Smartly Managing Traffic and Latency in a Content-Based data center using Modified Packet Headers, Machine Learning, Load Balancing, and Network Telemetry

Souryendu Das[1], Andrew White[2], Malcolm Lyn[3], and Stavros Kalafatis[4]

[1,2,3,4]Electrical and Computer Engineering, Texas A&M University
[1]souryendu@gmail.com
[2]andygwhite@tamu.edu
[3]mlyn148@tamu.edu
[4]skalafatis-tamu@exchange.tamu.edu

***Abstract*—This paper introduces SRLBA, an approach for optimizing request management in data centers, specifically addressing the challenges of routing and load balancing within server racks. Our primary objective is to derive efficient routing and load-balancing decisions that mitigate oversubscription, ensuring effective server utilization. SRLBA leverages Software-Defined Networking (SDN) and established Machine Learning (ML) techniques to classify packets and balance traffic. The method integrates SDN-based switches, OpenFlow for packet header modification, and ML algorithms for traffic classification and workload prediction. Unlike approaches relying on reinforcement learning (RL), SRLBA exclusively utilizes supervised learning, ensuring efficiency and scalability within modern data center environments. We demonstrate SRLBA's effectiveness using custom and public datasets, offering an innovative solution to request management through the integration of SDN and ML technologies.**

***Keywords*—Routing, Load Balancing, SDN, Mininet, Machine Learning, Packet Header Modification, Traffic Classification, Fairness**

I. Introduction

The rapid growth of internet traffic and the increasing complexity of data center workloads have made efficient routing and load balancing crucial for modern data centers [1]. Oversubscription, where incoming requests exceed server capacities, remains a significant challenge, leading to performance bottlenecks and reduced Quality of Service (QoS).

To address this, we propose the Smartly Routing and Load Balancing Algorithm (SRLBA), which integrates Software-Defined Networking (SDN) with established Machine Learning (ML) techniques to derive both routing and load-balancing decisions intelligently. The goal is to optimize traffic distribution across servers, minimize latency, and enhance overall network efficiency [2]. By leveraging supervised learning techniques, SRLBA classifies traffic based on real-time network metrics and workload characteristics, making data-driven decisions to optimize resource utilization.

Unlike reinforcement learning (RL) approaches, which may provide better adaptability to unforeseen traffic patterns but often incur higher computational overhead, SRLBA prioritizes efficiency and scalability by exclusively utilizing supervised learning for predictable data center environments. Future work could explore hybrid models to enhance adaptability further.

To validate SRLBA's performance, we tested the system on both custom and public datasets, demonstrating its ability to handle a wide range of traffic patterns, including encrypted packets, and significantly improve network performance.

II. Related Work

Research on traffic management in SDN-based data centers has evolved significantly in recent years, with a growing focus on machine learning-driven methods for improving routing and load balancing. Traditional routing algorithms, such as shortest path routing and static load balancing, provided simple and efficient solutions under predictable conditions. However, their lack of adaptability to dynamic traffic patterns, as highlighted by early works [3], [4], made them unsuitable for modern data center environments.

Recent studies have extensively applied machine learning (ML) techniques to enhance SDN-based load balancing and traffic management. For instance, Wang et al. [5] proposed a hybrid learning framework that combines supervised and reinforcement learning to achieve efficient traffic distribution and adaptability. Their method demonstrated improved scalability and adaptability compared to static and purely supervised approaches.

Deep learning methods, particularly those leveraging neural networks, have also been explored. Nguyen et al. [6] introduced a deep reinforcement learning-based approach for adaptive load balancing in SDN, achieving significant reductions in latency and packet loss under dynamic traffic conditions. Similarly, Ali et al. [7] utilized convolutional neural networks (CNNs) for traffic classification, optimizing routing and load balancing decisions in real-time. However, these methods often come with higher computational requirements, posing challenges for deployment in resource-constrained environments.

