Dynamic Load Balancing in SDN Using Machine Learning

Sonam Sharma[1], Gagandeep Singh[1]*, and Shubham Kumar[1]

[1] Apex Institute of Technology (CSE),
Chandigarh University, Mohali 140413, Punjab, India

*Corresponding author: gagan.d.singh.1209@gmail.com

Abstract: SDN is a centralized traffic management model with the separation between the control and data planes. Dynamic traffic undermines the conventional load-balancing methods and this brings congestion and underutilization of available bands. This paper provides an outline of a framework that uses the power of Machine Learning (ML) to forecast a congestion problem, as well as maximizing resource utilization. Compared to the reactive methods, the ML model evaluates the past and actual traffic statistics to discover trends and preset routing. It enhances throughput, reduces latency and its allocation effectively resources within the network. Actually, the assessment outcomes demonstrate 20 percent achievement of throughput, 15 percent decrease in latency and 30 percent dropped packets as compared to the traditional methods. These are accompanied by difficulties, such as computing costs and security threats. The future work is thus on reinforcement learning to solve autonomous traffic engineering and maximizing the ML inference in real-time. As cloud computing, IoT, and 5G ecosystems have to scale under any conditions, irrespective of their complexities, the work is one of the steps to develop AI-based networking.

Keywords: Software-Defined Networking, Dynamic Load Balancing, Traffic Optimization, Network Congestion Control.

P. Johri et al. (eds.), *Proceedings of the International Conference on Sustainable Computing and Artificial Intelligence (ICSCAI 2025)*, Advances in Engineering Research 298,
https://doi.org/10.2991/978-94-6239-674-6_18

1 INTRODUCTION

1.1 Background

The digitalization of communication and the development of cloud computing has contributed to a significant increase in the volume of network traffic. The historical network systems, which have a fixed configuration and decentralized control, may not easily cope with changing traffic loads. Software Defined Networking (SDN) has been introduced as a groundbreaking solution that breaks these constraints and allows the separation of the control plane and the data plane so that network resources can be centrally controlled [1]. SDN offers an interface that is programmable enabling the network administrator to dynamically manipulate traffic flows in real-time. In contrast

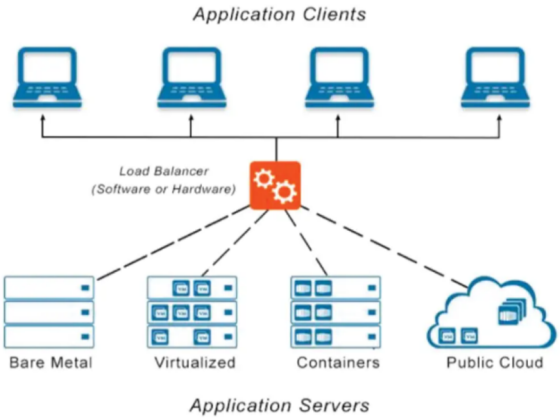


Fig. 1. Load balancer.

to the traditional networks, routing decisions are programmed into the network devices, SDN employs centralized SDN controller to control traffic flow. Such centralization is more agile to the network, efficient in resource distribution, and provides the opportunity to make an intelligent decision due to the existing network conditions [2].

1.2 Load Balancing in SDN

The most important functions involved in the network management are load balancing. It is used to ensure that traffic over the network is spread effectively over several paths to avoid congestion, to optimise bandwidth utilisation and enhance Quality of Service (QoS). In standard networks, load balancing systems are either fixed or reactive, based on set rules or naive instincts. These approaches though do not respond to the changing nature of a network, causing bottlenecks in performance [3]. SDN has enabled load balancing to be done with more efficiency because the controller is capable of tracking network traffic in real-time. SDN controller is capable of dynamically updating the flow tables in the network switches according to real-time statistics of traffic, which will offer an adaptive and scalable policy of distributing traffic. Although these benefits exist, the traditional SDN-based load-balancing techniques continue to be limited in

terms of their ability to forecast future traffic trends as well as proactively alleviating congestion [4].

1.3 Challenges in Traditional Load Balancing

Even though SDN has made progress, there are a number of limitations to traditional load-balancing methods:

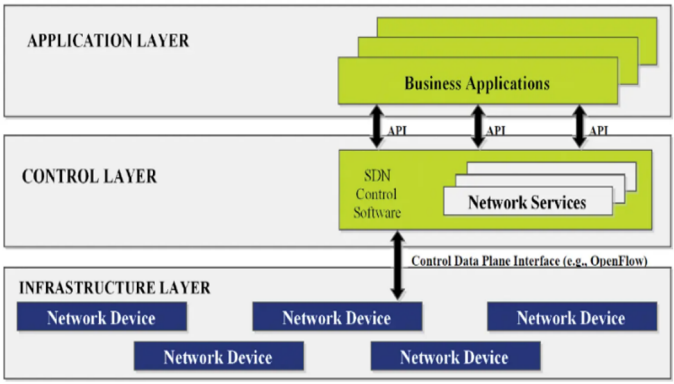
Static Routing Mechanisms: The several load-balancing algorithms are based on pre-written policies that are not responsive to dynamic network traffic.

