switches in the software-defined network system. This algorithm is one of the packet type LB algorithm. The request in the server is directly forwarded to the web-server with a higher level of server resources. From the result of the proposed algorithm, it was found that Dserv-LB is capable of improving the performance of the entire network and properly uses the server resource.
Keywords: Datacenter, load balance, cloud computing, cloud servers, software- defined network

1. Introduction

SDN is the emerging new networking system that is capable of dealing with a certain level of issues of the conventional network system. In SDN the data and control planes are separated. Now the network routers and switches will have the data plane packets. The control plane is the centralized one which is capable of controlling the forwards to the network(Xie et al., 2018). The control plane can simplify the configuration, packet processing, and innovations. By providing an international viewpoint in the aspects of networking and assist the configuration for the networks. The software-defined network can be classified into 3 categories as

1. Application plane
2. Control plane
3. Data plane

The application plane is placed along with application and protocol, the control plane is designed with the centralized locals of the network, and the data plane is located along with the switches and routers.

With the separate control plane and data plan the network process can be diminished and also the international view of the network will be removed. This will help in managing the network and evolution in the network will occur along with the different levels of innovation. The control plane can also be known as the controller which will control the data plane directly through API (Application Programming Interface) and it also includes the open-flow. The controller will process the open-flow switches rather than the conventional switches. (Van Adrichem et al., 2014) The open-flow switches will have 1 or more forwarding rules which contain a subset which will match correctly with the appropriate set. With a proper match only it can perform a certain function such as modifying, forwarding, and dropping. The forwarding rules are managed by the controller with the help of protocols and applications present in it. The open-flow sometimes acts as a router, firewall, load balancer, and a switch.

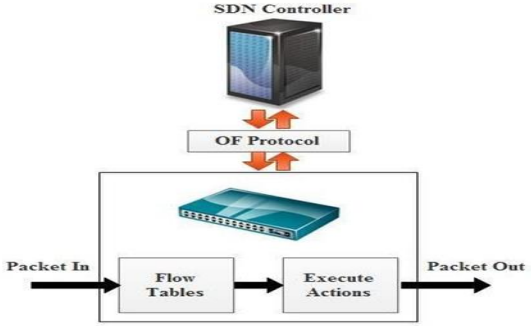

Fig. 1. Open Flow Switch.

2. Literature Survey

This chapter deals with previous study reviews based on load balancing for data centers. This chapter begins with a software-defined network, concepts, and models related to it. The difference between a Software-defined network and a conventional network will be discussed here. Later in this chapter certain information about load balancing and LB related to SDN will be discussed.

(Bannour et al., 2017) SDN is the most common and popular concept related to computer networking which helps in simplifying the control of the network, managing the network, and programs of the networks. The software-defined network can be classified into 3 layers as

1. Application layer
2. Control layer
3. Infrastructural layer

The layers of the software-defined network are illustrated in fig 1 as follows.

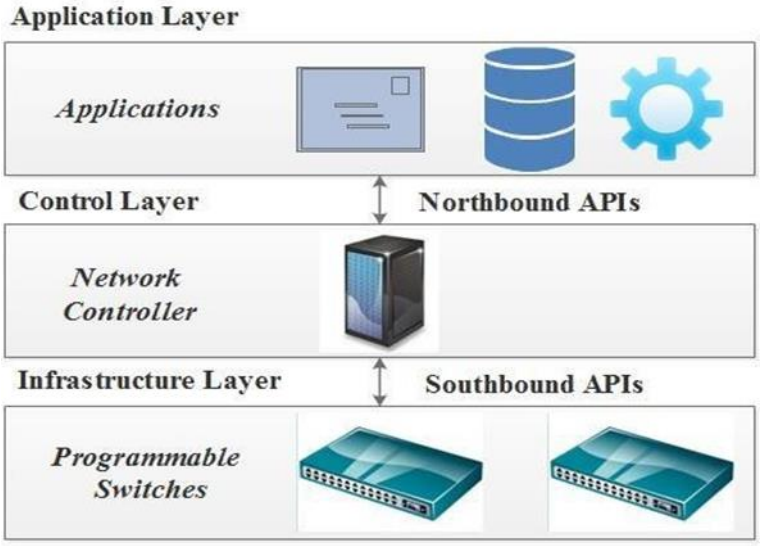

Fig. 2. Architecture of Software-Defined Network

2.1. How SDN Differs From Conventional Networking

(Hu et al., 2014) The SDN is a network typically based on software but the conventional network is based on physical properties of the network like switches and routers in order to run the network properly. In SDN the end-user can control the allocation of the resource through the layer by utilizing the control plane. There is no need for the end-user to communicate with the hardware instead of that the end-user can directly communicate with the software regarding the provisions of new devices. The administer present in the network will monitor the activity liked services and network path. The software-defined network can interact with all the networking devices. The entire software-defined network is virtualized and the abstract of the physical infrastructure will be created for monitoring or allocating the resources from the centralized phase.

