Hindawi
Mobile Information Systems
Volume 2023, Article ID 3896267, 10 pages
https://doi.org/10.1155/2023/3896267

Research Article

Optimizing QoS Metrics for Software-Defined Networking in Federated Learning

Mahdi Fallah, Parya Mohammadi, Mohammadreza NasiriFard, and Pedram Salehpour

Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

Correspondence should be addressed to Pedram Salehpour; psalehpoor@tabrizu.ac.ir

Received 29 April 2023; Revised 18 September 2023; Accepted 29 September 2023; Published 9 October 2023

Academic Editor: Nicola Bicocchi

In the modern and complex realm of networking, the pursuit of ideal QoS metrics is a fundamental objective aimed at maximizing network efficiency and user experiences. Nonetheless, the accomplishment of this task is hindered by the diversity of networks, the unpredictability of network conditions, and the rapid growth of multimedia traffic. This manuscript presents an innovative method for enhancing the QoS in SDN by combining the load-balancing capabilities of FL and genetic algorithms. The proposed solution aims to improve the dispersed aggregation of multimedia traffic by prioritizing data privacy and ensuring secure network load distribution. By using federated learning, multiple clients can collectively participate in the training process of a global model without compromising the privacy of their sensitive information. This method safeguards user privacy while facilitating the aggregation of distributed multimedia traffic. In addition, genetic algorithms are used to optimize network load balancing, thereby ensuring the efficient use of network resources and mitigating the risk of individual node overload. As a result of extensive testing, this research has demonstrated significant improvements in QoS measurements compared to traditional methods. Our proposed technique outperforms existing techniques such as RR, weighted RR, server load, LBBSRT, and dynamic server approaches in terms of CPU and memory utilization, as well as server requests across three testing servers. This novel methodology has applications in multiple industries, including telecommunications, multimedia streaming, and cloud computing. The proposed method presents an innovative strategy for addressing the optimization of QoS metrics in SDN environments, while preserving data privacy and optimizing network resource usage.

1. Introduction

In the dynamic and ever-changing realm of the Internet, where a vast number of individuals engage with diverse online platforms encompassing education, healthcare, research, and e-commerce, the imperative of guaranteeing a smooth and uninterrupted user experience has assumed utmost significance. Nevertheless, the highly competitive environment of the digital realm has presented other obstacles, including issues such as server overload and network congestion. In order to mitigate these limitations and optimize network performance, load-balancing methodologies have been implemented. These methodologies aim to effectively distribute communication workloads and client requests across various resources such as servers and routers [1]. The objective of these enhancements is to achieve the highest possible data processing rate, decrease the time it takes to receive a response, and optimize the utilization of the CPU and memory resources.

Traditional networking solutions, which involve the utilization of protocols and firewalls, have historically served as the primary means to manage and enhance networks. Although they are essential, the intricate execution of these systems has presented considerable difficulties [2]. The advent of SDN has brought about a significant transformation in the field of network design and operation. SDN has attracted significant attention within the realm of business and data center networks owing to its capacity to provide swift innovation and flexible administration [3]. The fundamental principle of SDN revolves around the segregation of the control and data planes. In SDN, the control plane is centralized, autonomous, and programmable, which deviates from the conventional network switches that integrate both the data and control planes. The data plane in SDN is located within the network switch, while interaction with the controller is facilitated by the "OpenFlow" protocol [4]. This protocol enables the controller to exercise control and supervision over network traffic by providing instructions to the data plane for packet forwarding. As a result, it offers exceptional flexibility and efficiency in the management of traffic.

Software-defined networking (SDN) introduces a novel phase in network administration through the consolidation of network traffic management into a single control panel. This eliminates the necessity of navigating across several switches, resulting in significant time and effort savings [1]. In addition, SDN is a financially advantageous option for network management by allowing the integration of essential functionalities as applications on the controller. This eliminates the need for costly dedicated network devices with specific attributes. The use of the SDN technique facilitates swift adjustment to dynamic network circumstances, enhances network scalability, and optimizes network performance, rendering it a favored option in corporate and data center networks. Significantly, SDN facilitates the automated configuration of network devices in real-time, enhances network security through the identification and mitigation of threats, and offers exceptional flexibility and control over network traffic.

Within the current dynamic environment, scholars are actively engaged in the development of novel algorithms and the optimization of preexisting ones, using innovative fitness functions. The primary objective of these endeavors is to augment the QoS metrics in software-defined networks (SDNs). The employed methodologies involve the utilization of established algorithms, such as genetic algorithms, for the purpose of ascertaining network paths [5]. In addition, novel switch migration algorithms are introduced, which assess the loads on controllers and perform load-balancing operations, with particular emphasis on response time [5].

This work makes three primary contributions which are as follows:

(1) This study presents a novel approach to enhance QoS in SDN by utilizing FL and genetic algorithms to enable efficient load balancing.

(2) The proposed methodology facilitates the decentralized consolidation of multimedia data, guaranteeing the confidentiality of information and the equitable distribution of network load.

(3) Through the utilization of the SDN control plane as a training phase at the local level (training plane) and the application plane for consolidation (aggregation plane), our approach attains exceptional network efficiency, decreased congestion, and improved resource utilization. These contributions are in line with the emerging discipline of distributed machine learning.