Reactive Traffic Handling: Most solutions are responsive to congestion after it has been experienced, but not before.

Scalability Problems: Conventional approaches cannot scale to a larger size and complexity of networks as the network scale grows.

Absence of Predictive Capabilities: Traditional systems do not employ the past to forecast network congestion, and hence, make suboptimal decisions [5].

In order to overcome these shortcomings, one of the solutions that can be proposed is to incorporate Machine Learning (ML) into SDN-based load balancing, which will



allow predicting and acting proactively on the traffic [6].

Fig. 2. Software-defined Architecture.

1.4 Role of Machine Learning in Load Balancing

Machine Learning (ML) has received a lot of publicity due to its capability to analyse large volumes of information and make intelligent predictions. Within the framework of SDN load balancing, ML is able to:

Forecast Traffic Changes: MLs can predict congestion prior to occurrence by examining historical network data.

Optimize Traffic Routing: ML algorithms can make real-time choices on the most suitable paths(network) in the network.

Increase Adaptability: ML-based systems can adapt and get better with time, unlike rule-based systems which are not able to learn about behaviour in the network.

Minimize Latency and Packet Loss: ML-based load balancing reduces delays and data loss through proactive utilization of network resources [7].

Decision Trees, Neural Networks, Reinforcement Learning, and Deep Learning are

some of the common ML techniques applied to perform network optimization. The algorithms are able to compute real-time network metrics, detect the pattern of congestion and propose the most optimum route [8].

1.5 Objective of the Study

The main goal of the research is to develop and deploy an ML-based dynamic load-balancing unit on SDN that:
Enhances Network Throughput: This guarantees effective data transfer with minimum delays.
Reduces Latency: Improves user experience through reduction of delays in data processing.
Maximizes the use of Resources: Stabilizes the traffic loads to avoid overloading certain routes in the network.
Improves Adaptability: enables the network to respond dynamically to the real-time traffic conditions [9]. Through these, the study will seek to add to the creation of intelligent, efficient and resilient network infrastructures that are able to sustain the needs of contemporary digital communication and cloud computing systems [10].

2 Related Work

2.1 Traditional Load Balancing Techniques

Traditional load-balancing systems in network have predetermined rules plus fixed algorithms to distribute the traffic to diverse servers or network paths. The commonly used methods are:
Round-Robin Load Balancing: In this technique, the connections are distributed among servers cyclically in a circular fashion. Although it is straightforward and simple to install, it fails to consider the server load or network congestion causing possible bottlenecks [11]. Least-Connections Load balancing: Traffic is sent to the server or network path having the number of active connections lowest. This approach is more flexible than the round-robin and fails to take into account other aspects like the response time of the servers or the real-time fluctuations in traffic [12].
Hash-Based Load Balancing: It is a load balancing method that involves the use of hashing algorithms to distribute network traffic according to definite parameters such as IP addresses or session numbers. It is a persistent approach in a session, though lacks suitability in dynamic traffic fluctuations [13]. Even though they can work well in small and static settings, they are ineffective in distributing traffic in highly dynamic and large traffic networks. Their inefficiency is due to their inability to forecast traffic behaviour and optimize routing in real-time, especially in SDN applications where the traffic patterns keep changing [14].

2.2 ML-Based Load Balancing

The current developments in machine learning have facilitated smart and dynamic load-balancing processes that remain better than conventional methods. Some ML models have been studied in the SDN-based traffic management:
Supervised Learning: Network traffic classification and predicting the level of congestion based on past data have been carried out using Decision Trees, Support

Vector Machines (SVM) and Random Forests [15].
Deep Learning Methods: Intricate traffic patterns have been examined with the help of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks and made in advance routing decisions about the traffic [16].
Reinforcement Learning (RL): SDN controllers can learn ideal load-balancing policies by trial and error during reinforcement learning, which includes Q-learning and Deep Q Networks (DQN) [17]. Such ML-based methods have demonstrated encouraging performances related to better performance of a network, a shorter latency, and efficient resource usage. Nevertheless, they also present issues of computational overhead, model training time and complexity of integration in SDN controllers [18].