In the conventional network, the control plane will be connected along with the data plane. Anyhow the control plane will be located far away due to its restricted access with the administrator. But in the case of a software-defined network, the control plane is designed with the help of software and is capable of accessing with a connected device, through which the administrator will have good control in the traffic flow from a central phase.

2.2. Understanding of Load Balancing

Cloud Computing (CC) is the newer version of the data center which is capable of giving computer services to the users. Cloud service is the one which provides the users with data center set up when needed or preferred by the user themselves. In general cloud network service will be limited to a particular location or zone. In case if the target area of the user is near then the server will complete the action needed for the user. Only certain servers can provide service for the user in such cases some servers will remain idle. If the provided service by the servers is not used properly this may also lead to the issue of processing and managing a larger set of traffic which is one of the difficult tasks to perform.

A higher level of change in the network due to traffic will create huge pressure on the data center network. In such a situation, Load Balancing is a unique technique that can be used for improving the performance of the

application and utilization of the resource. In this chapter features of load balancing and LB issues in the data center will be discussed for a better understanding of its reasons and solutions.

(Sen et al., 2013) proposed a framework considering new aspects of SDN, computer networks, issues, solutions for the problems, and challenges regarding it. Here basic software was constructed utilizing SDN and with certain tools like route-flow, mininet, onix, and verti-flow. The benefits of SDN was discussed in detail along with the implementation of SDN using mininet.

(Smara, 2018) proposed a framework by taking a survey for implementation of SDN. Here the basic concepts of SDN were focused and discussed in detail, along with its usage in day-to-day life scenarios like data center, rural connections, wireless virtual machine, internet research, etc.

3. Proposed algorithm for load balancing

(Xie et al., 2018) In the present day, the web servers experience issues and one of them is the problem of overloading. The issue of not only concerned with the happing the web serves by also on the in-house server that has a significant amount of users. Big companies and entities consider the issue of overloading to be the main concern. (Derhab et al., 2019) This issue can be mitigated through the method of load balancing that assists in the forwarding of significant traffic that is found in the server networking, sited in the clusters. In this research, we might utilize committed load balancers but the devices are expensive, which brings more scalability issues.

In the projected LB algorithm, the user request is transferred to the minimal loading server-centered on the three fundamental server loading frameworks like the preceding memories, Central Processing Unit, and connections presented. The projected lb architecture and lb management is indicated in Figure 3

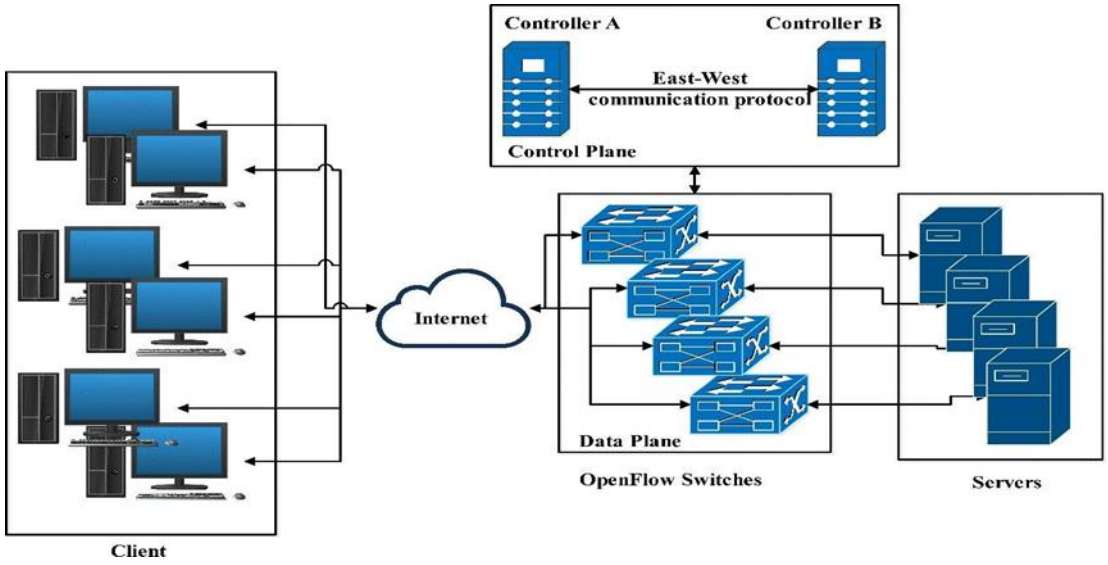

Fig. 3. The Dynamic Server Load Balancing Architecture

In this part, we have critically discussed the projected dynamic server load balancing algorithms, including its operational frameworks. The load balancing algorithms principally focus on the selection of the most vibrant server for the information process as indicated in Algorithm 1 below.

