Date of publication SEP-30, 2023, date of current version MAY-25, 2023.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.3.3237

Leveraging Software-Defined Networks for Load Balancing in Data Centre Networks using Linear Programming

VANI KURUGOD ASWATHANARAYANA REDDY, RAMAMOHAN BABU KASTURI NAGAPPASETTY

Department of Information Science and Engineering
Dayananda Sagar College of Engineering, Bangalore, Karnataka, India

Corresponding author: Vani Kurugod Aswathanarayana Reddy (e-mail: 16vaniram@gmail.com).

ABSTRACT A rapid increase in the number of online applications has led to exponential growth in traffic. In data centers, it is hard to dynamically balance such huge amounts of traffic while keeping track of server data. A load-balancing strategy is an effective solution for distributing such huge amounts of traffic. The major contribution of this research work is to improve the performance of the network by designing a dynamic load balancing algorithm based on server data using SDN, reduction of controller overhead and optimizing energy consumption in a server pool. The problem is formulated using a Linear Programming mathematical model. In order to demonstrate the effectiveness and feasibility of the proposed technique, the experimental setup is deployed using real hardware components such as a Zodiac-Fx switch, Ryu controller and various web servers in the data center network. This proposed scheme is compared with round-robin and random load balancing mechanisms. The experimental results show that the performance is improved by 87.4% while saving 78% of the energy.

KEYWORDS Software-defined networking; Quality of Service; Load balancing; Open Flow; Data Centre.

I. INTRODUCTION

IN the era of information systems, there is a rapid shift in network traffic. The requirements for quality of service and network requirements have changed dramatically, placing more emphasis on end-to-end goals in data centers. A wide range of studies have shown that optimizing the performance of data center networks (DCNs) is an essential factor[1][2][3][4][5].Quality of service is measured by various metrics like service availability, bandwidth, server utilization, latency, end-to-end delay, resource allocation, scalability, energy consumption and many more. One of the most important considerations in sustaining QoS is ensuring the vitality of time-sensitive applications. Load balancing plays a crucial role in achieving a high quality of service and distributing network traffic evenly. Any user experience is measured in terms of service availability, which is significantly influenced by the level of load balancing achieved. For example, imagine an e-commerce site that is widely used for online shopping. On regular days, a minimum number of servers may be necessary to accommodate the number of users accessing the site. Over the festive season, the number of users accessing the site will increasev drastically due to various offers.

If the servers cannot handle all the requests, in such scenarios, load balancing can be employed[6][7].

Load balancing can be performed at two levels. One is at the transport layer and the other one is at the application layer. Most data centers employ hardware load balancers. Dedicated hardware load balancers are prohibitively expensive to maintain. As a part of this research work, the concept of software-defined networking (SDN) was integrated to tackle the issue of load balancing. SDN allows programmability of network components, which makes it adaptable in every field of networking[8].

As illustrated in Figure 1, the data plane and control plane are decoupled in SDN. The entire intelligence in SDN is located in the control plane. The infrastructure layer, also called the data plane, consists of forwarding elements like switches, routers, and many more. The controller in the control layer acts as a decision maker, whereas data plane elements follow the controller's instructions. The next layer is SDN's application layer, which provides users with a wide array of application programming interfaces for creating customized development modules based on specific business needs[9][10][11].

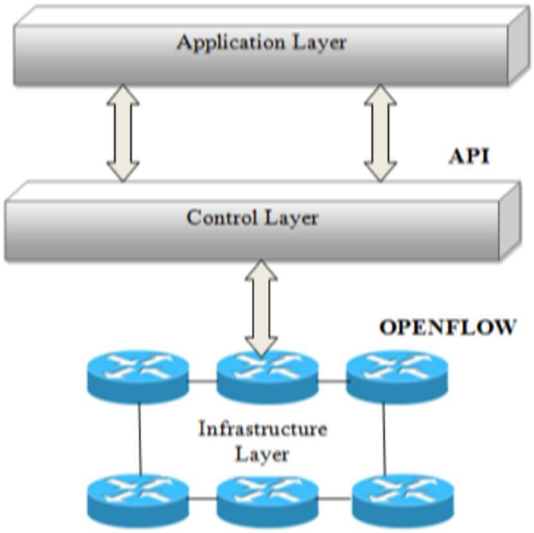

Figure 1. The SDN Architecture

The article presents a Server Metric Collection Load Balancing [SMC-LB] algorithm based on SDN. The proposed work consists of load balancing module that runs server metrics collector component and the controller module that runs best server allocator and flow installer components. The major contribution of the proposed research work is to improve the performance by adopting SDN while performing load balancing among the pool of servers by increasing the number of requests served, reducing controller overhead and optimize server energy consumption.

The remainder of the paper is organized as follows: Section 2 reports on related work. Section 3 describes the proposed system model. Section 4 includes implementation and performance analysis of the proposed method. Section 5: Conclusion and Future Enhancements.

II. RELATED WORK

The cutting-edge of this research work is employing SDN paradigm to provide load balancing and hence improve network QoS. The load balancing techniques are classified into static and dynamic algorithms[12][13][14].