Supervised learning has remained a popular choice for its simplicity and efficiency in scenarios requiring low-latency decision-making. Zhang et al. [8] explored the use of gradient boosting techniques for traffic classification and load balancing in edge computing environments. Their

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

work highlighted the effectiveness of supervised methods in handling predictable traffic patterns while achieving high throughput and reduced congestion.

In addition to ML-based methods, hybrid approaches integrating ML with traditional techniques have gained attention. Chen et al. [9] proposed a hybrid algorithm that combines ML-based prediction with rule-based routing, striking a balance between adaptability and computational efficiency. This approach aligns closely with the goals of SRLBA, focusing on real-time adaptability without introducing excessive overhead.

Our work builds on these recent advancements by introducing SRLBA, a machine learning-based solution that integrates supervised learning within SDN controllers for real-time traffic management. Unlike deep learning methods, which may require extensive computational resources, SRLBA prioritizes efficiency and scalability. By combining supervised learning for traffic classification with adaptive load balancing, SRLBA addresses the gaps left by static routing methods [3], [4] and computationally intensive approaches [6], [10].

Comparing SRLBA with prior studies highlights its contribution to advancing traffic management in SDN environments. While previous works have achieved significant improvements in either classification accuracy or dynamic adaptability, SRLBA offers a comprehensive approach that seamlessly integrates these aspects. By leveraging real-time data from M-Lab's internet performance metrics and implementing lightweight supervised learning models, SRLBA achieves scalability and efficiency, making it suitable for modern data center environments.

Future work can extend these contributions by exploring reinforcement learning or hybrid approaches to further enhance adaptability, as suggested in [9]. However, those considerations are beyond the scope of this manuscript.

III. System Model

The SRLBA system integrates SDN with established machine-learning techniques to manage traffic efficiently in data centers. Fig. 1 presents the architecture, which relies on hardware-based solutions like programmable switches or FPGA-based switches to ensure scalability and performance, an essential requirement in modern Clos-based data centers.

SRLBA's objective is to derive routing and load-balancing decisions by classifying and routing traffic based on real-time workload characteristics. The system uses supervised learning models to predict workload types and route traffic to the most suitable hosts, ensuring balanced resource usage and reducing congestion. This approach helps mitigate oversubscription issues and enhances the Quality of Service (QoS) within the network.

The model comprises 12 host machines grouped into CPU-intensive, network-intensive, and memory-intensive clusters. The SDN controller analyzes packet-level information and workload requirements to classify incoming packets and route them to the best-fit machine.

Key components of SRLBA:

- **Packet Modification:** Hosts generate IPv4 packets with modified headers to enable traffic classification.
- **SDN Controller and Switch:** The system includes an OpenFlow v1.4 switch and a Ryu-based controller for managing data and control planes.
- **Machine Learning Integration:** The controller integrates ML models for packet classification, updating the switch's flow table with routing rules.
- **Flow Management:** The controller routes packets based on real-time traffic classification from the ML models.
- **Host Machines:** Hosts provide task-specific processing and feedback to the ML models, helping optimize future decisions.
- **Cross Traffic Generator:** Provides real-time network data, enhancing the controller's traffic routing decisions.
- **Handling Encrypted Packets:** The system is capable of processing encrypted packets, a growing need in modern data centers.

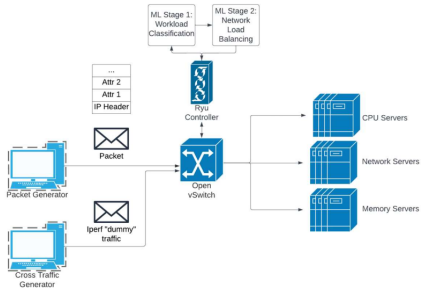

Fig. 1: System Model

IV. Machine Learning Implementation

SRLBA's machine learning framework is divided into two stages: traffic classification and load balancing. Both stages leverage supervised learning techniques optimized for the high-frequency demands of data center environments.