3.1 The sFlow agent algorithm

To select the most effective web server, it is relevant to evaluate three sever loading frameworks like the remaining memory capability, remaining CPU capability, and the present connection amount of the website servers. The sFlow agent gathers the three web server load parameters, which transfer information to the sFlow collector most of the time.

3.2 Controller algorithm

The sFlow collectors are situated in the controllers. It gets the load data of the relevant servers that are connected to the controllers. Utilizing the present memory capability, the remaining CPU capacity, and the present connection amount of servers; it can be possible to calculate the servers' loads in every server in the category and clusters known as balanced load, under load, overload servers. The loaded frameworks are shows by {m, c, and l} whereby 'm' shows the remaining memory capacity, whereas 'c' is the remaining CPU capacity and 'l' indicates the present connection amount. The load balancing frameworks are occasionally transferred by the sFlow agents onto the sFlow collectors that are situated in the controllers. Every user's request is executed based on the First Come-First Serve manner that implies the requests present are processed first.

Every networking framework has a varied dimension of effect on server loading. In that regard, we have presented the 'w' that shows the effect level of certain frameworks like the . This can be changed about the forms of service and evaluation of the web servers. For instance, the UDP necessitates the server application to countercheck the copied packets and the TCP on the sequent numbers that do not have any web server applications. In that regard, the UDP takes in a lot of CPU processing compared to the TP. In case we process the UDP packets, we require a significant set of weight onto the CPU. Concerning the forms of traffic, the CPU, and memory necessities, the weight is normally varied. In this case, we have utilized one level of configuration location for the web servers. So, every networking framework takes the same weights.

3.3. Algorithm 1 Dynamic Load Balancing Algorithms

sFlow agent algorithm:

Locally calculate the three load parameters m, c, and l.
m = remaining memory capacity
c = remaining CPU capacity
l = remaining connection counts
send_resource_info(m,n,l)

Controller algorithm:

sFlow Collector agent receives resource information sent by the sFlow agent.
For every server associated with the controller
Calculate the load of the server, SL as
$SL = (w_1 \times m)+(w_2 \times c)+(w_3 \times l)$
if (SL > overload_Threshold)
Server_Resource = OVERLOADED
elseif (SL < under_load_Threshold)
Server_Resource = UNDERLOADED
else
Server_Resource = BALANCELOADED
update_resource_info(server, Server_Resource)
If new request is received
For every UL_server in lookup(flow table, UNDERLOADED)
if (FCFS request queue in UL_server < queue capacity)
destination_IP = UL_server_IP
forward(new request, destination_IP)
return
For every BL_server in lookup(flow table, BALANCELOADED)
if (FCFS request queue in BL_server < queue capacity)
destination_IP = BL_server_IP
forward(new request, destination_IP)
return
Drop the request
return

Queue processing agent algorithm:

For each awaiting request in the queue
If (server has no new request && number of request in FCFS request queue > 1)
Select a request from the other server queue
Processes the request

Whenever the switches receive the unique request, it first considers the under-loaded servers in the flow tables. In case the switch identifies the under loaded server, it allows the web server for a certain request. In case it is not able to identify the under-loaded server, is located, it allows the web server for a certain request. Moreover, when the balanced loaded server is not located, it releases requests to eliminate the risk of networking congestion.

3.4. Queue processing agent algorithm

The algorithm identifies the queue in every web server has not user request identified in the queue. Moreover, it possible allots the user request from the remaining servers that are composing of more than a single request. The algorithm assists to reduce the processing time of the respective requests.

4. Results and discussion

Figure 4 shows the utility level of the CPU of the randomized, round-robin, and DServ- LB algorithm. The CPU remains to be one of the most relevant factors to evaluate the server load. In that regard, we deliberate the utility level of the CPU as the webserver framework to evaluate the web server loads. The CPU utilization of the remaining two algorithms is evaluated and differentiated with the recommended algorithm whenever the amount of requests is about 500/second. As for the round-robin and random algorithm, the CPU usage ratio of the webserver is not stable. Since the recommended DServe-LB can enable web servers to stabilize the loads more effectively in the servers, it has an effective resource planned capability to successfully eliminate unbalanced loads over the cluster servers. In that case, the general resource utilities of the networks are advanced.