The remainder of this paper is organized as follows. Section 2 reviews the related research on the algorithms used in load-balancing and software-defined networking. Section 3 describes our novel technique to use federated learning to improve a genetic algorithm to enhance QoS metrics in SDN, and Section 4 covers the experimental results and evaluation. Finally, Section 5 concludes the whole paper.

2. Related Work

To achieve load-balancing among the distributed control plane, Zhang et al. [6] introduced OCBL online framework. They begin by considering the load-balancing issue as minimizing the typical controller response time. Then, they break it down into a series of switch migrations, each migration modeled for maximizing the decrease in average response time in accordance with the real-time request distribution. They then created an OCLB method based on the termination circumstances and determined the optimality of switch migration. One of the drawbacks of the OCLB method based on termination circumstances is that it assumes that all tasks in the system are known in advance and that their arrival and departure times are deterministic. This assumption may not hold in dynamic and unpredictable environments, where tasks may arrive or depart randomly or even fail to complete. In such cases, the OCLB method may not be able to balance the load optimally, leading to suboptimal resource utilization and potentially degraded system performance.

Khaliq et al. in [7] conducted tests utilizing both a proactive and a dynamic approach to load balancing in a multipath architecture. They also checked the effects of expanding the TCP window size on performance. One of the primary drawbacks is that increasing the window size can increase network congestion, particularly in high-speed networks, where data can arrive faster than the receiver can process. As a result, the TCP congestion control mechanism may reduce the transmission rate, leading to decreased throughput and increased delay. Chen et al. in [8] explored how SDN switches may be used in M2M networks for load balancing. By utilizing the SDN advantages of rapid traffic identification and dynamic traffic rerouting, a TALB scheme is developed to meet the various QoS needs of the M2M network. However, the TALB scheme's adaptability is limited to threshold value adjustments, which may not be sufficient to meet various QoS needs. For instance, the scheme may not be able to address issues related to latency or jitter in the network. Arahunashi et al. in [4] used Python programming language to implement the round-robin and bandwidth-based load-balancing algorithms using the Mininet emulator. The QoS results of the aforementioned methods are then contrasted, including throughput and response time. While round-robin and bandwidth-based load-balancing algorithms are commonly used, they may not consistently and efficiently distribute network traffic, resulting in an underutilization of network resources. This is especially true in cases where there are significant variations in the traffic loads among servers. This can occasionally lead to some servers being overloaded while others sit idle.

Cui et al. in [5] looked at the issue of several overloaded controllers. To achieve a trade-off between load balancing impact and migration cost, they chose the extremum point on the change curve of response time vs. controller's load as the optimal response time threshold. Finally, they presented the design of SMCLBRT, a cutting-edge switch migration load-balancing strategy based on real-time response time. However, SMCLBRT requires additional network overhead to monitor real-time response time and migrate traffic, which can increase network latency and reduce overall performance.

Ongaro et al. in [9] presented a linear integer programming methodology for tackling the QoS and quality of experience (QoE) implementation difficulty in SDN in terms of network latency and congestion while also considering the network requirements and constraints of real-time applications. Linear integer programming is a rigid optimization technique that may not be able to accommodate new QoS/QoE requirements or changes in network conditions easily. As a result, it may not be suitable for use in rapidly evolving network environments.

To reduce the controller reaction time, Wang et al. in [10] conceptualized the load-balancing problem as a distributed service system for the control plane. As a result of their unacceptably high computational complexity and reliance on the original request distribution, the techniques used to address these global optimization issues can only be used sparingly and offline.

Egilmez et al. in [11] examined the traffic's routing algorithm in two distinct situations. The first scenario refers to the moment when the primary layer of traffic for scalable video encoding is identified as traffic with high QoS and no lost packets. The second instance is when QoS traffic is identified as incremental layers. The second scenario, on the other hand, takes packet loss into account. Through LEMON's network optimization module, the situation was put into practice and analyzed. The first scenario is 6.5% more efficient than the first scenario and 14% more efficient than the best-effort routing. However, LEMON's network optimization module may not be compatible with all types of network data structures or network optimization problems. This means that it may not be suitable for some specific network optimization requirements.

Using the similarity-to-ideal-optimized-priority-ordering (TOPSIS) algorithm in the context of controller decision-making, Shirmarz and Ghaffari [12] have developed a novel method for assessing network performance. This novel approach presents the idea of flow routing to evaluate network performance, with a special emphasis on key indicators including latency, jitter, packet loss ratio (PLR), and bandwidth usage. The approach is implemented within the floodlight controller framework, and then it is compared with other route allocation methods including the shortest path (SP), the greedy algorithm, and the weighted equal cost multipath (ECMP), among others. This study sheds light on the interesting possibility of trade-off analysis difficulties. The TOPSIS algorithm treats all performance measures as equally important, but in the real world of software-defined networking (SDN), there are often competing priorities and trade-offs to be made. To effectively handle and incorporate these trade-offs into decision-making algorithms, further methodologies or extensions beyond the fundamental TOPSIS methodology may be required. This research highlights the changing environment of SDN performance evaluation, highlighting the necessity for all-encompassing solutions that take into consideration the many facets of network optimization.