A. STATIC LOAD BALANCING ALGORITHMS

A static load balancing strategy is based on a fixed set of rules that do not depend on the current condition of the network. On the other hand in case of sudden system failures, these algorithms have a serious disadvantage. Some of the static load balancing techniques are round robin, random technique, weighted round robin, least connection; equal-cost multi-path routing protocol, and many more. The round robin assigns new job to each switch in a round robin fashion. On each switch, job allocation order is maintained locally. This algorithm works well when the workload distributions are equal. Hence, these technique may not suit for current data center networks[15][16][17].Equal cost multi-path routing (ECMP) is used to split flows towards the available paths as analyzed in[18]. Based on the hash value of the flow they are forwarded to different paths. For the current network demands, the mapping between flows and paths is not contributing towards utilizing bandwidth. ECMP does not take into account dynamic addition of flows. In random load balancing technique, each job is assigned randomly to the pool of servers. In least connection algorithm, the jobs are scheduled to the server with least number of active connections. The achievement of load balancing through traditional techniques not only entails high costs but also presents complex implementation challenges. To overcome these drawbacks dynamic load-balancing techniques were designed.

B. DYNAMIC LOAD BALANCING ALGORITHMS

These algorithms perform load balancing based on current state of the system. Dynamic load balancing offers low overhead, increased scalability, and fault tolerance. These load balancing techniques lead to improvement in performance of the entire network. With the flow control mechanism in SDN, it is now possible to build a dynamic load balancing at the software plane [19].

Axiomatically, an SDN load balancing scheme based on server response time is proposed in this scheme. It has a single controller and is used to acquire response time for the selection of a server with a minimum response time. The work proposed by the author in[20] performs web load balancing using Open Flow switches in software-defined networking, takes server response time and switch port traffic for performing load balancing among a pool of servers. The research in [21] suggested a dynamic server load-balancing algorithm using the sFlow protocol. A dynamic load balancing mechanism that ensures service quality was suggested in [22]. Another strategy of message-level scheduling and flow-level scheduling performed in overburdening on a given access point (AP) is addressed in[23]. To solve the problem of load balancing in cloud data centers, a new dynamic approach to dynamic load balancing routing in Open Flow enabled networks is used. A path switching algorithm was designed that is capable of balancing the workloads dynamically in the networks during transmission, as discussed in[24]. The work proposed in[25] is intended to design an algorithm that re-routes the traffic to an alternate path from the original path when the throughput decreases or data loss reaches a certain threshold.

In order to address energy-efficient load balancing, an algorithm that ensures energy-efficient resource management in cloud data centers was created in [26]. Based on the SNMP protocol, a server load balancing scheme is designed in [27] that addresses the scheduling of connections to the servers based on different metrics. The literature on green networking has been widely studied and a variety of solutions have been offered. SDN also lends itself to most of these pieces. For instance, the authors in [28] presented the GreenSDN approach that makes use of load profile and linear profile for optimizing the total energy efficiency. The research work in[29] presented the Time Efficient Energy Aware Routing (TEAR) algorithm. This research aims to reduce the number of links used for packet delivery in order to reduce energy consumption. The study conducted in [30] shows the usage of technologies like SDN, virtualization, and edge computing to optimize the energy consumption in data center networks. The research work in [31] proposed a survey on an energy-efficient network configuration in SDN predicted using a Machine Learning method based on Logistic Regression. The end-to-end mobility support required to maintain service continuity and quality of service using stochastic network calculus (SNC) framework to control Mobile edge computing (MEC) data flows was examined in [32]. In the realm of IoT implementations, a multitude of challenges arise, encompassing the effective

management of substantial network data, the establishment of robust privacy and security measures, the fulfilment of demanding Quality of Service (QoS) criteria, and the adept handling of the heterogeneous nature of underlying networking components. To address the problem of energy efficiency the research work in [33] centers around an in-depth examination of allocating Virtual Machines (VMs) to end devices using Weighted Sum Method.

Based on preliminary research, it appears that using SDN for network management and load balancing has potential benefits. However, only a few approaches address controller overhead. In addition, some systems have trouble comprehending packet headers, resulting in difficulties in routing traffic. Another significant factor during performance evaluation is taking energy consumption by the server pool into consideration. The proposed research work is motivated by the design of a technique named the Server Metric Collection Load Balancing [SMC-LB] algorithm to reduce the controller overhead and achieve optimization of server energy consumption.

The rest of the paper is structured as follows: Section 3 describes the proposed system model. Section 4 includes implementation and performance analysis of the proposed method. Section 5 includes Summary and proposes Future Improvements.

PROPOSED SYSTEM MODEL

The proposed model is built in the Open Flow environment. The model is as depicted in Figure 2. It consists of the RYU controller, load balancer, web server's pool, and clients connected together via Zodiac-Fx switches.

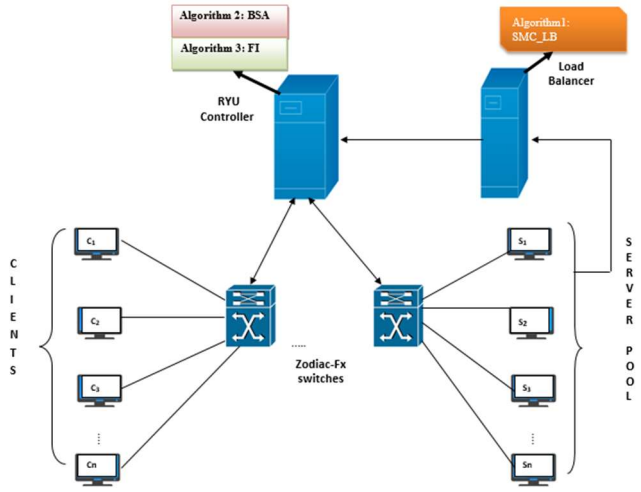

Figure 2. SDN based System Model

The proposed scheme performs dynamic load balancing using three different algorithms. Algorithm1 runs on a load balancer and collects server metrics from each server within a window period of 5ms as per the assumption. Algorithm 2 runs on the controller to decide the best server based on load balancer results. On the controller, Algorithm 3 installs flow into the zodiac switch. Upon the request's arrival, the Zodiac-Fx searches for the flow. If the flow is not defined, it is forwarded to the controller. If the flow is defined, the switch forwards the request to the best server concerned. The concerned server responds directly to the request. During load balancing, the load balancer takes care of communications related to server status. Nevertheless, the load balancer is separated from the controller logic, and so controller overhead is reduced.