Fig.4. CPU Usage Ratio of Load Balancing Strategies

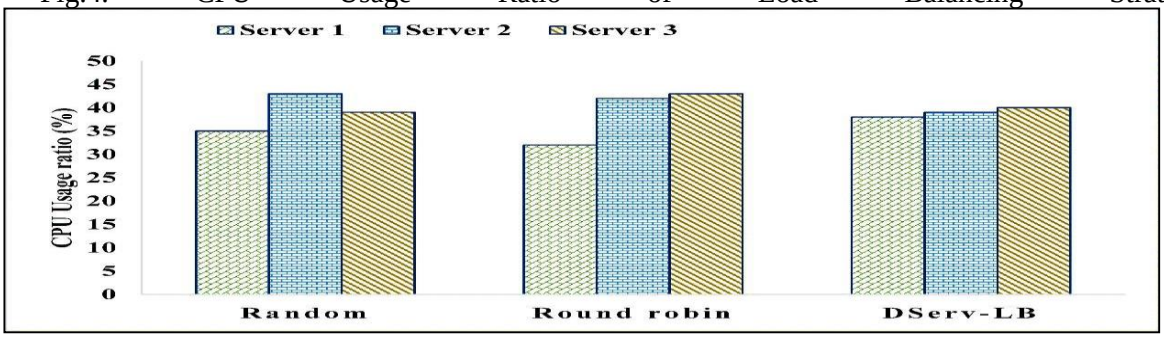

Fig.4. CPU Usage Ratio of Load Balancing Strategies

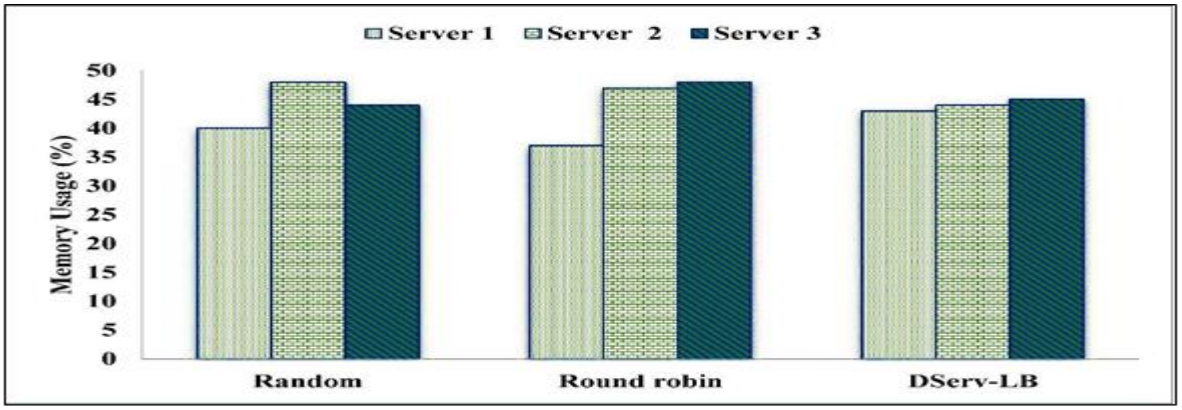

Fig.5. Memory Usage Ratio of Load Balancing Tactics

Figure 5 shows the memory of the servers and their usage based on the coinciding algorithms whenever the request rate is 500/second. As for the round-robin and random algorithms, the memory utility ratio of the webserver is never balanced because the round-robin and random algorithms do not relate to the webserver load framework for load balancing. Since the recommended DServ-LB is capable of making up a balanced sever and load servers more efficiently, the general resource usage of the networks is enhanced.

Figure 6 shows the overall throughput of the load balancing algorithm. It is evident from the general throughput that the recommended DServeLB is considered greater compared to the round-robin and random algorithms due to the three load balancing frameworks that effectively make use of the server memories.