A. Traffic Classification

In the first stage, logistic regression, SVM, MLP, and linear regression models classify network traffic based on packet length, content category, timestamp, time in queue, priority, and user permissions. These parameters are embedded in the IP header's options field for efficient routing decisions [11], [12]. The models predict the most suitable server cluster (CPU, network, or memory-intensive) for processing.

Category	% Traffic	Category No.
Not Specified	N/A	0
Video	57.7	1
Web	17.0	2
Gaming	7.8	3
Social Media	5.1	4
Content Sites	4.6	5
File Sharing	2.8	6
Audio Streaming	1.0	7

TABLE I: Internet Traffic Categories and Shares

B. Queue and Network Parameters

The "time in queue" parameter measures the delay a packet experiences in the input queue before forwarding, calculated using timestamps. Additionally, latency, bandwidth, and congestion metrics are collected from the controller-server links to inform routing and load-balancing decisions, ensuring real-time traffic management.

C. Load Balancing

The second stage selects the optimal server within a cluster based on real-time metrics such as latency, bandwidth, congestion, and host utilization. The system also leverages M-Lab internet performance data to enhance routing efficiency [13].

- **Latency, Bandwidth, Congestion:** Dynamically monitored to optimize routing decisions.
- **Packet Retransmission Rate:** Reflects network congestion levels for adaptive routing.
- **Link and Host Utilization:** Ensures balanced traffic distribution and prevents bottlenecks.

D. Model Selection and Rationale

We selected logistic regression, SVM, MLP, and linear regression due to their speed and suitability for high-frequency decisions. Each model was evaluated for accuracy and efficiency using data center traffic patterns.

1) Logistic Regression: Efficient for categorical classification, logistic regression uses probability scores to map inputs to server clusters in real time.

2) Support Vector Machine: SVM establishes decision boundaries between traffic categories and handles both linear and non-linear data effectively [14].

3) Multilayer Perceptron: MLPs are flexible, managing complex relationships in traffic data to improve classification [15].

4) Linear Regression: A custom linear regression model is used as a baseline for predicting traffic characteristics, though less effective than logistic regression for classification [16].

5) Model Clarifications: While linear regression is traditionally used for regression tasks, SRLBA includes it as a comparative baseline. In contrast, logistic regression is tailored for classification by mapping input features to probabilities using a sigmoid function, making it more suitable for traffic classification.

E. Traffic Dataset and Training

The dataset used includes both custom-generated traffic patterns and publicly available data. Custom datasets simulate real-world workloads (CPU, network, memory-intensive), while public datasets validate the model's performance across diverse conditions. Features like packet length, content type, and QoS parameters are embedded in packet headers for classification. The models were trained using labeled data to predict which server cluster should handle the traffic, with cross-validation and grid search optimizing model performance.

F. Model Inputs and Outputs

SRLBA's models take inputs such as packet length, content category, time in queue, and user priority to classify traffic into CPU, network, or memory-intensive clusters. The output is the predicted cluster, which the SDN controller uses to update the switch's flow table and route the packet accordingly.

G. IPv4 Header Modification

To facilitate traffic classification, SRLBA modifies the IPv4 header by embedding metadata involving content category, priority, and QoS parameters. This is achieved using the options field in the IPv4 header. Traffic metadata is added to the IPv4 header with minimal processing overhead, maintaining compatibility with existing network protocols.

