DOI: http://dx.doi.org/10.26483/ijarcs.v17i1.7401

Volume 17, No. 1, January-February 2026

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

OPTIMIZING SERVER-SIDE DYNAMIC LOAD BALANCING IN SDN USING NOVEL ALGORITHMS BASED ON CPU, RAM, AND CONNECTION METRICS

Maghrib Abidalreda Maky Alrammahi &
Mohanad Yahya Al-hamami
Information Technology Research and Development Centre,
University of Kufa, Najaf, Iraq

Ali Mohammed Taher
Faculty of Basic education, Department of Mathematics,
University of Kufa, Najaf, Iraq

Abstract: A software-defined network (SDN) is a network architecture that decouples the data forwarding plane, which forwards data, from the controller plane, which conducts network management and decisions. SDN has exhibited considerable advantages over traditional non-SDNs. However, traffic allocation in SDN impacts efficiency and introduces various challenges. For instance, an unbalanced load distribution in the SDN significantly impacts the performance of the network. This study aims to optimise the utilisation of cluster resources by addressing the load-balancing issue on the server side. It attempts to improve the performance of servers with varying processing capabilities while improving the quality of service in terms of delay, waiting time and service time. Four dynamic load balancing techniques are proposed: least random access memory (LRAM), least control process unit (LCPU), LRAM with LCPU (LCPURAM) and least connection (LC) with LCPURAM (LCLCPURAM). The default dynamic load balancing algorithm based on LC is compared with these strategies. The evaluation employs a Mininet emulator and an Open-flow switch with a POX controller. The evaluation results indicate that the proposed LCLCPURAM algorithm significantly enhanced the delay metric by 15.5%. In addition, it achieved the highest improvement in the waiting time metric, with a 19.2% enhancement. In the meantime, the LC algorithm exhibited the greatest improvement in the service time metric, with a 27% enhancement.

Keywords: Software-defined network (SDN), Load balancing (LB), Least random access memory (LRAM), Least control process unit (LCPU), Least connection (LC), POX, Mininet.

1. INTRODUCTION

The software-defined network (SDN) paradigm separates the control and data planes; network intelligence and control logic are centralized to the network controller [1]. This control plane controller undertakes centralised control and network administration functions that guide the forwarding behaviour of all the dispersed units within the system [2].

In the data plane, switches and network nodes will compare the metadata in flow packets with the rules and instructions offered by the network controller [3]. The data is compared, and a decision is made on forwarding. This basic centralization of SDN can guarantee that the network controller is always aware of the health of the network, and all traffic flows into the controller at least once to decide how to forward traffic [4].

LB, among others, plays a significant role in SDN. It enables the operation of services and routes even when there is congestion on a network or when a route or device fails. When applied to SDN, load balancing can be described as a technology that balances the dynamically varying load uniformly across the networks or pathways [5]. This method will ensure that some servers are busy and some are not over utilized, and thus, when there are sudden increases in traffic, there will not be overloads on the servers. The main aims of load balancing in SDN are to enhance quality of service (QoS) parameters, reduce delays, service times, and wait times, avoid congestion, and enhance resource utilization [6].

There are usually three classes of the main algorithms used to load balance (LB): the static, dynamic, and hybrid methods [7].

i. Static algorithms: These are based on the past experience of network layout and workload. As an example, the Round Robin evenly allocates tasks to servers, and the Min-Min algorithm picks the shortest-timed tasks based on an approximation [8].

ii. Dynamic algorithms: These adapt the load allocation according to real-time measurements of the state of each server or router. These kinds of algorithms are quick to change with the network [9]. Examples are the Least Connection algorithm, which allocates requests to the least busy server, and the Ant Colony Optimization, which tries to emulate the behavior of the ants to determine optimal routes.

iii. Hybrid algorithms: These are based on the capabilities of the two methods of load balancing that are static and dynamic in nature [10]. As an example, machine learning algorithms like decision trees and support vector machines (SVMs) can be combined to identify the best load distribution and forwarding paths.