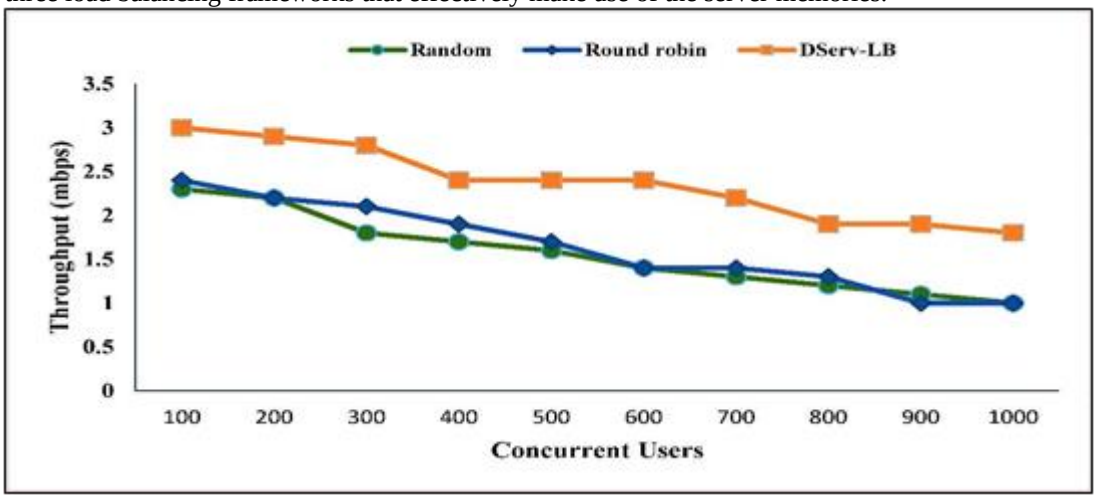

Fig.6. Server Throughput of Load Balancing Tactics

Figure 7 shows the dropping rates of the user request for the vibrant server load balancing, random, and round-robin algorithm in a single second. The x-axis shows the number of requests produced every second while the y-axis shows the rate of drop in every second. The rate of dropping of the vibrant server load balancing because of the relation of the three loading balancing frameworks that lead to the significant usage of the server resources.

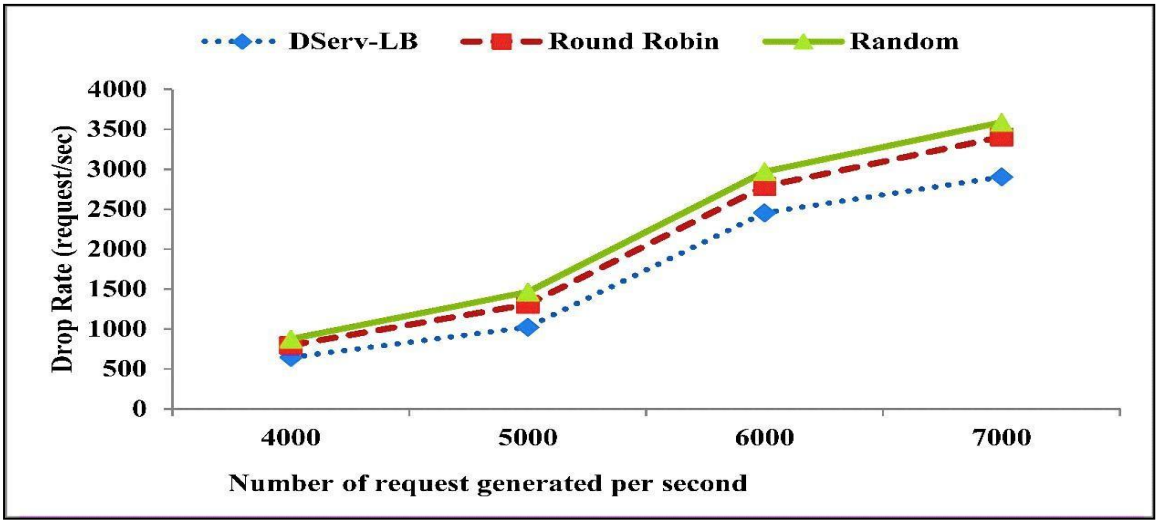

Fig.7. Drop Ratio vs No.of Requests in every second

Figure 8 shows the duration taken by the vibrant server load balancing, random, and round-robin algorithm in every second. The x-axis shows the number of requests provided per second, whereas the y-axis shows the time taken in seconds. The time of response witnessed from the graph vividly indicate that the DServ-LB consumes minimum time as differentiated from the load balancing algorithm. Because our recommended algorithm chooses the most relevant sever with extreme resources, the request can be processed most effectively.