A. MEASURING SERVER METRICS

This module outlines the strategy for obtaining server metrics. The start-up server scripts running on each server are responsible for sending different metrics like requests_per_sec (RPS), time_per_request, transfer_rate, waiting_time, CPU_TIMES, and energy consumption [EC] to the RYU controller. The proposed model is evaluated for different cases in experimentation. Algorithm 1 depicts the process of measuring server metrics. The symbols are listed in Table 1 below.

Algorithm 1: Server Metrics Collector [SMC]

```
Input: sends server metrics for each 5milli seconds
Output: Collects server metrics

for each server do
    {
     if time% t ==0 do
      {
       Send HTTP request to server record metrics
         Rps, Ec
       Rps = avg [(no. Of request) / (time taken per
        request in 5ms)]
       Ec= avg (Gc)
         Gc= Oc+Xc
         for each Wtime = 5ms
                    {
                    send Rps and Ec to load
                    the balancer
                    }
      } end for
  } end if
} end for
```

Algorithm 2: Best Server Allocator [BSA]

```
Input: max Rps and Min energy consumption
Output: Finds the best server
    While start up load balancer do
    {
       Collect the metrics Rps and Ec
       Record max (Rps) and min (Ec) each 5ms
       from each  server.
     for each server.
      {
            If (dpkt=Creq)
              {
                    route the traffic to server
                    with max
                (Rps) and min (Ec)
            } end if
      } end for
} end while
```

Algorithm 3: Flow Installer [FI]

```
        Input: max (Rps) and min (Ec)
        Output: Sends load information to RYU
                controller
1   send best_server metrics to RYU controller
2                       {
3       get current best server information from load
        balancer
4       add_flow to zodiac
5                       }
6                       change_destination_address
7                       {
8       for each HTTP connection
9                           {
10                          Destip = LBip
11                          add flow in the switch.
12                          }end for
14      Flow of packets via Zodiac-Fx
15                      for each incoming HTTP request
16                          {
17                          if (match =true)
18                          {
19                                  follow the flow
20                                  else
21                                  forward the packet to
                                    The RYU controller
22                          } end if
23          } end for
```

Table 1. Symbols and Description

Symbol	Corresponding Description
Rps	Requests per second
Ec	Energy consumption
Gc	Global energy consumption
Oc	Operating system power
Xc	Energy consumption of applications.
Nr	Total Number of requests
Cc	Concurrent connections
Ts	Time taken per requests
Rps_i	Max number of requests served in window period
Ec_i	Initial energy consumption
Rps_f	Average of Rpsi
Ec_f	Average of Eci
Ecrr	Energy consumption in round-robin
Ecrand	Energy consumption in random
Rt	Response Time of all requests

The performance evaluation and the experimental results are discussed in the next section.

IV. EXPERIMENTATION

The experimental setup is created by connecting web servers like Apache, SimpleHTTPServer, and NGINX installed on Ubuntu machines in a data center environment. All the servers, clients, and load balancers are connected to the Ryu controller[34] via Zodiac-Fx switches[35]. To generate traffic from different hosts, the Apache Bench (ab) tool is used. The necessary assumptions and descriptions are given in Table 2 below.

Table 2. Assumptions and Corresponding Description

Assumptions	Description
Time = 5ms Cc=10 to 2000	The window period for server metric collection is assumed every 5 milliseconds
Nr = 300 to 400 Nr = 150 to 200 Nr = 10 to 50	High Average Low

A. TESTBED FOR IMPLEMENTATION

The experimental setup is depicted in Figure 3. The servers from hs1 to hsn are connected to the controller and load balancer via Zodiac-Fx switches. The clients from hc1 to hcn connected to the controller via zodiac send HTTP requests to the server. In modern web servers, it is common practice to utilize persistent connections, where a single TCP connection is employed to handle multiple HTTP requests. Here TCP connection is often utilized as a flow for transmitting data in network communication.

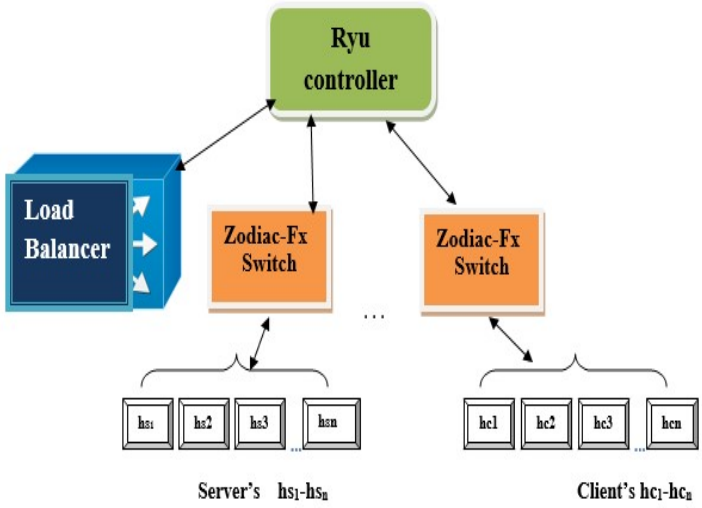

Figure 3. Experimental topology

B. Mathematical modelling.

The proposed method is modelled using a linear programming mathematical model. A linear programming problem typically involves finding an extreme value for a linear function. This linear function can either be used to maximize profit or revenue, or to minimize costs. To achieve optimization in the proposed method the scheduling of tasks is represented as a linear function. The objective of linear programming model in the proposed method is to maximize the number of requests served by each server. The objective function is represented as shown below:

$$f(x) = \sum_{i=1}^{n} (NX_{ij} + C_{ij}) \quad \text{.... (1)}$$

Subject to

$$f(x) = (\sum_{I=1}^{n} [NX_i + C_i] + \sum_{i=1}^{n} [NX_j + C_j] \leq X(\max)$$
$$\text{X(total)} = \max(\sum_{i=1}^{n} X_i + \sum_{j=1}^{n} X_j) \quad \text{.... (2)}$$

Non-negativity constraints:

$$x_i, x_j \geq 0,$$

where

N = numbers of users
C_i= concurrency of i-th server
C_j= concurrency of j-th server
X_i=Request served by i-th server
X_j=Request served by j-th server
X (max) = Max Requests served per second
X (total) = total requests served by each server

Substituting the values of concurrency C_{ij}, N and X_{ij} shows us the linear growth in the result, though the number of users increases along with the increase in concurrency.

C=10-2000	N= 1000,2000,5000	X=400,300,200,100,50,10

When x=400
$f(x)$ = (1000*400) +10
$f(x)$ = 400010

Similarly the variation of $f(x)$ is shown in the graph below for different values of C_{ij}, N and X_{ij}.

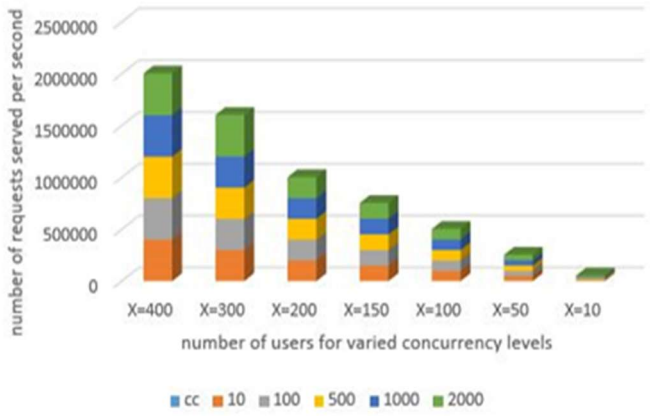

Figure 4. Linear growth of requests served per second

The experimentation was carried out as discussed below.

The switch forwards the client's HTTP request to the controller as soon as it is received. The controller installs the flows according to the metrics received from the load balancer. Based on the flow, the switch forwards the packet to the particular best server with max (Rps) and min (Ec). Here, to calculate energy consumption, a python-based Running Average Power Limit [pyRAPL] is used[36].

The experimentation was carried out as discussed below.

Stage 1: The apache bench tool is installed on the client machines for performance evaluation. Using apache bench, each client sends the HTTP requests to the webserver.

Stage 2: Initially, this experiment was carried out using the RR algorithm, where each client's sends HTTP requests to the server, and these requests will be handled by servers in turns, i.e., the first request will be handled by server K ($\boldsymbol{S_k}$), the next request will be handled by server (K+1) mod M, where M represents the number of servers in the server pool. Here, Rps is obtained as shown in equation (3).

$$\boldsymbol{Rps} = \left(\frac{\boldsymbol{N_r}}{\boldsymbol{T_s}}\right)\boldsymbol{S_k} \quad \ldots. (3)$$

Stage 3: The evaluation is done using a random algorithm. This technique chooses the servers randomly. Upon arrival of a client's request, the Zodiac-Fx switch forwards it to the controller. Now the controller allocates a random server to process the request by configuring the flow table with the selected server. Here, Rps is obtained as shown in equation (4).

Stage 4: The experimentation is carried out using SMC-LB scheme that works as described in algorithm 1, 2 and 3. To increase the system performance, the multi-threading concept is used in the RYU controller by creating different threads for different tasks. Due to the employment of multi-threading, the RYU controller receives the statistics more rapidly and installs the flows accordingly. The proposed method considers keep-alive extensions that enable persistent connections to provide long-lived HTTP connections, allowing multiple requests to be sent over the same TCP connection. According to equation (2), the initial $\boldsymbol{Rps_i}$ is calculated by considering the server, which processes the total number of requests ($\boldsymbol{N_r}$) within a window period of 5ms($\boldsymbol{T_s}$). Using equation (5) average $\boldsymbol{Rps_f}$ is obtained.

$$\boldsymbol{Rps_i} = \boldsymbol{max}\left\{\left(\frac{\boldsymbol{N_r}}{\boldsymbol{T_s}}\right)\boldsymbol{S_k}\right\} \quad \ldots. (4)$$

$$\boldsymbol{Rps_f} = \boldsymbol{avg}(\boldsymbol{RPS_i}) \quad \ldots. (5)$$

The performance (P) of round robin, random and SMC-LB is evaluated as given in equation (6)

$$\boldsymbol{P} = \left(\boldsymbol{Rps_f} - \boldsymbol{Rps_i}\right) * \boldsymbol{100} \quad \ldots. (6)$$

The Average Response Time (ART) is calculated as given below in equation (7)

$$ART = (\Sigma(\text{Response Time of all requests})) / (\text{Total Number of requests})$$

$$ART = (\Sigma(Rt)) / (Nr) \quad \ldots. (7)$$

V. RESULTS

According to the experiment conducted in SDN networks, SMC-LB is proposed and round robin and random techniques are compared as shown in Table 3.