Several notable contributions of this study are outlined below:

- This study proposes four dynamic algorithms. The first proposed algorithm employs the least random access memory (LRAM) method. The LRAM algorithm identifies the server with the lowest memory usage through calculations.
- The second proposed approach is a least control process unit (LCPU) algorithm that determines the minimum processor utilization among servers.
- A novel technique called LCPURAM merges the LRAM and LCPU algorithms. This integration is achieved using a mathematical equation to calculate the lowest processor and memory consumption across several servers.
- This study introduces an approach that combines three algorithms: least connection (LC) with LCPURAM (LCLCPURAM). This technique utilises a mathematical

equation to calculate the LC and the least processor and memory consumption across servers.

- The performance of the presented algorithms is assessed on the Mininet platform using the POX controller. The experimental findings are compared with a baseline mechanism called LC. Lastly, the optimum approach is selected by calculating the QoS parameters.

The rest of the article is organised as follows. The relevant research is discussed in Section (2). Section (3) explains the standard algorithm. While, section (4) explains the proposed algorithms. Section (5) outlines the network structure, presents the experimental methods of the proposed technique and displays the simulation outcomes. Finally, section (6) presents the conclusion of this work.

2. RELATED WORK

This section provides an overview of prior works on the dynamic load balancing of SDNs. The related contributions are summarized in Table 1.

The authors in [11] propose a technique for load balancing called 'enhance_conn', which they demonstrate to be an improvement over LC scheduling. The algorithm dynamically calculates server load factors, response times and connection numbers. Compared with round_robin and least_conn strategies, the enhance_conn strategy achieves superior overall performance of server clusters in terms of resource efficiency, system throughput and processing capacity.

The authors in [12] suggest using an SDN-based three-tier data centre network to address these challenges by utilising a dynamic load balancing algorithm. This algorithm obtains operational topology and device details, such as IP, MAC address and ports. The second algorithm identifies the shortest paths using Dijkstra's algorithm to compute the total link cost. The third algorithm considers the minimum transmission cost for generating flows, selects the least cost path and installs static flows into each switch in the chosen path. The testing focuses on QoS attributes, including throughput, bandwidth and response time.

The authors in [13] introduce the adaptive least load ratio approach to mitigate the problem of resource counter isolation in OpenFlow switches within SDNs. The method employs dynamic load balancing schemes, such as LC-based and least bandwidth-based. Server performance is assessed using the HTTPerf tool. The results demonstrate faster server response times, quicker connection speeds and higher network data transfer rates.

The authors in [14] evaluate the effectiveness of four load-balancing methods: random, round robin (RR), weighted round robin (WRR) and LC. The evaluation is performed using a Mininet emulator and an Open Flow switch. The primary goal of this research is to determine which of the load-balancing techniques results in the least resource waste. Effectiveness is measured based on workload, load balance degree and time values. This study demonstrates that the LC algorithm can achieve maximum workload and a high level of RT measurement accuracy.

The authors in [15] apply two load balancing algorithms: shortest delay and LC. These algorithms are evaluated for their efficiency using the Mininet and RYU simulators with the Flask web server. Across all connection tests, LC appears to have the lowest CPU utilization and response time, while shortest delay performs the best under high connection testing.

The authors in [16] propose a dynamic method to weight servers in the WRR algorithm using the LC algorithm. This technique adjusts to the real-time data rate of each server to handle changing workloads effectively and balance the load. The LC method resolves tie-breaking and ensures equitable load distribution. In distributed contexts, the proposed technique improves server resource utilization, reduces response times and enhances overall system performance compared with standard static WRR techniques.

The authors in [17] analyse the effectiveness of static (random, RR, WRR and LTWRR) and dynamic (dynamic weight random selection) load balancing methods. They simulate and compare static and dynamic algorithms in terms of delay, throughput, packet delivery ratio and packet loss. The dynamic algorithm also improves QoS results.