Similarly, Shirmarz and Ghaffari [13] found that, inside the SDN paradigm, a complete algorithm emerges as a potent tool for optimizing the overall network performance. The utilization, latency, jitter, packet loss ratio, blocking likelihood, and link cost are just some of the performance measures considered by this algorithm. The optimization of flow routing inside the SDN architecture is at the heart of this study. Here, we want to have a look at the wide range of KPIs-like blocking probability, latency, jitter, packet loss ratio, and cost while also trying to get the most out of our network. A unique flow routing algorithm is proposed in this research, making use of nondominated ranking and crowd distance sorting as its foundational building blocks. This innovative method is developed to improve the effectiveness of SDN architectures by minimizing blocking likelihood, latency, jitter, packet loss ratio, and cost while increasing resource utilization. However, it is important to note that there are several caveats to this study. The simulations are the backbone of the algorithm's evaluation, and its performance in the real world may vary. The report also acknowledges the difficulties associated with realizing dynamic routing algorithms in operational networks. In addition, it emphasizes the need for more controller computational power, suggesting that the proposed solution may only work with certain SDN designs. Therefore, it may be necessary to further improve and adapt in order to properly cater to the varied requirements and constraints of SDN settings.

The problem of biased policies has recently become an important topic in the field of network traffic management. Shirmarz and Ghaffari [14] offered a future-proof solution for SDN that makes use of automated deep encryption and the ensemble method. Accurately classifying network traffic is a crucial difficulty in SDN setups. The authors presented a novel flow categorization model based on a comprehensive analysis of network needs to overcome this obstacle. To automatically extract crucial network flow features, this method makes use of deep automated coding. In addition, three well-known classifiers, softmax, support vector machine, and random forest, are used to improve flow classification precision. It is important to keep in mind that while automated deep encodings and ensemble techniques perform admirably in static or moderately dynamic situations, the time needed for training and adaptation may affect their performance in highly dynamic conditions. This study emphasizes the continual quest for more efficient and versatile solutions to counteract discriminatory behaviors in network traffic management within the ever-changing context of SDN.

Manzoor et al. [15] explored the nuances of a subset of high-density Wi-Fi networks that are typically used in outdoor settings such as stadiums. These networks are distinguished by the use of specialized software networks for network administration, which brings its own set of difficulties. Load balancing is the primary issue here because of how important it is to the overall functioning of the network. In order to prevent congestion and maintain QoS standards, it is crucial that such networks employ efficient load balancing to distribute traffic fairly across access points. Using current controllers in the field of SDN is the key method provided in this study to address this issue. To effectively run high-density Wi-Fi networks in public spaces, this technique offers dynamic resource allocation and traffic flow management. This research is important because it elucidates the significance of SDN controllers in optimizing network performance and resource consumption in this specific network scenario.

A groundbreaking architectural framework by Shirmarz and Ghaffari [16] heralds in a new era of autonomy for optimizing the operational efficiency of autonomous software-defined networks. Autonomous SDN must meet performance limitations such as latency, scalability, efficient resource use, and real-time response to changing network conditions. The technique relies on performance drivers across network topologies. This complex strategy includes network expansion and capacity control. Increased capabilities of these core parts will improve autonomous network performance. To enable autonomous decision-making and self-configuration, autonomic SDN architecture may require more computing resources, such as processing power and memory. Considering how this added burden may affect network speed and scalability, especially in large deployments, Shirmarz and Ghaffari [17] also extensively examined the existing methods and technology to address performance issues. Quality of service assurance, load-balancing algorithms, traffic engineering methodologies, and network monitoring tools are covered in this extensive study. This article explores SDN-based data center performance difficulties, using examples and statistics to demonstrate their importance. This research shows how dynamic the SDN area is and how it seeks innovative solutions to its many performance issues.

An important model for assessing and estimating wireless traffic load in the context of next-generation wireless networks is described in the in-depth study by Manzoor et al. [18]. To better understand the dynamics of wireless traffic loads, especially in contexts where next-generation wireless networks are prominent, this model was developed. Many different kinds of complexity contribute to the central difficulties in the field of wireless traffic load modeling. These include the dynamic nature of wireless networks and how it is always changing, the complex web of user activities, and the revolutionary effects of new technology such as 5G and its successors. To overcome such tremendous obstacles, Manzoor et al. presented a novel conceptual framework for addressing load imbalances at various levels of a network. This novel method converges on a four-tier design that integrates SDNs, WLANs, and edge computing without any noticeable performance degradation. It is important to note, however, that the introduction of next-generation wireless networks has greatly amplified the complexity of network topologies. The integration of multicarrier systems, the use of multiple radio access technologies, and the creation of heterogeneous networks all add additional layers of complexity. Therefore, estimating traffic load in these sophisticated networks requires an in-depth understanding of the complex interconnections and interactions between a wide variety of network elements. This set of work is an important addition to the developing field of next-generation wireless networks, illuminating the complex problems and novel approaches that characterize this fast-paced industry.