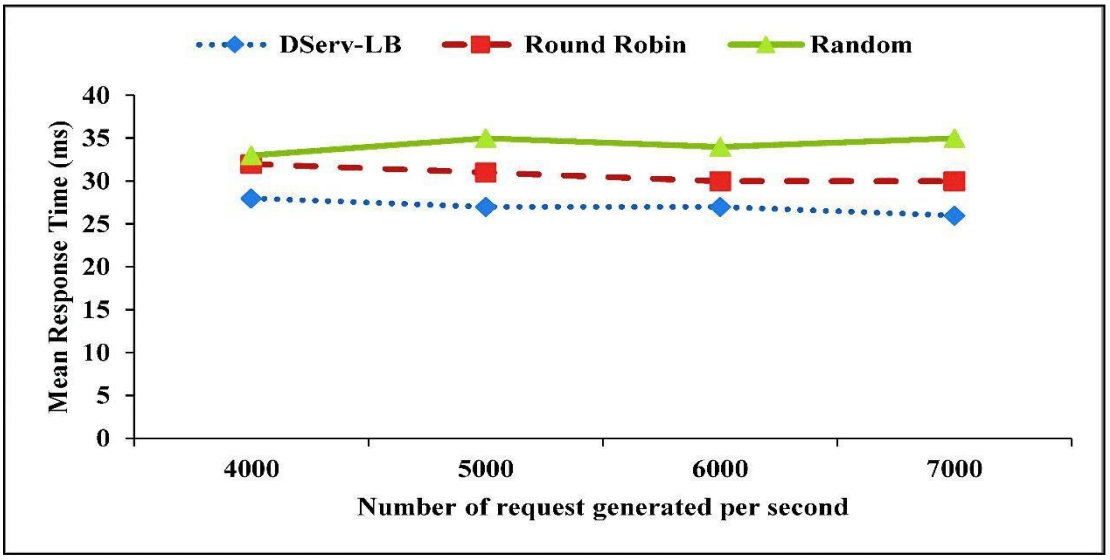

Fig.8. Mean Response Time vs. No.of Request per second

Up to the present, we differentiated our research works with other static forms of load balancing algorithms. As of now, we shall compare our recommended framework work with other vibrant load balancing algorithms known as the Load Balancing Scheme Based on the Server Response Timeframe (LBBSRT). This work has its information already included in other related research. Figure 9 indicates that our recommended algorithm effectively operates on the usage of memory due to the framework utilized to evaluate the utility of resources. On the LBBSRT, the response timeframe taken for by the load balancing is the only element under consideration, which fails to evaluate the availability of server resources.