Table 1: Summary of Prior Works on Dynamic Load Balancing in SDNs

Ref..	Proposed Algorithms	Comparative Algorithms	Pros	Cons	Results
[11] 2018	Proposed of dynamic load balancing, enhance_conn strategy	Compared with round_robin (RR) and least_conn (LC)	Improved resource efficiency, throughput, and processing capacity	Complexity and overhead on the system.	Resource efficiency is 20% better compared to round-robin and least-conn algorithms.15% rise in throughput. 25% increase in processing capacity
[12] 2021	Dynamic load balancing, topology gathering, Dijkstra's algorithm, flow generation	Tested both before and after implementing the load balancing method without using any standard algorithms.	Addressing network challenges, optimal path selection, improved throughput, bandwidth, and response time	The study lacks security ,also challenging to integrate with current network infrastructures.	The algorithm significantly increased throughput at distribution and core levels by at least 50% and 33%, respectively, resulting in faster response times
[13] 2021	Proposed Adaptive Least Load Ratio Algorithm (ALLR)	Comparing with the least connection (LC) and the least bandwidth	Quicker server response times, enhanced connection speeds, and increased	Energy efficiency, computational overhead, and memory consumption have	Reduced the server response time by 13.37%, connection time by 16.3, and network throughput by 8%. Also, the server CPU

			network data transfer rate	not been computed	being utilized and average queue length decreasing to 2.5% and 14%
[14] 2022	utilized standard algorithms such as Random, RR, WRR, LC methods but did not include the suggested algorithm	Comparing the Random, RR, WRR, and LC algorithms to determine the most effective one	Determine three key parameters: response time, workload, and degree of load balancing	The paper lacks any suggested algorithm and applies standard algorithms directly	For the workload metric, the accuracy of the LC method was highest (LC= 99.87%), followed by the RT average (LC= 97.6%) and the average degree of LB (Random= 98.7%, LC= 96.9%).
[15] 2023	Proposing the integration of two algorithms: Shortest Delay and Least Connection (LC)	This study evaluates the performance of two load-balancing algorithms, namely Shortest Delay and Least Connection.	Lower CPU consumption, faster response times	It needs more algorithms to compare with the proposed algorithm	The Shortest Delay algorithm reduces CPU consumption and response time in connection tests.
[16] 2023	Proposing the integration of two algorithms: dynamic-weight WRR and LC	In comparison to conventional static-weight WRR algorithms	Enhanced load distribution, decreased response durations	It needs more algorithms to compare with the proposed algorithm	Enhanced server resource efficiency, decreased response times, and enhanced overall system performance
[17] 2023	Utilized standard algorithms such as Random, RR, WRR, LTWRR, DWRS methods but did not include the suggested algorithm	Comparing between of static and dynamic load balancing methods (Random, RR, WRR, LTWRR, DWRS)	Improved QoS, better performance, as latency, throughput, packet delivery ratio, and packet loss	The paper lacks any suggested algorithm and applies standard algorithms directly (static and dynamic)	Dynamic algorithms outperform static algorithms. in delay, throughput, packet delivery ratio, and packet loss metrics

Previous investigations have highlighted several drawbacks. Some studies do not propose new algorithms; instead, they only implement standard algorithms directly, as noted in [14, 17]. Other studies referenced in [11-13, 15, 16] suggest merely a single algorithm. In addition, these studies could have calculated crucial parameters such as waiting time and service time, which are essential for evaluating service quality and improving network performance. The present research addresses these shortcomings by proposing four algorithms and calculating important metrics: latency, waiting time and service time.

3. STANDARD ALGORITHM

The Least Connection (LC) algorithm is another dynamic load balancing technique that apportions network traffic to servers according to the number of active connections that each server is currently supporting [18]. The load balancer examines the number of active connections in each server when a new client request comes. Then the request is directed to the server with the fewest active connections. Finally, the number of active connections made on the server reduces as connections close, and it can accept new requests once again. The mathematical calculation is illustrated in Eq. (1).

$$Selected\ Server = \arg\min\ (conni((t)) \quad (1)$$

where

- conni(t) is the least connection of server i at time t.
- arg min returns the index of the server with the least number of connection.