3. Methodology

The proposed methodology employs a three-layered architecture, as illustrated in Figure 1. The first layer, referred to as the data plane, consists of data-gathering objects, such as sensors, CCTV cameras, access points, and network switches. In the second layer, referred to as the training plane, local models are trained on the data collected from the first layer using local model training servers. A genetic algorithm is used to optimize the training process. The communication between the first and second layers is facilitated by the southbound API and OpenFlow protocol [19]. The topmost layer is the aggregation plane, which is composed of a central aggregator server. The aggregation plane collects all the local models from the training plane using the northbound API, such as the REST API, and performs aggregation using the MOE aggregation algorithm on the transformed local models received from the training plane [20].

The present study employed a heterogeneous dataset consisting of pictures obtained from CCTV camera systems located in different geographic areas. The dataset was meticulously curated to accurately reflect real-life situations, incorporating a range of environmental circumstances, variances in lighting conditions, and diverse camera angles. The collection of photographs exhibited a diverse array of resolutions and formats, effectively emulating the real-world obstacles encountered in the handling of multimedia traffic inside a framework of SDN.

Each of the chosen locations preserved its collection of images on a localized server, which was seamlessly included in the network architecture via networking components such as routers and switches [21]. The institution in question replicated a standard data plane setup within a SDN simulation. In this configuration, multimedia content is dispersed across different geographic areas, and network devices are responsible for facilitating the transfer of data.

The simulation procedure entailed the establishment of a virtual SDN environment, whereby we mimicked network traffic management situations by utilizing the acquired pictures from the CCTV camera system. In this section, we

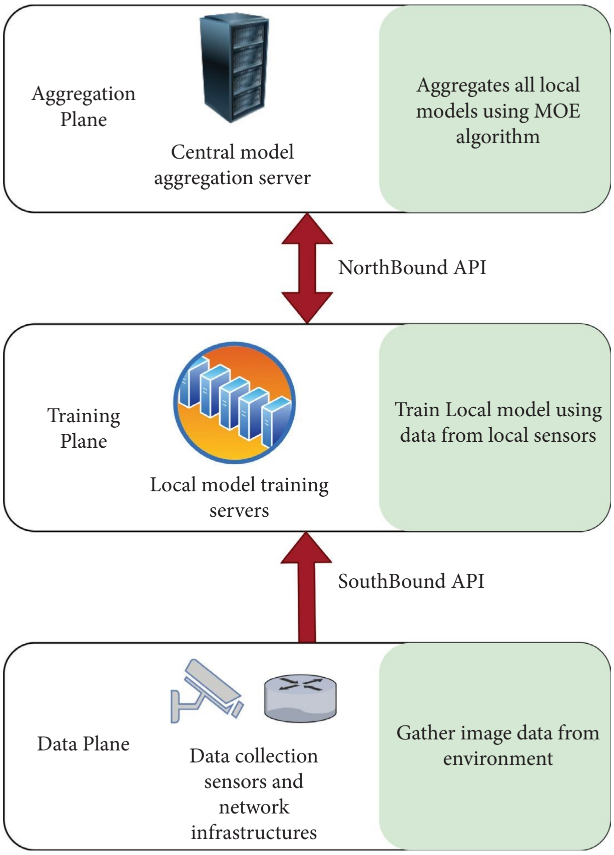
Aggregation Plane
Central model aggregation server
Aggregates all local models using MOE algorithm
NorthBound API
Training Plane
Local model training servers
Train Local model using data from local sensors
SouthBound API
Data Plane
Data collection sensors and network infrastructures
Gather image data from environment

Figure 1: Architecture of SDN using federated learning.

will present a succinct summary of the procedures entailed in our simulation.

(1) The network architecture was meticulously constructed to accurately represent the diverse composition of contemporary networks. The implementation involved the utilization of SDN controllers, switches, and routers in a manner that accurately simulated network architectures found in real-world scenarios.

(2) The chosen CCTV camera system pictures were included in the simulation environment, replicating the live retrieval of multimedia content from different areas. The multimedia traffic flows were regarded as pictures within our SDN architecture.

(3) The distribution of multimedia traffic throughout the virtual network architecture was coordinated by us. The SDN controller assumed a crucial role in dynamically directing network traffic by considering QoS demands and load-balancing strategies.

The SDN trainer is equipped with a comprehensive network architecture that enables it to update topology information, identify routes between different source and destination nodes, and monitor their traffic load status. The proposed architecture comprises two clusters of cameras connected to the SDN trainer server through network routers, as shown in Figure 2. The local model aggregation server serves as the SDN trainer and is connected to a load balancer that distributes traffic load to each router. The load balancer employs a genetic algorithm to optimize the selection of the best path for faster communication between the cameras and the SDN trainer.

The training plane in each cluster serves as the system's core, which is responsible for optimizing QoS metrics by generating machine learning models and improving the genetic algorithm. These models are used to determine the optimal route for each packet in the routers or optimize data to enable faster communication. The primary objective of the local model trainer server is to maximize throughput while minimizing response time. In doing so, it aims to optimize CPU and memory resource utilization. By generating local machine learning models, the trainer server can effectively capture the network's behavior and make accurate predictions, facilitating faster communication and reduced network congestion. On the other hand, the genetic algorithm provides an efficient mechanism for exploring the solution space to identify the optimal path that meets the desired QoS metrics. Overall, the combination of machine learning and genetic algorithm enables the proposed system to adapt dynamically to changing network conditions, providing an efficient and effective mechanism for optimizing QoS metrics in software-defined networking.