H. Security and Compatibility Considerations

While the modification of IPv4 headers enables effective traffic classification and routing, it also introduces potential security and compatibility challenges. These challenges are carefully addressed in SRLBA to ensure robust performance without compromising network security or interoperability:

- **Security Measures:** To prevent vulnerabilities such as spoofing or tampering with packet headers, SRLBA employs cryptographic mechanisms to authenticate and verify modified headers. The system utilizes secure hashing (e.g., HMAC) to generate integrity checks for each modified header, ensuring that unauthorized changes can be detected.
- **Compatibility with Legacy Systems:** SRLBA is designed to operate within hybrid environments where legacy systems coexist with modern SDN-based components. To maintain compatibility, header modifications are implemented in a way that complies with the standard IPv4 options field, avoiding disruptions to legacy systems that might misinterpret non-standard modifications.
- **Minimal Overhead:** The modifications to IPv4 headers are minimal and adhere to fields within the header reserved for optional extensions. This approach ensures that the changes do not violate the IPv4 specification, maintaining compatibility across different network devices and software versions.

These measures collectively enhance SRLBA's reliability and security in handling real-world network scenarios, making it a practical solution for modern data center environments.

I. Implementation Workflow and Configuration Details

The implementation of SRLBA involves the following steps:

1) **Environment Setup:** Mininet is used to emulate the data center network, including programmable switches and host machines. A Ryu controller configured with OpenFlow v1.4 manages the SDN environment.
2) **Feature Extraction:** Packet-level information, such as packet length, timestamp, and QoS parameters, is extracted and embedded into the IPv4 header using the options field.
3) **Model Training:** Supervised learning models (logistic regression, SVM, and MLP) are trained using a combination of custom-generated traffic patterns (simulating CPU, network, and memory-intensive workloads) and public datasets. Training is performed offline with labeled data to ensure accuracy.
4) **Integration with SDN Controller:** The trained models are integrated into the Ryu controller. The controller processes incoming packets, classifies

traffic using the models, and dynamically updates the flow table in OpenFlow switches.

5) **Real-Time Traffic Management:** During runtime, the SDN controller collects real-time metrics (latency, bandwidth, congestion) and uses them to refine routing and load-balancing decisions.

Practical Configuration Example:

- The IPv4 options field is modified using custom Python scripts running on the SDN controller, ensuring minimal processing overhead.
- Traffic classification is implemented using scikit-learn, and the trained models are serialized using joblib for fast integration with the SDN controller.
- Flow rules in OpenFlow switches are updated using the Ryu API, leveraging RESTful commands for seamless communication between the data and control planes.

This detailed workflow ensures that SRLBA is practical for real-world deployment and demonstrates its effectiveness in dynamic data center environments.

V. Results and Analysis

A. Classification Stage

The classification accuracy of each of the models is shown in Figure 2, and the latency each model introduces is shown in Table II. All models exhibit similar performance, with linear regression having nearly the highest degree of accuracy and introducing the lowest latency to SRLBA. Logistic regression, MLP, and SVM also demonstrated strong performance in balancing classification accuracy with latency.

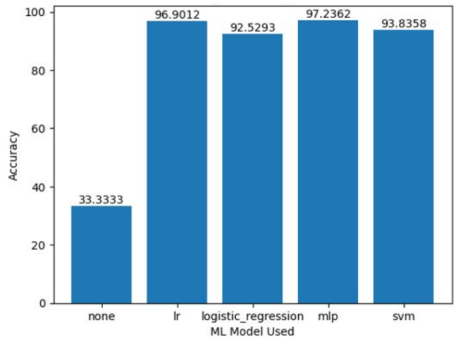

Fig. 2: First Stage Accuracy Results

Model	Average Packet Handling Time (ms)
No model	2.63
Lin. Regression	3.20
Log. Regression	3.80
MLP	3.65
SVM	3.75

TABLE II: First Stage Timing

1) Cross Traffic: We assessed the impact of cross-traffic on experimental traffic by running an iperf speed test on a 1Gbps link while sending the dataset across the same link with a packet generator. Cross-traffic increased congestion, leading to higher latency for the experimental traffic. To accurately measure latency, tcpdump was used on both the packet generator and receiver machines, filtering out cross-traffic. Latency was calculated from the timestamps of sent and received packets. Two trials were conducted, and their averaged results provided a reliable estimate of the cross-traffic's impact on performance, as shown in Fig. 3.