The Fig. 1 illustrates the steps of the least connection algorithm, and in the second step, based on Equation 1, it is used to select the minimum number of connections between the servers.

```
Algorithm Least_Connection_LB

Input:
   R = incoming client request
   S = {S1, S2, .., Sn} // set of available servers
   Conn[i] = number of active connections on server Si

Output:
   Selected server for request R

Begin
   // Step 1: Monitor servers' active connections
   For each server Si in S do
      Get Conn[i]   // current number of active connections

   // Step 2: Select server with minimum active connections based on Eq (1)
   selected_server ← arg min (Conn[i])

   // Step 3: Forward request to the selected server
   Forward(R, selected_server)

   // Step 4: Update connection count
   Conn[selected_server] ← Conn[selected_server] + 1

   // Step 5: When connection closes, decrement counter
   On Connection_Close(selected_server):
      Conn[selected_server] ← Conn[selected_server]  ,  1

End
```

Figure 1: Steps of LC Algorithm

4. PROPOSED ALGORITHMS

In this paper, four algorithms are proposed to operate in a dynamic environment in SDN environments. Based on Figure

1, which shows the steps of the least connection algorithm, all proposed algorithms were modified in steps one and two. The first step is based on the proposed algorithms; the first proposed algorithm uses real-time calculations using a function that calculates the CPU parameter. The second algorithm relies on RAM calculations, the third on CPU and RAM parameters, and the fourth algorithm relies on CPU and RAM parameters and the lowest number of connections. In the second step, the proposed algorithms are calculated using the mathematical equations mentioned below in each algorithm. The Fig. 2 shows the general proposed flowchart for the proposed algorithms.

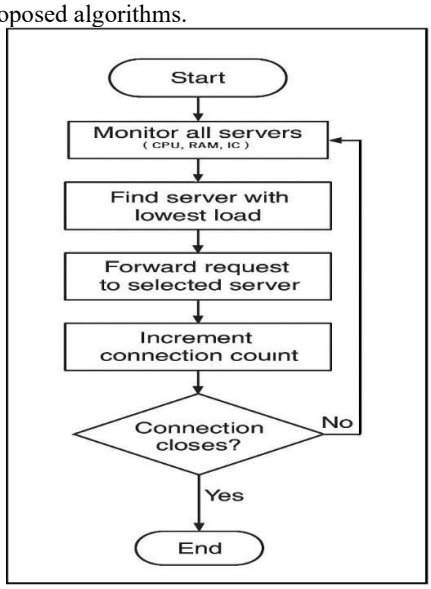

Figure 2: General Flowchart of Proposed Load Balancing Algorithms

Algorithm 1, which is called LCPU, is developed to calculate CPU usage within servers during real-time execution, which selects the server with the lowest LCPU. The mathematical calculation is illustrated in Eq. (2).

$$Selected\ Server = arg\ min(CPUi(t)) \quad (2)$$

where

- CPUi(t) is the CPU usage of server i at time t.
- arg min returns the index of the server with the minimum CPU usage.

Algorithm 2, which is called LRAM, is designed to calculate RAM usage within servers during real-time execution in each packet reception, which selects the server with the least RAM usage. The mathematical calculation is explained in Eq. (3).

$$Selected\ Server = \arg\min\big(RAMi(t)\big) \quad (3)$$

where

- RAMi(t) is the RAM usage of server i at time t.
- arg min returns the index of the server with the minimum RAM usage.

Algorithm 3, based on the first and the second algorithms, is merged and combined and known as LCPURAM. It determines the CPU and RAM usage of servers during real-time operations to identify the server with the lowest overall load. The calculation is mathematical and described in Eq. (4).

$$Selected\ Server = argmin\big(\alpha \cdot CPUi(t) + \beta \cdot RAMi(t)\big) \quad (4)$$

where

- α and β are weighting factors to balance CPU and RAM usage.
- CPUi(t) and RAMi(t) represent the CPU and RAM usage of server i at time t.