Due to the limited amount of data in each cluster, the model's accuracy is compromised. To address this problem, federated learning is utilized as it is a cutting-edge approach to machine learning. The local model trainer servers communicate with a central cloud server to share their local models, which are then aggregated using the MOE algorithm. After generating the global model, it is broadcasted to each cluster's trainer server [22]. This technique is highly secure as it never sends raw data outside the network, ensuring the preservation of sensitive information. By using this approach, the lack of training data in the clusters is mitigated, and each trainer plan receives a well-trained global model that incorporates data from all clusters, thereby improving the performance of the genetic algorithm.

The number of participating clients, the sophistication of the machine learning models employed, and the communication overhead may all affect the time and space complexity of federated learning.

3.1. Time Complexity

(1) Training at each client: the time complexity associated with training a machine learning model at each client in federated learning is largely determined by the model's complexity, which is often measured by the number of parameters it possesses, as well as the quantity of local data accessible at each client. The computational complexity of models can vary, with a lower bound of $O(1)$ for simpler models and an upper bound of $O(N)$ for more intricate models, where N represents the quantity of local data samples.

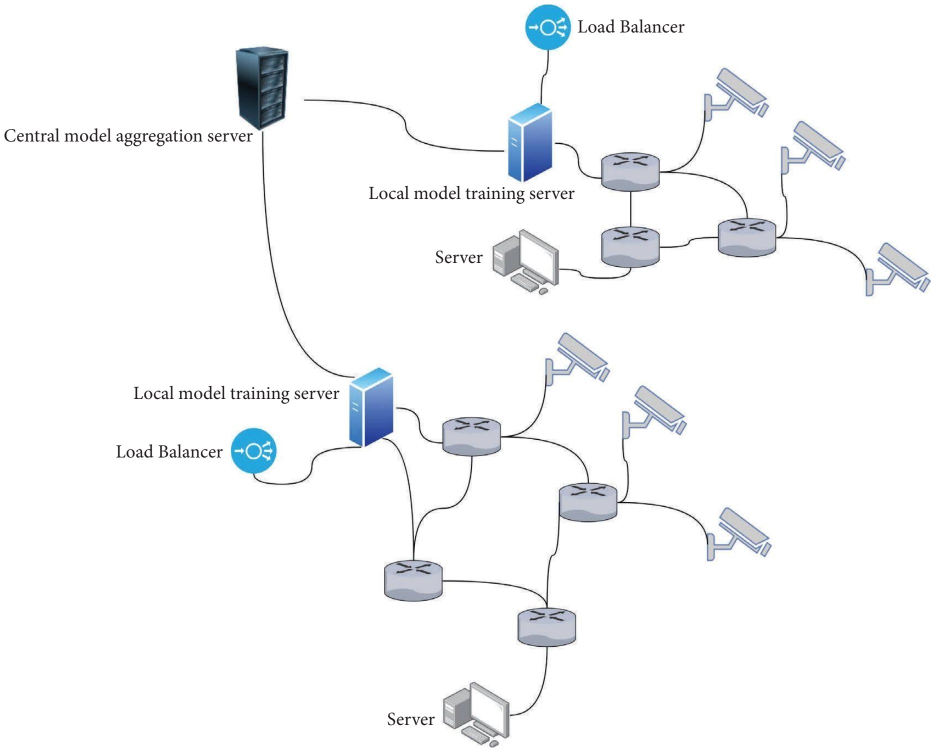
Load Balancer
Central model aggregation server
Local model training server
Server
Local model training server
Load Balancer
Server

Figure 2: Topology of network with a mixture of SDN and federated learning.

(2) Communication overhead: federated learning entails the transmission of information between a central server and the participating clients throughout each iteration. The temporal complexity of communication is contingent upon several elements, including but not limited to network capacity, the number of clients, and the size of model changes. The time complexity can be expressed as $O(K)$, where K is the number of communication rounds.

In general, the temporal complexity of federated learning is mostly influenced by the training time required for each client and the number of communication rounds involved.

3.2. Space Complexity

(1) Model size: the space complexity of the training process is predominantly determined by the dimensions of the machine learning model. This encompasses the structural design of the model as well as the quantity of parameters involved. The space complexity required for storing the model may be expressed as $O(M)$, where M is the total number of parameters in the model.

(2) Local data storage: the storage of local data by each client contributes to the overall space complexity. The spatial complexity for local data storage may be expressed as $O(N)$, where N is the quantity of local data samples.

(3) Model updates: in the context of federated learning, the process involves the communication of model updates between the central server and the clients. The space complexity required for storing these updates is contingent upon their size and may be expressed as $O(M)$.

The primary determinant of the space complexity in federated learning is the size of the model and the storage demands for local data. It is noteworthy that federated learning has been specifically devised to mitigate the necessity of exchanging raw data between clients and the central server, hence contributing to the reduction of space complexity associated with data storage.