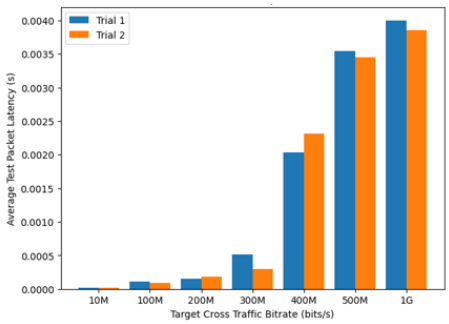

Fig. 3: Cross Traffic Experiment Results

2) Resource Utilization: The average CPU utilization for each model is shown in Fig. 4. CPU usage above 100% indicates the need for multiple cores. Table III shows the average memory utilization for each model.

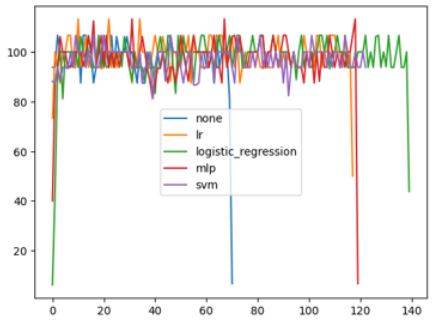

Fig. 4: Classification Stage: Average CPU Utilization

Model	Average Memory Utilization
No model	2.70%
Lin. Regression	2.70%
Log. Regression	3.10%
MLP	3.10%
SVM	3.10%

TABLE III: Classification Stage: Average Memory Util.

B. Load Balancing Stage

1) Link and Host Utilization: We evaluated the load-balancing algorithms by assessing how effectively they distributed traffic across servers and links within CPU, network, and memory clusters. Key experiment details are as follows:

- **Packet Validation Datasets:** CPU, network, and memory validation datasets with QoS parameters were sent to the controller, which used the load-balancing models.
- **Single Cluster Testing:** Each algorithm was tested on one cluster at a time to observe host and link utilization.
- **Host Utilization Results:** Fig. 6 shows CPU host utilization. Logistic regression, SVM, and MLP distributed workloads evenly compared to round-robin, while linear regression performed worse.
- **Average Host Utilization:** Table IV shows the average host utilization per model, with similar results across clusters.

- **Link Utilization Results:** Fig. 5 shows link utilization for each algorithm. Round-robin distributed traffic equally, while the ML models preferred less-utilized links, with MLP performing best under 40% utilization.
- **Average Link Utilization:** Table V shows that ML algorithms used resources more efficiently by selecting less-utilized links.

Overall, logistic regression, SVM, and MLP balanced the workload more evenly across servers, with linear regression underperforming. MLP exhibited the best behavior in selecting less-utilized links and optimizing resource distribution.

Model	Average Host Utilization
No model	48%
Lin. Regression	58%
Log. Regression	38%
MLP	38%
SVM	38%

TABLE IV: Average Host Utilization Among Models

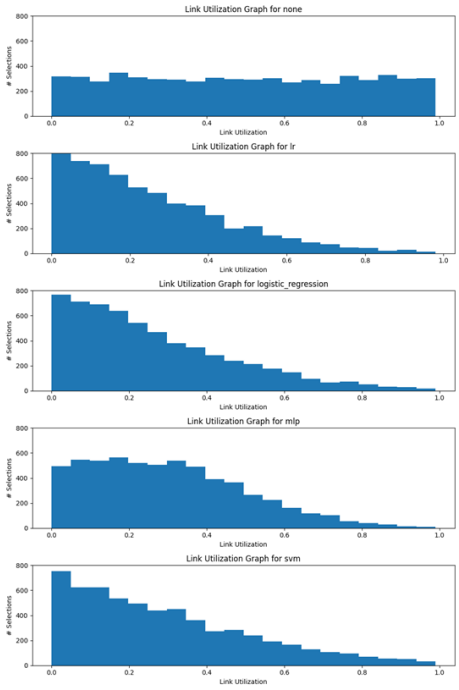

Fig. 5: Link Utilization Results

Upon further analysis, both logistic regression and SVM exhibited similar link utilization patterns, as they prioritize balancing link loads and minimizing congestion, leading to a nearly identical performance in our network setup (Fig. 5).