Table 3. Comparative Results of Three Different Techniques

No. of requests	Concurrency level[cc]	Techniques and Observations		
		Round-robin	Random	SMC-LB
1000	10	High	High	High
	100	High	High	High
	500	Average	Average	High
	1000	Depleted	Depleted	High
	2000	Depleted	Depleted	High
2000	10	Average	Average	High
	100	Low	Low	High
	500	Low	Low	High
	1000	Depleted	Depleted	High
	2000	Depleted	Depleted	High
5000	10	Low	Low	High
	100	Low	Low	High
	500	Low	Low	High
	1000	Depleted	Depleted	High
	2000	Depleted	Depleted	High

Based on the experimental results, it is clear that the number of requests served by round robin and random

algorithms started to deteriorate as the (cc) reached 1000. From these experimental values, it is observed that this algorithm performs 87% better. In comparison with round robin and random techniques, the number of requests served per second by the proposed method is improved by 25.37% and 33.23%, respectively.

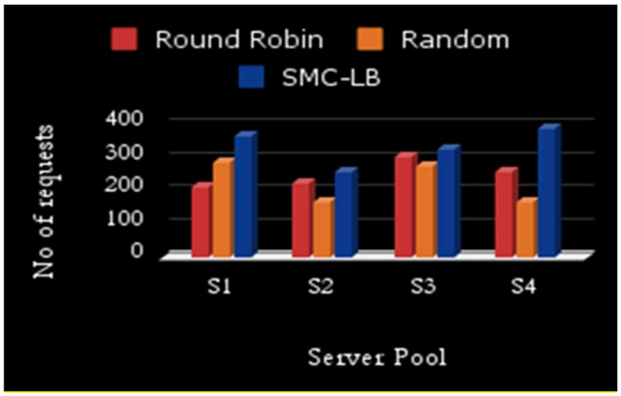

Figure 5. max Rps at cc=10 & n=1000

Figure 6. max Rps at cc=100 & n=1000

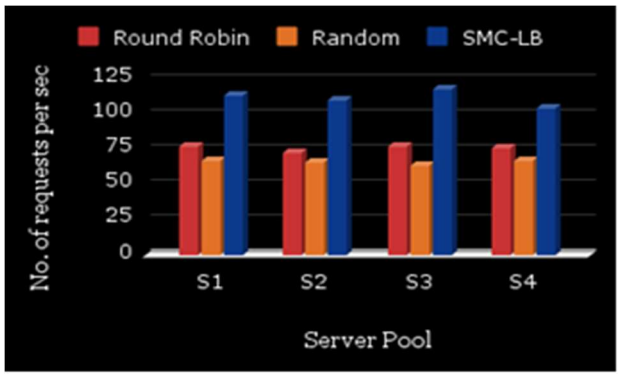

Figure 7. max Rps at cc=500 & n=1000

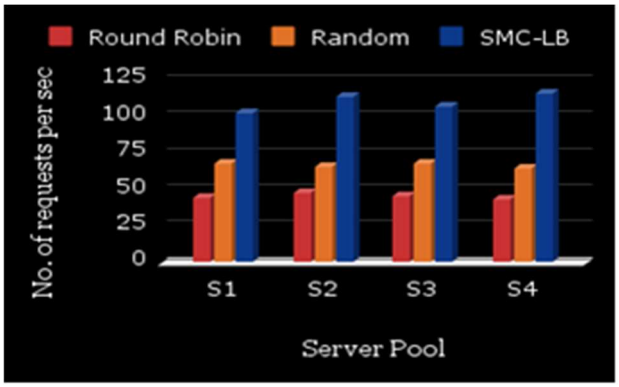

Figure 8. max Rps at cc=1000 & n=1000

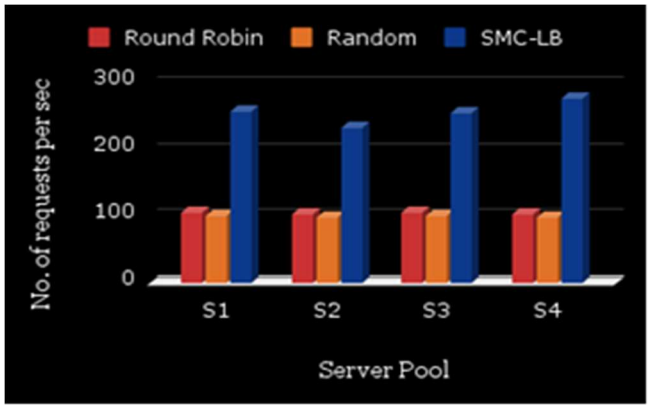

Figure 9. max Rps at cc=10 & n=2000

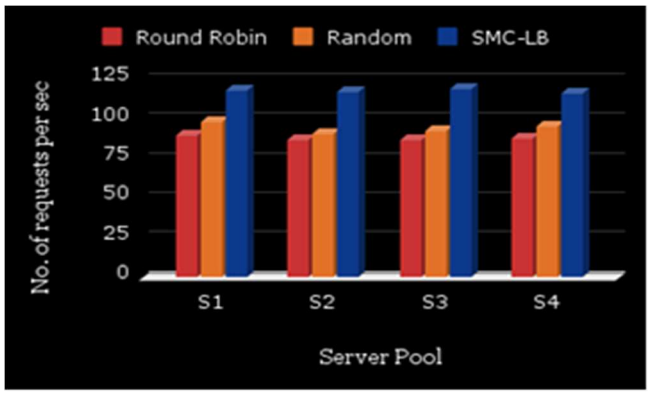

Figure 10. max Rps at cc=100 & n=2000

Figure11. max Rps at cc=500 & n=2000

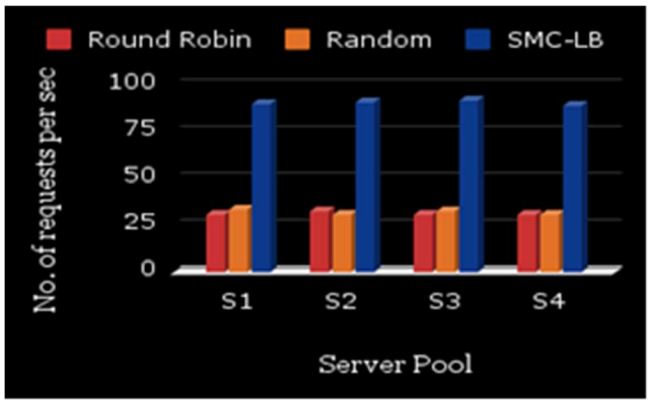

Figure 12. max Rps at cc=1000 & n=2000

A. SERVER ENERGY CONSUMPTION

Current networking technology consumes a significant amount of energy, which reduces its efficiency. In order to optimize the energy consumption in the proposed technique, the traffic is diverted to a server, which can handle a greater number of requests within less time and less energy. To achieve this optimization, the server is put into sleep mode whenever it is not serving any requests, so the energy consumption is reduced. This intelligence is the key factor for the SMC_LB algorithm to perform better than round robin and random techniques. The energy consumption $\boldsymbol{E_c}$ and $\boldsymbol{G_C}$ of round robin and random techniques is obtained based on equations (8) and (9). The energy consumption $\boldsymbol{E_{C_f}}$ of SMC-LB is given by equation (10) respectively.