4. Results and Evaluation

The primary objective of the proposed method is to enhance QoS metrics while maintaining data privacy on networks. The proposed method employs federated learning, a secure mechanism for executing machine learning algorithms, since it never transmits raw data outside the network. Instead, it uses communication rounds to transmit local models to the aggregator server and generate a global model. In addition, the proposed method was implemented on three servers using the open load test tool to evaluate its performance and compare it with other load-balancing methods based on metrics such as CPU and memory utilization. By conducting performance evaluation tests, we can obtain valuable insights into the proposed method's effectiveness and identify areas where further improvements may be necessary. The use of federated learning and performance evaluations provides a comprehensive approach to optimizing QoS metrics while preserving data privacy in network environments.

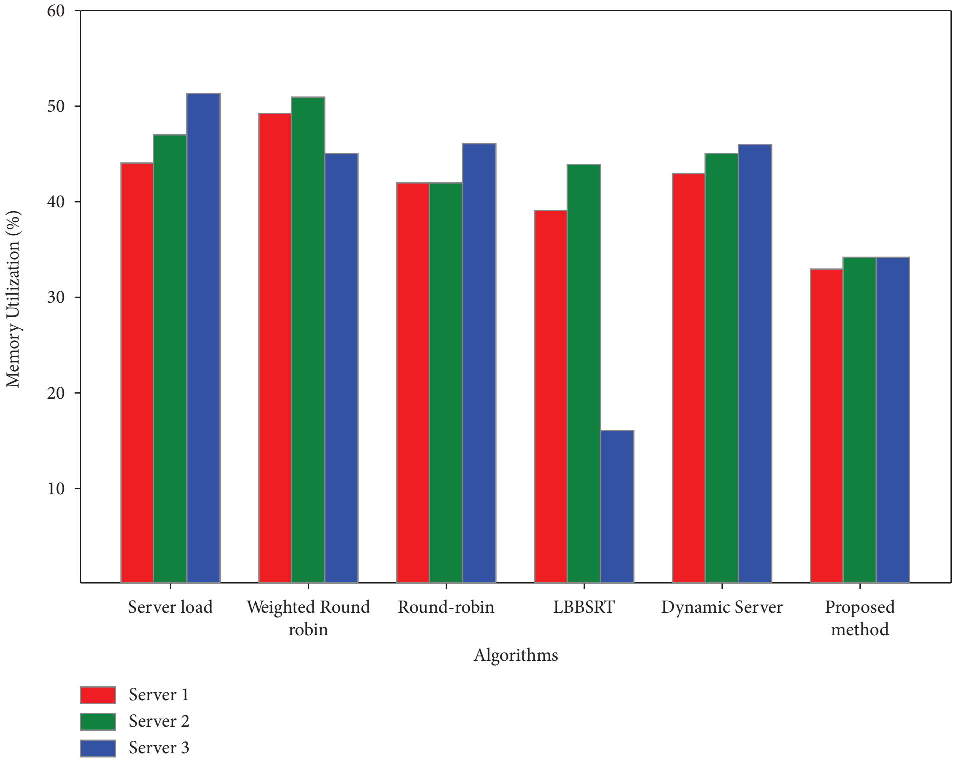

Figure 3: Comparison of memory usage.

Table 1: The improvement percentage of the proposed method on various servers' requests according to memory usage.

	Proposed method over round-robin (%)	Proposed method over weighted round-robin (%)	Proposed method over server load (%)	Proposed method over LBBSRT (%)	Proposed method over dynamic server (%)
Server 1	4	12	8	5	5
Server 2	3	11	6	4	4
Server 3	6	10	9	4	3

In our research, we conducted extensive simulations to assess the performance of our proposed method and compare it with the state-of-the-art approaches. To create a robust and reproducible experimental setup, we utilized the Google Colab Pro Plus environment, a cloud-based platform with access to high-performance GPUs and TPUs, while implementing our algorithms in the Python programming language. Key Python libraries, including TensorFlow, PyTorch, and Scikit-learn, were employed to facilitate deep learning, machine learning, and data preprocessing tasks. We carefully curated a relevant dataset encompassing diverse network traffic scenarios and QoS metrics to evaluate our approach comprehensively. Our experiments adhered to rigorous protocols, including data preprocessing, model training, hyperparameter tuning, and cross-validation, with multiple runs to account for variations in random initialization and data splits. This setup ensures transparency and reproducibility in our research, enabling a thorough evaluation of our proposed QoS optimization method in SDN.

The memory utilization of several load-balancing algorithms, including the proposed method, weighted round-robin algorithm, round-robin algorithm, LBBSRT, and dynamic server approach, were compared and evaluated in terms of their efficiency. The results are shown in Figure 3 and summarized in Table 1. The proposed algorithm outperformed the other methods in memory usage on all three servers, with improvements of 4%, 12%, 8%, 5%, and 5% on server 1 compared to the round-robin, weighted round-robin, server load, LBBSRT, and dynamic server methods, respectively. Similarly, the proposed method achieved memory utilization improvements of 3%, 11%, 6%, 4%, and 4% on server 2 and 6%, 10%, 9%, 4%, and 3% on server 3, in the same order as for server 1. These results demonstrate the effectiveness of the proposed method in optimizing memory utilization and load balancing.