Model	Average Link Utilization
No model	49%
Lin. Regression	25%
Log. Regression	26%
MLP	30%
SVM	29%

TABLE V: Average Link Utilization Among Models

2) Resource Utilization: The average CPU utilization for each model is shown in Figure 7. The CPU usage was limited to one core, and percentages over 100 indicate the need for the use of more than a single core. The average memory utilization of each model is reported in Table VI.

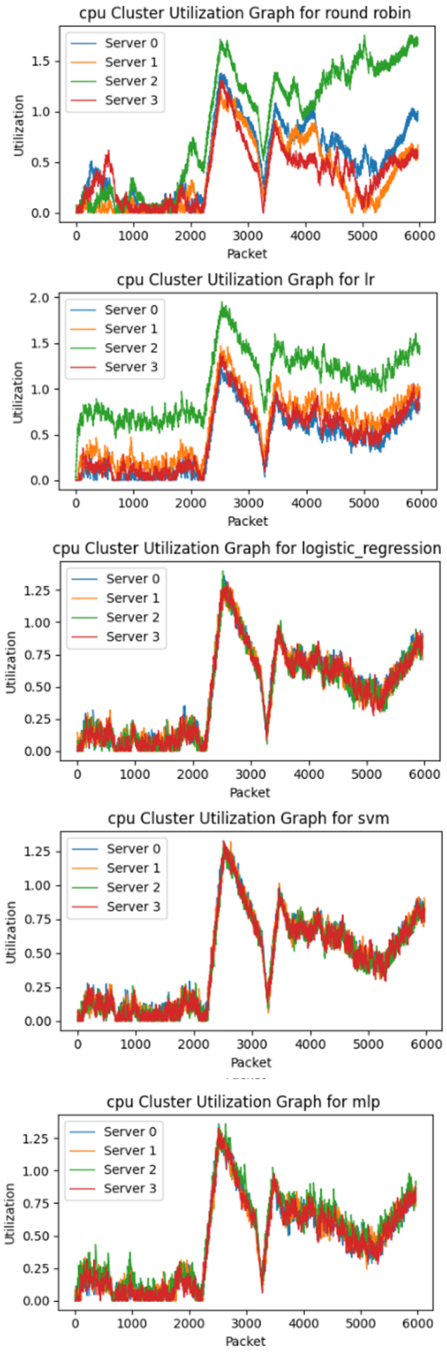

Fig. 6: Server Utilization Results

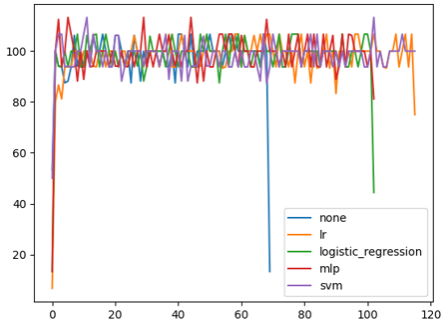

Fig. 7: Load Balance Stage: Average CPU Utilization

Model	Average Memory Utilization
No model	2.70%
Lin. Regression	2.80%
Log. Regression	3.10%
MLP	3.10%
SVM	3.10%

TABLE VI: Load Balance Stage: Average Memory Util.