Algorithm 4 combines three parameters to determine server loads: CPU usage, RAM usage, and LC latency, and it is known as LCCPURAM. It chooses the least loaded server in general. Mathematically this is calculated as in Eq. (5).

$$Selected\ Servr = \arg\min\big(\alpha \cdot CPUi(t) + \beta \cdot RAMi(t) + \gamma \cdot Leasti(t)\big) \quad (5)$$

where

- α, β and γ are weighting factors for CPU usage, RAM usage and LC.
- Leasti is the connection least of server i at time t.

5. EVALUATION AND RESULTS

This section describes the proposed architecture and the performance of the four suggested algorithms as compared to another standard algorithm known as LC, within an SDN environment. The main objective is to measure the functionality of all algorithms to enhance network performance, load distribution, and optimal utilization of resources on the servers. The evaluation is based on metrics such as LC, CPU and RAM, using Mininet simulations and the POX controller. The best algorithm is demonstrated to optimise resource allocation and improve network performance by calculating key parameters such as service time, waiting time and latency.

A) Proposed Topology

This paper suggests utilising a network with a single topology type, specifically a 'Single Switch' comprising three hosts and three servers. The proposed network is illustrated in Fig. 3. Each differently sized packet (1, 10, 25, 50, and 75 MB) is transmitted in a homogeneous environment, which ranges between 8,000 and 40,000 packets.

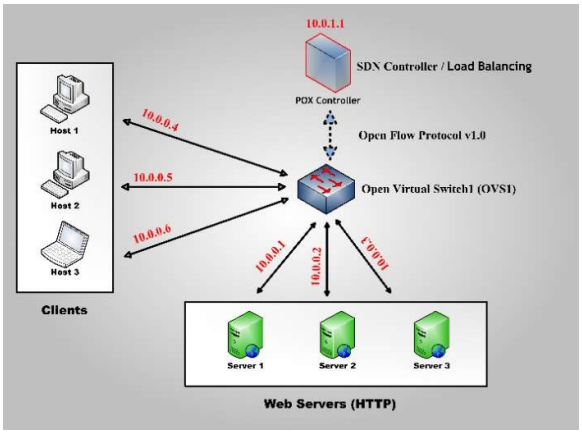

Figure 3: Proposed Topology

The specifications of the computers and network devices used in the simulation are detailed in Table 2.

Table 2: Specifications of Computers and Network Devices Used in the Simulation

Computer		Emulator	
Parameters	**Values**	**Emulator**	**Mininet**
OS	Linux Ubuntu 20.04.4 LTS	Controller architecture	Single controller

CPU	Intel Core i7-3625QM 2.2GHz *8	Type of Controller	POX
Memory	8 GB	Number of Switch & type	1, OVS
HDD	*512GB SSD*	Number of servers & type	3, Web Servers
Graphics	AMD/Intel4000	Number of Clients	3
OS Type	64-bit	Southbound interface	OpenFlow 1.0

B) Result and Experiments

To calculate the QoS and identify the optimal algorithm among LRAM, LCPU, LCPURAM, LCLCPURAM and LC for improving network performance and load distribution, three parameters are adopted: delay, waiting time and service time.

A) Delay

In an SDN, the delay parameter represents the total time taken for a data packet to travel from the source to the destination. This parameter consists of propagation delay, transmission delay, processing delay and queueing delay. Table 3 presents the results of calculation delay for five algorithms: LRAM, LCPU, LCPURAM, LC and LCLCPURAM. The analysis of the improvement percentages revealed that the LCLCPURAM algorithm showed the greatest improvement at 15.5%, followed by LC (12.6%), LCPU (11.5%), LCPURAM (8.5%) and LRAM (1.5%). The comparative analysis of the results among the algorithms is shown in Fig. 4.