2.3 Gaps in Existing Research

The even though the ML-based load balancing has improved, there are a number of challenges that are still not addressed:
Dynamic Response: Most of the existing models need much training and cannot effectively respond to sudden spikes in traffic or the emergence of new network conditions in real time [19].
Scalability Problems: ML models that have been trained on smaller data sets might not be able to be effectively utilized in large-scale SDN systems [20].
Computational Complexity: Deep learning models are also accurate but very demanding in terms of computation and can hardly be implemented in resource-constrained networks in real-time [21].
Absence of Standardized Datasets: The majority of studies use artificial or small real-world datasets, therefore, it is challenging to compare models to various network settings [22]. It is important to fill these gaps in order to facilitate a functional ML-based load-balancing system capable of scaling as well as dynamically responding to dynamic SDN traffic. The methodology, experiment setting and evaluation metrics of our proposed approach are as follows.

3 PROPOSED METHODOLOGY

3.1 System Architecture

The suggested system architecture will provide an integrated approach in SDN and ML-based load balancing to support adaptable and intelligent traffic flow control. The basic elements of this architecture are:
SDN Controller: The SDN controller acts as the central intelligence used to coordinate network traffic by collecting real-time network metrics, flow rule processing as well as dynamic recalibration of routing paths in response to changes in congestion. OpenFlow is the main protocol used to facilitate ease of communication between the controller and network switches to provide fine control of data forwarding policies [23].
Network Traffic Monitor: This module is a systematic type of capturing real-time network performance data, including the bandwidth usage, the delay of the packets transmission, and the hotspots of the congestions. It also covers a steady network data recording and examination to offer important data that helps in developing the ML-based anticipated frameworks of antecedent traffic control [24]. ML Model: The machine learning model is charged with the responsibility to forecast the trends of the

network congestion based on history of the network traffic besides the detection of the complex trends. Under such prediction, the model generates the optimum routing paths, which the SDN controller uses to achieve dynamic load balancing [25].

3.2 Data Collection and Preprocessing

To create a powerful ML model that will be capable of optimizing the network traffic flow, it is necessary to collect and preprocess data in detail. The steps of methodology followed are:

Collection of Traffic Data: Mininet which is a sophisticated SDN emulator is applied to retrieve network traffic data and ensure development of virtualized network topology. Also, Wireshark is installed to monitor the real-time packet flow, which offers a finer view of network behaviours and anomalies [26].

Feature Engineering: The choice of parameters of the network is essential to increase the predictive ability of ML models. The main characteristics of throughput, latency, packet loss, jitter, and queue length are obtained and refined as input features to model training [27].

Data Cleaning: Raw traffic data are preprocessed in order to remove inconsistencies, correct missing values and normalize numerical attributes. This move guarantees the integrity of the data and improves the reliability of the training data, which increases the ability to generalize the models [28].

3.3 Machine Learning Model

ML model is the predictive core to apply the dynamic load balancing in SDN. The process of development includes the following:

Selection of algorithm: A number of supervised learning algorithms are discussed in terms of their effectiveness in congestion prediction, such as Decision Trees, Random Forest, and Long Short-Term Memory (LSTM) networks. The evaluation of each algorithm is done on the basis of the efficiency in calculations and accuracy in prediction [29].

Training and Validation: The data is divided into 80 percent and 20 percent training and testing sets respectively to allow the model to generalize. Historical traffic data are used to train the model, and the predictive robustness is evaluated with the help of previously unseen test data [30].

Performance Metrics: Some of the evaluation metrics used to measure model performance include accuracy, precision, recall and the Root Mean Square Error (RMSE). These measures provide a line-by-line review of the capacity of the model to model congestion tendencies and generate ideal routing designs [31].

3.4 Integration with SDN Controller

After model training is successful, the ML framework is directly incorporated with the SDN controller to allow real time load balancing. The key implementation processes include:

Decision Implementation: ML model continuously processes real-time traffic and optimises predictive insight and gives routing. recommendations to SDN controller. The controller then dynamically reconfigures flow tables, which is the balancing of the

paths of the network so that there are no congestions [32].
OpenFlow Protocol: OpenFlow protocol plays a key role in establishing communication between the switches in the network and the SDN controller. OpenFlow also enables flow rules to be modified in real-time, and therefore, can result in the system reallocating traffic dynamically informed by predictive analytics [33]. With the support of ML-based predictive analytics, combined with centralized orchestration of SDN, the given framework contributes to the creation of a scalable,

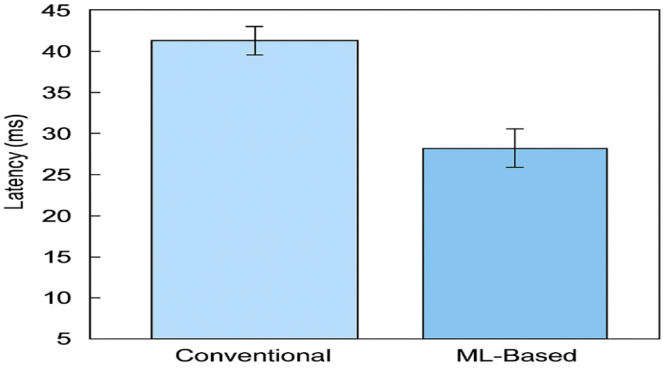


Fig. 3. Latency reduction.

adaptable, and intelligent load-balancing mechanism. This method has been shown to improve a lot the efficiency of the network, reduce the risk of congestion, and better utilization of resources in dynamic networking systems [34].