$$E_c = G_c \quad \ldots. (8)$$

$$\boldsymbol{G_C = \sum_{i=0}^{n}(O_c + X_c)} \quad \ldots. (9)$$

$$\boldsymbol{E_{C_f} = avg(E_{C_i})} \quad \ldots. (10)$$

The covariance is calculated among the varying energy levels of all three schemes as given in equation (11). The term (Xi) represents data value of Rps_{rr} and Rps_{rand}, $\bar{X}$ represents mean of Rps_{rr} and Rps_{rand}, (Yi) represents data value of $Rps_{SMC\text{-}LB}$, $\bar{Y}$ represents mean of Rps SMC-LB.

$$\boldsymbol{cov(X,Y) = \sum_{i=1}^{n}\left(\frac{(X_i-\bar{X})-(Y_i-\bar{Y})}{(N-1)}\right)} \quad \ldots. (11)$$

Based on obtained values in Table 4, the energy consumption of servers using the SMC-LB is lower in comparison with other techniques. This is due to the efficient way of diverting the traffic to a more stable server, which can handle more number of requests within less time. The covariance in equation (11) is calculated to find out the relation between the random energy levels. According to the observation, as the numbers of concurrent users increased, the energy consumption also spiked in round robin and random schemes. The graphical representation of this is given in Figure 12.

Table 4. Energy Consumption Table

C	N	Ec_{rr}	Ec_{rand}	$Ec_{SMC\text{-}LB}$
10	1000	5.67	5.78	5.22
100	1000	9.02	9.5	8.89
500	1000	11.10	11.67	9.98
1000	1000	16.02	16.78	15.45
10	2000	7.66	7.56	7.03
100	2000	12.32	12.65	12.02
500	2000	17.85	17.83	17.10
1000	2000	23.98	23.89	23.04

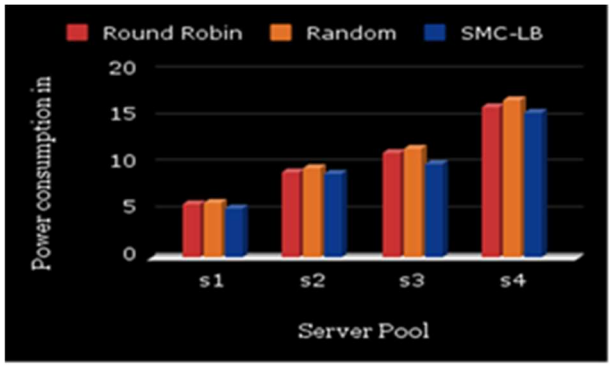

Figure 13. Server energy consumption

In server1, the energy consumption using this SMC-LB scheme is lower in comparison with round robin and random techniques. This is due to the optimized power consumption technique applied in the scheme. Whenever the server is not serving any requests, it is put to sleep mode hence the energy consumption is kept minimum. This intelligence is the key factor for the SMC_LB algorithm to save energy of server during its idle time and perform better than round robin and random techniques. It is observed from the experimental results that the proposed scheme performs 78% better. In comparison with round robin and random techniques the proposed method shows improvement in saving energy by 5.7% and 9.6 % respectively.