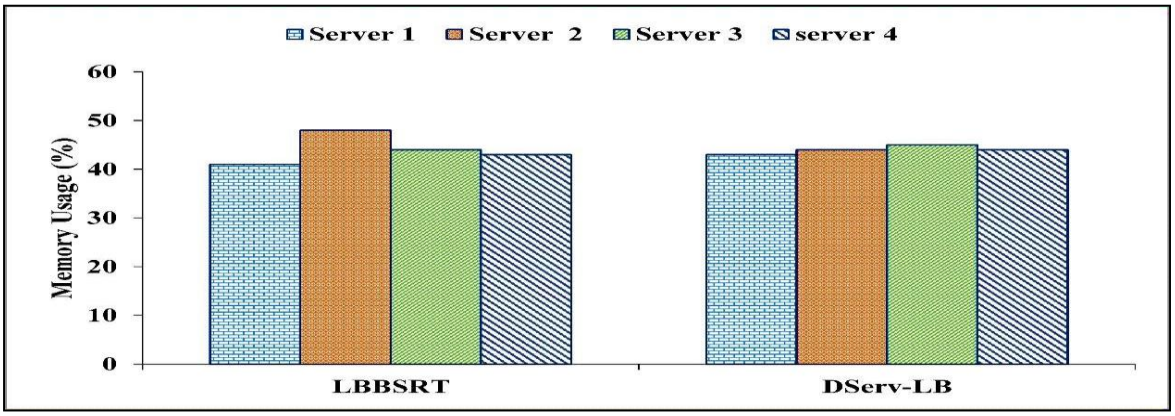

Fig.9. Memory usage contrast

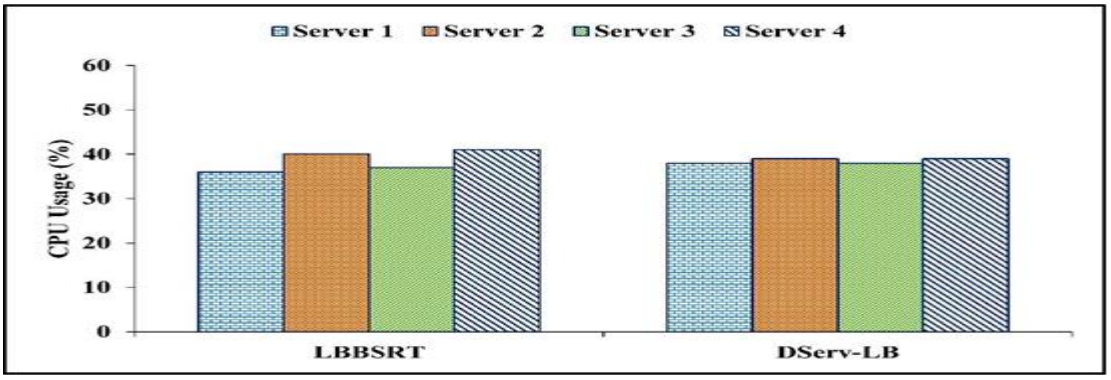

Fig.10. CPU usage Assessment

Figure 10 indicates the recommended algorithm together with the LBBSRT algorithm. As for the LBBSRT algorithm, the response timeframe is the element under consideration when evaluating the server loads. However, the servers may be composed of various sizes of resource capabilities and every request is capable of consuming various capacities of resources. In that case, the response timeframe is not enough for effective load balancing. The findings indicate our recommended algorithm, which significantly stabilizes the load, which is different from the LBBSRT algorithm.

Fig.11.The Average Interval of Single Controller and Multi-Controller

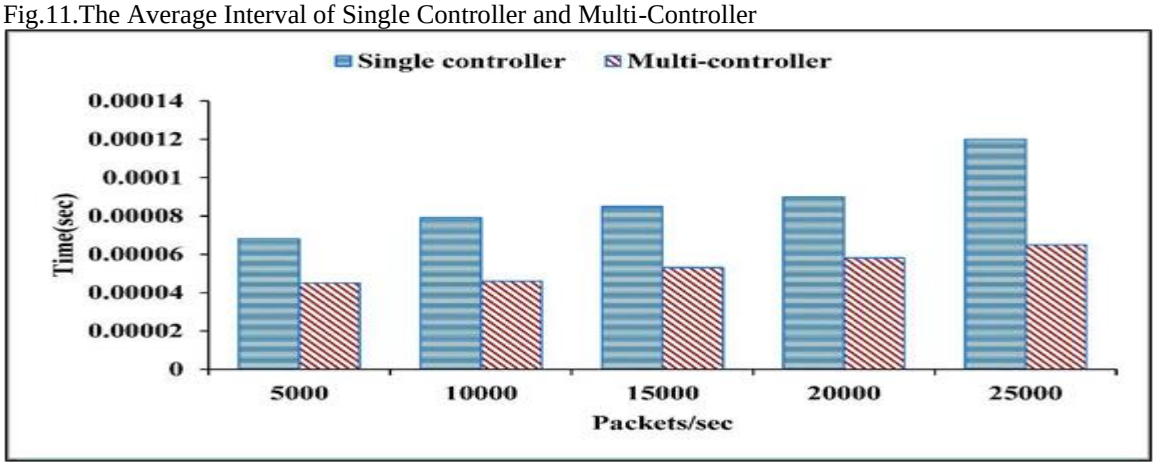

Fig.12. The Extreme Interval of Single Controller and Multi-Controller

Some evaluation concerning the single and multiple controllers to evaluate their performances. Fig 11 indicates the standard delay of one controller and multiple controller environments. Fig 12 indicates the ultimate delay of one and multiple controller set-ups. We have considered the X-axis to be the packets in a second with an increasing figure of about 5000 whereas the y-axis is the time considered per second. The findings indicate that the multiple controllers process the packet more quickly compared to one controller because the workload shared in the multiple controllers other than the one evaluated by one controller. At the start, the controller takes less time to execute the instructions and to install them into the switches as indicated in Figure 11.

6. Conclusion

This study utilized DServ-LB algorithm for load balancing which helped in reducing the overloaded and issues related to it. This was possible with the help of certain applications like sFlow agent and sFlow collector for evaluating the resource present in the network and its availability for the servers. The evaluation was based on the web server load concerning the remaining memory capability, remaining CPU capability, and the present connection amount. The requests have been transferred to the web servers that have a maximum service capability. The vibrant server load-balancing algorithm is contrasted against the present load balancing algorithm and the simulation findings show that our recommended load balancing server computes the balancing algorithms on servers. The drop rate of the vibrant server load balancing algorithms is diminished whereas the throughputs are advanced. Whenever the drop rates are reduced, the server provides a huge throughput. In that case, it possibly processes more user requests, CPU resources, and memory resources, which are used evenly. The DServ-LB can be altered by certain factors like knowing the traffic pattern in LB, pack priorities, and QoS restrictions. This work can be further extended by analyzing the proposed algorithm for a broader data center networking system by using different configuration techniques.