Table 3. Statistics of Delay Parameter for Algorithms

No. of Request	LRAM	LCPU	LCPURAM	LC	LCLCPURAM
8000	0.178	0.0807	0.0817	0.0817	0.0826
40,000	0.181	0.0838	0.0833	0.0826	0.0812
Improving	1.5%	11.5%	8.5%	12.6%	15.5%

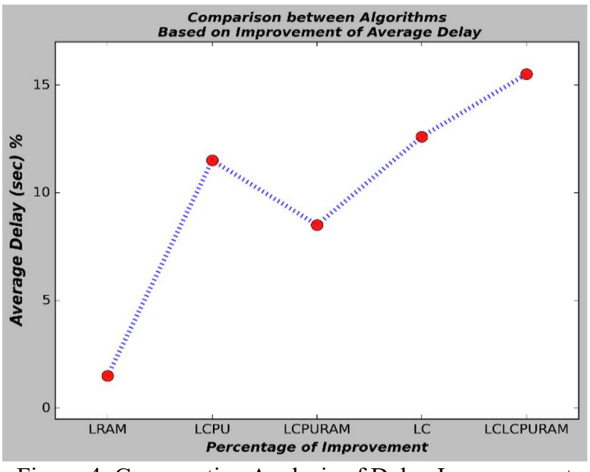

Figure 4: Comparative Analysis of Delay Improvement Percentages Across Algorithms

A) Waiting Time

Defined as the time a packet of data or request takes to wait in a queue before it is handled by network components (e.g., switches or controllers) in an SDN. The delay is a major issue in an SDN environment that affects performance, efficiency, and QoS. Table 4 shows the outcomes of computing the waiting time of the five algorithms, i.e., LRAM, LCPU, LCPURAM, LC, and LCLCPURAM. A comparative analysis of the percentiles of improvement revealed that the LCLCPURAM algorithm improved by 19.2 percent, followed by LC (10.6%), LCPU (7.4%), LCPURAM (7.3%), and LRAM (0.5%). The Fig. 5 represents the comparison of the results via the algorithms.

Table 4. Statistics of Waiting Time Parameter for Algorithms

No. of Request	LRAM	LCPU	LCPURAM	LC	LCLCPURAM
8000	0.158	0.0730	0.0738	0.0738	0.0741
40,000	0.160	0.0755	0.0747	0.0742	0.0726
Improving	0.5%	7.4%	7.3%	10.6%	19.2%

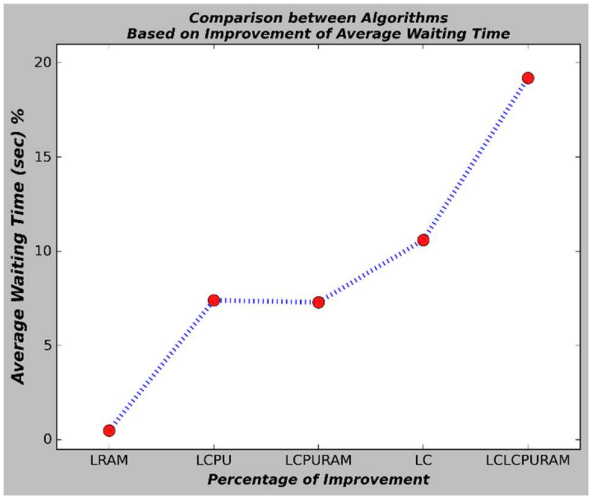

Figure 5: Comparative Analysis of Waiting Time Improvement Percentages Across Algorithms

A) Service Time

The term "service time" refers to the duration required by a network device or controller to perform functions for handling packets. Table 5 provides the results of the algorithmic calculation of service time. The percentile improvement analysis revealed that the LC algorithm recorded the highest improvement of 27, followed by LCPURAM (26%), LCLCPURAM (24.5%), LCPU (1.7%), and LRAM (0.2%). The Fig. 6 shows the relative comparison of the results of the algorithms.