VI. CONCLUSION

Due to the rapid growth of network traffic in a data center environment, it is a censorious issue to balance the incoming requests and divert them to the right server in a server pool. In this research work, a more dynamic load-balancing scheme is presented based on the SDN architecture in data centers. It is observed from the results that the usage of random and round robin techniques is not feasible in scenarios where dynamic load balancing is required. To address this issue, the proposed technique SMC-LB performs better. The comparative study demonstrates that the adoption of the proposed SMC-LB scheme in any data center network achieves better throughput. It shows better performance in handling a greater number of requests than round robin and random schemes as the results shows 87.4% improvement in performance while saving 5.7% to 9.6% of energy by reducing the controller overhead. Currently, this method is implemented using web servers, but as a future enhancement, heterogeneous servers could be explored.

References

[1] G. Kumar *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," *Proceedings of the ACM SIGCOMM*, 2020, pp. 514–528. https://doi.org/10.1145/3387514.3406591.

[2] A. Saeed *et al.*, "Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates," *Proceedings of the 2020 Annu. Conf. ACM Spec. Interes. Gr. Data Commun. Appl. Technol. Archit. Protoc. Comput. Commun. SIGCOMM 2020*, pp. 735–749, 2020, https://doi.org/10.1145/3387514.3405899.

[3] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenter networks," *IEEE/ACM Trans. Netw.*, vol. PP, no. January, pp. 1–15, 2021, https://doi.org/10.1109/TNET.2021.3119986.

[4] T. Zhang *et al.*, "Rethinking fast and friendly transport in data center networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2364–2377, 2020, https://doi.org/10.1109/TNET.2020.3012556.

[5] G. Zeng *et al.*, "Congestion control for cross-datacenter networks," *Proceedings of the Int. Conf. Netw. Protoc. ICNP*, vol. 2019 October, no. January, 2019, https://doi.org/10.1109/ICNP.2019.8888042.

[6] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, 2017, https://doi.org/10.1016/j.jnca.2016.12.019.

[7] M. M. Tajiki, B. Akbari, and N. Mokari, "Optimal Qos-aware network reconfiguration in software defined cloud data centers," *Comput. Networks*, vol. 120, pp. 71–86, 2021, https://doi.org/10.1016/j.comnet.2017.04.003.

[8] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *Comput. Commun. Rev.*, vol. 44, pp. 87–98, 2014, https://doi.org/10.1145/2602204.2602219.

[9] P. Göransson, C. Black, and T. Culver, *The OpenFlow Specification*. 2017. https://doi.org/10.1016/B978-0-12-804555-8.00005-3.

[10] S. Huang, J. Griffioen, and K. L. Calvert, "Network hypervisors: Enhancing SDN infrastructure," in *Computer Communications*, June 2014, vol. 46, pp. 87–96. https://doi.org/10.1016/j.comcom.2014.02.002.

[11] M. Hamdan *et al.*, "A comprehensive survey of load balancing techniques in software-defined network," *J. Netw. Comput. Appl.*, vol. 174, no. October 2020, p. 102856, 2021, https://doi.org/10.1016/j.jnca.2020.102856.

[12] B. P. Mulla, C. Rama Krishna, and R. K. Tickoo, "Load balancing algorithm for efficient VM allocation in heterogeneous cloud," *Int. J. Comput. Networks Commun.*, vol. 12, no. 1, pp. 83–96, 2020, https://doi.org/10.5121/ijcnc.2020.12106.

[13] Z. Benlalia, K. Abouelmehdi, A. Beni-hssane, and A. Ezzati, "Comparing load balancing algorithms for web application in cloud environment," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 17, no. 2, p. 1104, 2020, https://doi.org/10.11591/ijeecs.v17.i2.pp1104-1108.

[14] T. E. Ali, A. H. Morad, and M. A. Abdala, "Load balance in data center SDN networks," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 5, pp. 3084–3091, 2018, https://doi.org/10.11591/ijece.v8i5.pp3084-3091.

[15] A. A. Alkhatib, A. Alsabbagh, R. Maraqa, and S. Alzubi, "Load balancing techniques in cloud computing: Extensive review," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 6, no. 2, pp. 860–870, 2021, https://doi.org/10.25046/aj060299.

[16] M. R. Belgaum, S. Musa, M. M. Alam, and M. M. Su'Ud, "A systematic review of load balancing techniques in software-defined networking," *IEEE Access*, vol. 8, pp. 98612–98636, 2020, https://doi.org/10.1109/ACCESS.2020.2995849.

[17] S. Kaur, J. Singh, K. Kumar, and N. S. Ghumman, "Round-robin based load balancing in software defined networking," *Proceedings of the 2015 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2015*, pp. 2136–2139, 2015.

[18] F. Rhamdani, N. A. Suwastika, and M. A. Nugroho, "Equal-cost multipath routing in data center network based on software defined network," *Proceedings of the 2018 6th International Conference on Information and Communication Technology (ICoICT)*, 2018, pp. 222–226. https://doi.org/10.1109/ICoICT.2018.8528730.

[19] H. Zhong, Y. Fang, and J. Cui, "LBBSRT: An efficient SDN load balancing scheme based on server response time," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 183–190, 2017, https://doi.org/10.1016/j.future.2016.10.001.

[20] K. Soleimanzadeh, M. Ahmadi, and M. Nassiri, "SD-WLB: An SDN-aided mechanism for web load balancing based on server statistics," *ETRI J.*, vol. 41, no. 2, pp. 197–206, 2019, https://doi.org/10.4218/etrij.2018-0188.