In distributing network traffic load on servers, CPU utilization plays a vital role. As shown in Figure 4, prior methods could not distribute network traffic load uniformly, resulting in uneven CPU utilization across the servers. However, in the proposed method, the network traffic is uniformly distributed across all servers, resulting in almost equal CPU utilization. Table 2 shows that the proposed method outperforms other methods in terms of CPU

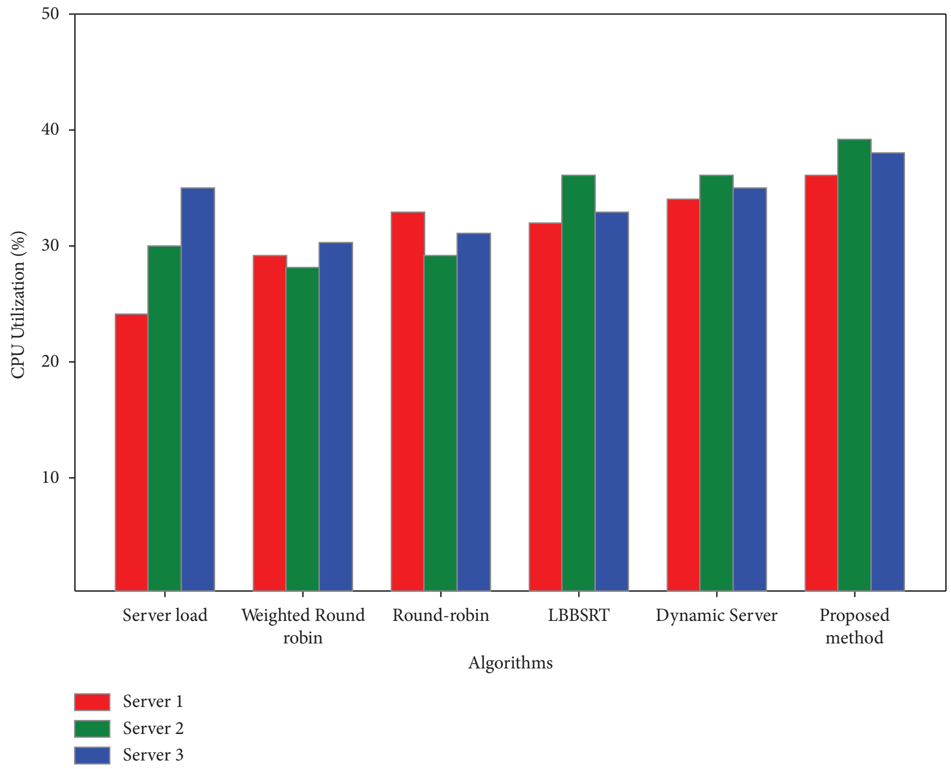

Figure 4: Comparison of CPU usage.

Table 2: The improvement percentage of the proposed method on various servers' requests according to CPU usage.

	Proposed method over round-robin (%)	Proposed method over weighted round-robin (%)	Proposed method over server load (%)	Proposed method over LBBSRT (%)	Proposed method over dynamic server (%)
Server 1	6	12	24	6	2
Server 2	14	16	18	5	4
Server 3	9	13	12	5	4

utilization, improving it by 6%, 12%, 24%, 6%, and 2% on server 1 when compared with round-robin, weighted round-robin, server load, LBBSRT, and dynamic server methods, respectively. In addition, the proposed method improved CPU utilization by 14%, 16%, 18%, 5%, and 4% on server 2 and 9%, 13%, 12%, 5%, and 4% on server 3 compared with the same methods.

5. Conclusion

Software-defined networking (SDN) is a new paradigm aiming to make network management scalable and customizable. Unlike conventional load balancers or network managing tools, SDN does not need particular infrastructure and can work with any network infrastructure. Quality of service (QoS) is an essential aspect of any system, and SDN aims to improve QoS factors in networks such as throughput, response time, and resource utilization. This paper aimed to use federated learning as a data-preserving approach to enhance results for the genetic algorithm to improve QoS metrics in networks. We used a central server as our aggregator plan.

After receiving all local models from the training plan, it aggregates them by the MOE algorithm and then broadcasts the global model to all local training servers. This well-trained global model performs load-balancing and route finding by the genetic algorithm for packets.

We used the open load test tool to generate traffic on three servers to test this approach. Results show that the proposed approach improves memory utilization compared to conventional load-balancing techniques. CPU is the core for distributing network traffic on servers, and it reduces QoS if it is not utilized uniformly in servers. Results describe that the proposed method improves CPU Utilization by using a well-trained global model with the genetic algorithm.

Our ongoing study is to contribute to the advancement of QoS improvement in SDN. It serves as a foundation for exploring several prospective avenues in this sector. The combination of generative learning, incremental learning, and recurrent learning ideas has the potential to significantly enhance the capabilities and adaptability of SDN-based multimedia traffic management.

An area of potential future investigation is the utilization of generative learning models, namely, generative adversarial networks (GANs) and variational autoencoders (VAEs). The incorporation of these methodologies into our SDN architecture has the potential to enhance its capabilities in efficiently handling multimedia traffic and generating synthetic multimedia content. The increased range of capabilities provided by this development allows for the exploration of many applications, such as enhancing content and identifying anomalies, within the framework of SDN. Consequently, our methodology gains more depth and variety.