Table 5. Statistics of Service Time Parameter for Algorithms.

No. of Request	LRAM	LCPU	LCPURAM	LC	LCLCPURAM
8000	0.020	0.0164	0.0079	0.0079	0.0084
40,000	0.021	0.0083	0.0085	0.0083	0.0085
Improving	0.2%	1.7%	26%	27%	24.5%

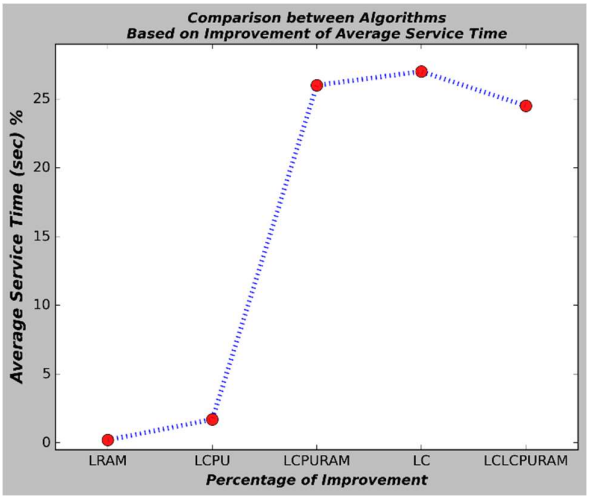

Figure 6: Comparative Analysis of Service Time Improvement Percentages Across Algorithms

6. Conclusion

This paper highlights the significance of solving load balancing issues in SDNs to realize the maximum use of resources and improved network performance. Four dynamic load balancing algorithms are proposed and analyzed in the research; they are LRAM, LCPU, LCLCPURAM, and LCPURAM. It gives insight into performance measures that are critical, like delay, waiting time, and service time. The evaluation outcomes show that LCLCPURAM has the best improvement, having had a decrease of 15.5 in delay and 19.2 in the waiting time. In such a way, it is a great option in situations where QoS is valuable. Additionally, the LC algorithm demonstrated the greatest improvement in service time among all algorithms, achieving an enhancement of up to 27%. These results highlight the promise of the dynamic allocation of network traffic to dynamically fit the SDN environment with dynamic load-balancing methods.

Future studies can examine how to incorporate the application of advanced methods like machine learning and real-time analytics into load balancing. This solution is scalable to larger and more highly structured network systems.

Conflicts of interest

We, Maghrib Abidalreda Maky Alrammahi, Mohanad Yahya Al-hamami, and Ali Mohammed Taher, state that the content of this article entitled 'Optimizing Server-Side Dynamic Load Balancing in SDNs Using Novel Algorithms Based on CPU, RAM, and Connection Metrics' does not contain any conflicts of interest.

Author contributions

The first author performed the primary study work for the given paper, including the development of the methodology, preparation of files, their implementation, analysis, and discussion of the results, and preparation and writing of the manuscript. The second author gave supervision, technical advice, a critical review of the work, preparation of the related work section, analysis, and abstract. The third author helped with supervision, technical advice, and critical evaluation of the work.

Acknowledgments

The authors gratefully acknowledge the support provided by the Information Technology Research and Development Centre (ITRDC), University of Kufa, for this work.

References

[1] "Enhancing an Intelligent Model for Enhancing Software-Defined Networking (SDN) Achievement Using Fog Computing and Reinforcement Learning for Operational Performance and Dynamic Resource Management," *International Journal of Intelligent Engineering and Systems,* vol. 18, no. 5, pp. 347-362, 2025/06/30 2025.

[2] T. Darwish, T. A. Alhaj, and F. A. Elhaj, "Controller placement in software defined emerging networks: a review and future directions," *Telecommunication Systems,* vol. 88, no. 1, 2025/01/18 2025.