References

1. Sen, Siddhartha, et al. "Scalable, optimal flow routing in datacenters via local link balancing." Proceedings of the ninth ACM conference on Emerging networking experiments and technologies. 2013.
2. Smara, Mounya. Fault tolerance in embedded systems: cloud computing systems. Diss. 2018.
3. Xie, Junfeng, et al. "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges." IEEE Communications Surveys & Tutorials 21.1 (2018): 393- 430.
4. Derhab, Abdelouahid, et al. "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security." Sensors 19.14 (2019): 3119.
5. Bannour, Fetia, Sami Souihi, and Abdelhamid Mellouk. "Distributed SDN control: Survey, taxonomy, and challenges." IEEE Communications Surveys & Tutorials 20.1 (2017): 333-354.
6. Huang, Tao, et al. "A survey on large-scale software defined networking (SDN) testbeds: Approaches and challenges." IEEE Communications Surveys & Tutorials 19.2 (2016): 891-917.
7. Hu, Fei, Qi Hao, and Ke Bao. "A survey on software-defined network and openflow: From concept to implementation." IEEE Communications Surveys & Tutorials 16.4 (2014): 2181-2206.
8. Van Adrichem, Niels LM, Christian Doerr, and Fernando A. Kuipers. "Opennetmon: Network monitoring in openflow software-defined networks." 2014 IEEE Network Operations and Management Symposium (NOMS). IEEE, 2014.
9. Niven-Jenkins, B., et al. "Requirements of an MPLS transport profile." (2009).
10. Curtis, Andrew R., et al. "DevoFlow: Scaling flow management for high-performance networks." Proceedings of the ACM SIGCOMM 2011 conference. 2011.
11. Akyildiz, Ian F., et al. "A roadmap for traffic engineering in SDN-OpenFlow networks." Computer Networks 71 (2014): 1-30.
12. Milani, Alireza Sadeghi, and Nima Jafari Navimipour. "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends." Journal of Network and Computer Applications 71 (2016): 86-98.
13. Pao, Tsang-Long, and Jian-Bo Chen. "The scalability of heterogeneous dispatcher-based web server load balancing architecture." 2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06). IEEE, 2006.
14. Zhang, Lin, Li Xiao-Ping, and Su Yuan. "A content-based dynamic load-balancing algorithm for heterogeneous web server cluster." Computer Science and Information Systems 7.1 (2010): 153-162.
15. Zhong, Hong, Yaming Fang, and Jie Cui. "Reprint of "LBBSRT: An efficient SDN load balancing scheme based on server response time"." Future Generation Computer Systems 80 (2018): 409-416.
16. Kaur, Sukhveer, and Japinder Singh. "Implementation of server load balancing in software defined networking." Information Systems Design and Intelligent Applications. Springer, New Delhi, 2016. 147-157.
17. Singh, Harikesh, and Shishir Kumar. "WSQ: web server queueing algorithm for dynamic load balancing." Wireless Personal Communications 80.1 (2015): 229-245.
18. Xiaolong, X. U., et al. "MTSS: multi-path traffic scheduling mechanism based on SDN." Journal of Systems Engineering and Electronics 30.5 (2019): 974-984.
19. Al-Fares, Mohammad, et al. "Hedera: dynamic flow scheduling for data center networks." Nsdi. Vol. 10. No. 8. 2010.
20. Guo, Zhiyang, Jun Duan, and Yuanyuan Yang. "On-line multicast scheduling with bounded congestion in fat-tree data center networks." IEEE Journal on Selected Areas in Communications 32.1 (2013): 102- 115.

21. Cui, Wenzhi, Ye Yu, and Chen Qian. "DiFS: Distributed Flow Scheduling for adaptive switching in FatTree data center networks." Computer Networks 105 (2016): 166-179.
22. Pacheco, Fannia, et al. "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey." IEEE Communications Surveys & Tutorials 21.2 (2018): 1988- 2014.
23. Curtis, Andrew R., et al. "DevoFlow: Scaling flow management for high-performance networks." Proceedings of the ACM SIGCOMM 2011 conference. 2011.
24. Tang, Feilong, et al. "An efficient sampling and classification approach for flow detection in SDN- based big data centers." 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). IEEE, 2017.
25. Dhaliwal, Harpreet Kaur, and Chung-Horng Lung. "Load balancing using ECMP in multi-stage Clos topology in a datacenter." 2018 IEEE Conference on Dependable and Secure Computing (DSC). IEEE, 2018.