C. Summary

The load balancing models performed as expected, accurately classifying and routing traffic to appropriate clusters while efficiently distributing it across servers. However, the first stage introduced noticeable latency. Further testing with real-world traffic and hardware is required to evaluate the full impact of routing on congestion reduction. In the second stage, all models except linear regression demonstrated efficient traffic distribution, with the MLP classifier standing out by evenly routing traffic across less-utilized links, ensuring balanced resource usage. Other models showed some bias toward less-utilized links, potentially leading to suboptimal performance. While linear regression performed well in some scenarios, optimizing all models may lead to improved outcomes.

VI. Conclusion

This paper presents SRLBA, a novel methodology for handling network traffic in data centers using SDN and machine learning. Our custom SDN topology and ML-integrated controller demonstrate effective traffic management, reducing latency and optimizing resource utilization. By leveraging the IP header field, we open new possibilities for network performance improvements. While SRLBA's reliance on supervised learning ensures efficiency and scalability in handling predictable traffic patterns, future research could incorporate reinforcement learning or hybrid approaches to address dynamic and unforeseen traffic conditions, further enhancing its adaptability and robustness. It would also include testing SRLBA in live network environments and comparative studies with other methodologies, such as DRL-based approaches, to benchmark its performance. Our research contributes significantly to the field of intelligent routing and load balancing, providing a foundation for further advancements in network traffic management.

References

[1] I. Patronas, A. Kyriakos, and D. Reisis, "Switching functions of a data center top-of-rack (tor)," *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, vol. 84, pp. 364–367, 2016.

[2] W. Tao, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers." *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications.*, pp. 1–9, 2016.

[3] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet load-balanced, low-latency routing for clos-based data center networks," *In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pp. 49–60, 2013.

[4] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," *In Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 503–514, 2014.

[5] A. Alhilali and A. Montazerolghaem, "Artificial intelligence based load balancing in sdn: A comprehensive survey," *Internet of Things*, vol. 22, p. 100814, 2023.

[6] G. Beissenova, A. Zhidebayeva, Z. Kopzhassarova, P. Kozhabekova, B. Myrzakhmetova, M. Kerimbekov, D. Ussipbekova, and N. Yeshenkozhaev, "Load balancing in dcn servers through software defined network machine learning," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 2, 2024.

[7] N. Al-Jamali and H. Al-Raweshidy, "Intelligent traffic management and load balance based on spike isdn-iot," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1640–1651, 2020.

[8] V. Punitha and C. Mala, "Traffic classification for efficient load balancing in server cluster using deep learning technique," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8038–8062, 2021.

[9] W. Liu, J. Lu, J. Cai, Y. Zhu, S. Ling, and Q. Chen, "Drl-plink: Deep reinforcement learning with private link approach for mix-flow scheduling in software-defined data-center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1049–1064, 2021.

[10] H. T. Tran and T. Ho-Phuoc, "Deep laplacian pyramid network for text images super-resolution," *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 1–6, 2019.

[11] S. Adibi, "Traffic classification – packet-, flow-, and application-based approaches," *International Journal of Advanced Computer Science and Applications*, vol. 1, no. 1, p. 6, 2010.

[12] D. M. Divakaran, "A spike-detecting aqm to deal with elephants," *Computer Networks*, vol. 56, no. 13, p. 3087–3098, 2012.

[13] Measurement Lab, "The M-Lab NDT data set," https://measurementlab.net/tests/ndt, (2009-02-11 – 2015-12-21), bigquery table measurement-lab.ndt.download.

[14] R. Berwick, "An idiot's guide to support vector machines (svms)," 2003, accessed: 2023-04-01. [Online]. Available: https://www.svms.org/tutorials/Berwick2003.pdf

[15] B. Akkaya and N. Çolakoğlu, "Comparison of multi-class classification algorithms on early diagnosis of heart diseases," 09 2019.

[16] K. Kumari and S. Yadav, "Linear regression analysis study," *Journal of the Practice of Cardiovascular Sciences*, vol. 4, p. 33, 01 2018.