[3] "Dual-Agent Reinforcement Learning for Adaptive Bandwidth Allocation and Congestion Control in SDN," *International Journal of Intelligent Engineering and Systems,* vol. 18, no. 7, pp. 338-361, 2025/08/31 2025.

[4] A. M. Abdulghani, A. Abdullah, A. R. Rahiman, N. A. W. A. Hamid, B. O. Akram, and H. Raissouli, "Navigating the Complexities of Controller Placement in SD-WANs: A Multi-Objective Perspective on Current Trends and Future Challenges," *Computer Systems Science and Engineering,* vol. 49, no. 1, pp. 123-157, 2025.

[5] L. Shirani, H. Movahednejad, M. Sharifi, and M. Mahmoudi, "Improving load balancing in distributed software-defined networks with a bidirectional long short-term memory-based controller design," *The Journal of Supercomputing,* vol. 81, no. 10, 2025/07/10 2025.

[6] M. A. Z. Soltani, S. A. Hosseini Seno, and A. Mohajerzadeh, "Optimizing SDN resource allocation using fuzzy logic and VM mapping technique," *Computing,* vol. 107, no. 1, 2024/11/27 2024.

[7] M. A. Maky Alrammahi and W. S. Bhaya, "A State-of-the-Art Survey and Taxonomy of Classification, Algorithms, and Techniques for Load Balancing in SDN," *Journal of Al-Qadisiyah for Computer Science and Mathematics,* vol. 15, no. 3, 2023/09/30 2023.

[8] R. Farahi, "A comprehensive overview of load balancing methods in software-defined networks," *Discover Internet of Things,* vol. 5, no. 1, 2025/01/20 2025.

[9] "A Novel Approach to DDoS Detection in Multi-Controller SDNs: Adaptive Learning Models and Real-Time Feedback for Enhanced IoT Security," *International Journal of Intelligent Engineering and Systems,* vol. 17, no. 5, pp. 570-592, 2024/10/31 2024.

[10] D. Yu and W. Zheng, "A hybrid evolutionary algorithm to improve task scheduling and load balancing in fog computing," *Cluster Computing,* vol. 28, no. 1, 2024/11/20 2024.

[11] L. Zhu, J. Cui, and G. Xiong, "Improved dynamic load balancing algorithm based on Least-Connection Scheduling," in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2018, pp. 1858-1862: IEEE.

[12] G. B. Haile and J. Zhang, "Dynamic Load Balancing Algorithm in SDN-based Data Center Networks," *International Journal of Engineering Research & Technology (IJERT),* pp. 2278-0181.

[13] M. N. Jasim, "A proposed adaptive least load ratio algorithm to improve resources management in software defined network OpenFlow environment," *Karbala International Journal of Modern Science,* vol. 7, no. 1, p. 6, 2021.

[14] M. A. M. Alrammahi and W. S. Bhaya, "Performance Analysis for Load Balancing Algorithms using POX Controller in SDN," in *2022 International Conference on Data Science and Intelligent Computing (ICDSIC)*, 2022, pp. 175-180: IEEE.

[15] H. Nurwarsito and H. K. P. Widagdo, "Comparative Analysis of Load Balancing with Shortest Delay and Least Connection

Methods on Software Defined Network," in *Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology*, 2023, pp. 589-598.

[16] M. S. Almhanna, T. A. Murshedi, F. S. Al-Turaihi, R. M. Almuttairi, and R. Wankar, "Dynamic Weight Assignment with Least Connection Approach for Enhanced Load Balancing in Distributed Systems," 2023.

[17] M. Shona and R. Sharma, "Implementation and Comparative Analysis of Static and Dynamic Load Balancing Algorithms in SDN," in *2023 International Conference for Advancement in Technology (ICONAT)*, 2023, pp. 1-7: IEEE.

[18] M. D. Tache, O. Păscuțoiu, and E. Borcoci, "Optimization Algorithms in SDN: Routing, Load Balancing, and Delay Optimization," *Applied Sciences,* vol. 14, no. 14, p. 5967, 2024/07/09 2024.