In dynamic network contexts, the capacity to adapt and engage in continual learning is of utmost importance. Subsequent investigations will focus on exploring incremental learning methodologies, enabling our SDN solution to adaptively revise its models and rules in response to evolving network circumstances. The capacity to adapt is crucial in maintaining the effectiveness and resilience of QoS optimization in modern networks, as it allows for the accommodation of the constantly evolving network environment.

Recurrent learning models such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have proven to be successful in capturing temporal dependencies in multimedia traffic patterns. By integrating these recurring learning principles, our SDN solution has the capability to effectively predict and handle variations in network traffic, leading to enhanced QoS, especially in situations when the network is very dynamic.

The aforementioned future research directions are in accordance with the continuous development of SDN and machine learning. They hold the potential to enhance and expand the capabilities of our proposed solution. Our objective is to expand the limits of SDN-based multimedia traffic management by the adoption of generative, incremental, and recurrent learning approaches. The proposed extension would enhance our methodology's capability to address more intricate and varied difficulties in contemporary networking settings. This will contribute to the ongoing progress of QoS optimization and the effective exploitation of network resources inside SDN.

Data Availability

The data used to support the findings of the study are available from the corresponding authors upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. I. Hamed, B. M. ElHalawany, M. M. Fouda, and A. S. T. Eldien, "A new approach for server-based load balancing using software-defined networking," in *Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 30–35, IEEE, Cairo, Egypt, December 2017.

[2] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[3] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] A. K. Arahunashi, G. G. Vaidya, S. Neethu, and K. V. Reddy, "Implementation of server load balancing techniques using software-defined networking," in *Proceedings of the 2018 3rd international conference on computational systems and information technology for sustainable solutions (CSITSS)*, pp. 87–90, IEEE, Bengaluru, India, December 2018.

[5] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A load-balancing mechanism for distributed SDN control plane using response time," *IEEE transactions on network and service management*, vol. 15, no. 4, pp. 1197–1206, 2018.

[6] S. Zhang, J. Lan, P. Sun, and Y. Jiang, "Online load balancing for distributed control plane in software-defined data center network," *IEEE Access*, vol. 6, pp. 18184–18191, 2018.

[7] A. Khaliq, S. H. Adil, and J. Jamshid, "Enhancing throughput and load balancing in software-defined networks," in *Proceedings of the 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–6, IEEE, Sukkur, Pakistan, March 2018.

[8] Y.-J. Chen, L.-C. Wang, M.-C. Chen, P.-M. Huang, and P.-J. Chung, "SDN-enabled traffic-aware load balancing for M2M networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1797–1806, 2018.

[9] F. Ongaro, E. Cerqueira, L. Foschini, A. Corradi, and M. Gerla, "Enhancing the quality level support for real-time multimedia applications in software-defined networks," in *Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 505–509, IEEE, Garden Grove, CA, USA, February 2015.

[10] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN controller assignment in data center networks: stable matching with transfers," in *Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, San Francisco, CA, USA, April 2016.

[11] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, and S. Civanlar, "Scalable video streaming over OpenFlow networks: an optimization framework for QoS routing," in *Proceedings of the 2011 18th IEEE international conference on image processing*, pp. 2241–2244, IEEE, Brussels, Belgium, September 2011.

[12] A. Shirmarz and A. Ghaffari, "Automatic software defined network (SDN) performance management using TOPSIS decision-making algorithm," *Journal of Grid Computing*, vol. 19, no. 2, pp. 16–21, 2021.

[13] A. Shirmarz and A. Ghaffari, "A novel flow routing algorithm based on non-dominated ranking and crowd distance sorting to improve the performance in SDN," *Photonic Network Communications*, vol. 42, no. 3, pp. 167–183, 2021.

[14] A. Shirmarz and A. Ghaffari, "Network traffic discrimination improvement in software defined network (SDN) with deep autoencoder and ensemble method," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 5, pp. 6321–6337, 2023.

[15] S. Manzoor, Z. Chen, Y. Gao, X. Hei, and W. Cheng, "Towards QoS-aware load balancing for high density software defined

Wi-Fi networks," *IEEE Access*, vol. 8, pp. 117623–117638, 2020.

[16] A. Shirmarz and A. Ghaffari, "An autonomic software defined network (SDN) architecture with performance improvement considering," *Journal of Information Systems and Telecommunication (JIST)*, vol. 8, no. 30, pp. 121–129, 2020.

[17] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7545–7593, 2020.

[18] S. Manzoor, K. B. Bajwa, M. Sajid et al., "Modeling of wireless traffic load in next generation wireless networks," *Mathematical Problems in Engineering*, vol. 2021, Article ID 7293093, 15 pages, 2021.

[19] R. Thenmozhi, T. Preethi, S. Sadhana, V. Shruthi, and B. Yuvarani, "Integrating multimedia services over software defined networking," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, 2017.

[20] A. T. Oliveira, B. J. C. Martins, M. F. Moreno, A. B. Vieira, A. T. A. Gomes, and A. Ziviani, "SDN-based architecture for providing QoS to high performance distributed applications," in *Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 00602–00607, IEEE, Natal, Brazil, June 2018.

[21] Y. Liu, A. Huang, Y. Luo et al., "Fedvision: an online visual object detection platform powered by federated learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 8, pp. 13172–13179, New York, NY, USA, February 2020.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, Seattle, WA, USA, February 2017.