[21] S. Wilson Prakash and P. Deepalakshmi, "DServ-LB: Dynamic server load balancing algorithm," *Int. J. Commun. Syst.*, vol. 32, no. 1, pp. 1–11, 2019, https://doi.org/10.1002/dac.3840.

[22] V. Koryachko, D. Perepelkin, and V. Byshov, "Approach of dynamic load balancing in software defined networks with QoS," *Proceedings of the 2017 6th Mediterranean Conference on Embedded Computing (MECO)*, 2017, pp. 1–5. https://doi.org/10.1109/MECO.2017.7977237.

[23] A. S. AbdelRahman and A. B. El-Sisi, "Dynamic load balancing technique for software defined Wi-Fi networks," *Proceedings of the 2017 12th International Conference on Computer Engineering and Systems (ICCES)*, Cairo, Egypt, 2017, pp. 289-294, https://doi.org/10.1109/ICCES.2017.8275321.

[24] H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: Dynamic load-balanced Routing in OpenFlow-enabled Networks," *Proceedings of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 290–297. https://doi.org/10.1109/AINA.2013.7.

[25] F. S. Fizi and S. Askar, "A novel load balancing algorithm for software defined network based datacenters," *Proceedings of the 2016 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications, CoBCom 2016*, 2016, pp. 1–6. https://doi.org/10.1109/COBCOM.2016.7593506.

[26] Y. Gao and L. Yu, "Energy-aware load balancing in heterogeneous cloud data centers," *Proceedings of the ACM Int. Conf. Proceeding Ser.*, 2017, pp. 80–84, https://doi.org/10.1145/3034950.3035000.

[27] T. Malbasic, P. D. Bojovic, Z. Bojovic, J. Suh, and D. Vujosevic, "Hybrid SDN networks: A multi-parameter server load balancing scheme," *J. Netw. Syst. Manag.*, vol. 30, no. 2, pp. 1–28, 2022, https://doi.org/10.1007/s10922-022-09642-y.

[28] B. B. Rodrigues, A. C. Riekstin, G. C. Januario, V. T. Nascimento, T. C. M. B. Carvalho, and C. Meirosu, "GreenSDN: Bringing energy efficiency to an SDN emulation environment," *Proceedings of the 2015 IFIP/IEEE Int. Symp. Integr. Netw. Manag. IM 2015*, pp. 948–953, 2015, https://doi.org/10.1109/INM.2015.7140416.

[29] Y. H. Chen, T. L. Chin, C. Y. Huang, S. H. Shen, and R. Y. Huang, "Time efficient energy-aware routing in software defined networks," *Proceedings of the 2018 IEEE 7th Int. Conf. Cloud Networking, CloudNet 2018*, pp. 1–7, 2018, https://doi.org/10.1109/CloudNet.2018.8549457.

[30] J. Light, "Green networking: A simulation of energy efficient methods," *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 1489–1497, 2020, https://doi.org/10.1016/j.procs.2020.04.159.

[31] S. Rout, K. S. Sahoo, S. S. Patra, B. Sahoo, and D. Puthal, "Energy efficiency in software defined networking: A survey," *SN Comput. Sci.*, vol. 2, no. 4, pp. 1–15, 2021, https://doi.org/10.1007/s42979-021-00659-9.

[32] Y. Narimani, E. Zeinali, and A. Mirzaei, "QoS-aware resource allocation and fault tolerant operation in hybrid SDN using stochastic network calculus," *Phys. Commun.*, vol. 53, Aug. 2022, https://doi.org/10.1016/j.phycom.2022.101709.

[33] S. S. Patra, R. Govindaraj, S. Chowdhury, M. A. Shah, R. Patro, and S. Rout, "Energy efficient end device aware solution through SDN in edge-cloud platform," *IEEE Access*, vol. 10, no. November, pp. 115192–115204, 2022, https://doi.org/10.1109/ACCESS.2022.3218328.

[34] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," *Proceedings of the 2018 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2018*, 2018, pp. 1–5. https://doi.org/10.1109/ICCTAC.2018.8370397.

[35] S. Wang, K. G. Chavez, S. Kandeepan, and P. Zanna, "The smallest software defined network testbed in the world: Performance and security," *Proceedings of the IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018, pp. 1–2. https://doi.org/10.1109/NOMS.2018.8406116.

[36] A. Chakib-Belgaid, "pyRAPL," 2019. [Online]. Available at: https://pyrapl.readthedocs.io/en/latest/#

Mrs. Vani K.A is Assistant Professor at Dayananda Sagar College of Engineering, Visvesvaraya Technological University (VTU), Bengaluru, Karnataka, India. She received her Bachelor of Engineering and Master of Technology degree in Computer Science and Engineering from VTU, Belagavi, Karnataka, India. She is currently pursuing Ph D from VTU, Belagavi, Karnataka, India. She has about 14 years of experience in teaching. Her areas of interest are computer networks, QoS in SDN, mobile networks, serverless computing and machine learning. She can be contacted at email: vanika-ise@dayanandasagar.edu.

Prof. RAMA MOHAN BABU K.N is currently working as Professor in the Department of Information Science and Engineering at Dayananda Sagar College of Engineering, Bengaluru, India. He obtained his B.Tech in Computer Science and Engineering from Mangalore University, India, M.S from BITS-PILANI India and PhD from Dr.MGR University, India. His areas of interest are computer networks, wireless mobile networks,QoS in SDN and network security. He can be contacted at email: ramamohanbabu-ise@dayanandasagar.edu.