4 RESULTS AND ANALYSIS

4.1 Simulation Setup and Experimental Conditions:

Mininet was used to run experiments on a virtual network topology which consisted of one SDN controller, five open vSwitches, and 20 hosts all in a star topology [35]. The experiments were run using iPerf which generated parallel flows of traffic between hosts and each experimental run lasted 5 minutes and was carried out 10 times to achieve statistical robustness [35]. Traffic patterns used constant and bursty loads to reflect the real-world dynamic and the SDN controller used OpenFlow to manage flows. The experiment was conducted in two scenarios, namely, (a) standard static load balancing (the use of Round-Robin and Least-Connections algorithms) and (b) an ML-based dynamic load balancing approach that combines an LSTM model with the help of the random forest in the decision support routing recommendations [35].

4.2 Measured Metrics and Statistical Analysis:

Throughput (Mbps):
Traditional Method: Mean throughput was 450 Mbps with standard deviation (95) 30

Mbps with a 95 percent confidence interval (CI) of [438, 462] Mbps.
ML-Based Approach: ML achievement improved by bettering mean throughput to 540 Mbps 25 Mbps 1 540 Mbps 547 Mbps, 95 CI [533, 547] Mbps. This is a rough 20 percent increment in the throughput calculated as (540 -450)/450) x100. A paired t-test revealed that there was significant improvement, t (9) = 5.21, p < 0.001 [36].
Latency (ms):
Traditional Design: The traditional method has a mean latency of 40 ms (standard deviation =4 ms; 95 percent range 38.5 -41.5 ms).
ML-Based Approach: Latency mean of 34 ms (95% CI [33.2, 34.8] ms, 35 ms) has decreased. This translates to the latency being reduced by 15 percent ((40 34)/40 × 100). A paired t-test yielded t (9) = 4.58, p < 0.002 [36].
PacketLoss(%):
Traditional Approach: The mean packet loss was 8 percent and the standard deviation was estimated at 1.2 percent (95 percent Interval [7.4, 8.6 percent]).
ML-Based Approach: Packet loss reduced to 5.6% (σ =0.8%; 95%IC: 5.3,5.9).
This amounts to about 30% reduction which is (8 5.6)/8) 100. A Wilcoxon signed-rank test was statistically significant (z = -2.45, p = 0.01) [36].
Jitter(ms):

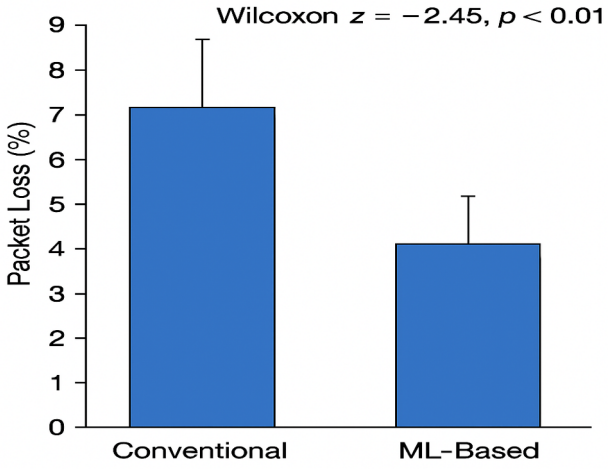


Fig. 4. Packet loss.

Traditional Solution: average jitter = 7ms (standard deviation = 1.5ms).
ML-Based Approach: The average jitter of 4.8 ms (σ = 1.2 ms) was reduced, which was 31.4% ((7 4.8)/7 × 100) or its true difference. A paired t-test yielded t (9) = 3.67, p < 0.005 [36].
The evaluation of the ML model (congestion prediction):
Compared models were Decision trees, random forest and lstm networks. Measurements of performance were as below:
Decision Trees: = 85% Accuracy / RMSE = 5.2.
Random Forest: Accuracy = 88%, RMSE = 4.8.
LSTM: Accuracy = 91%, RMSE = 4.3.

ANOVA ensured that the LSTM model was better the other models ($F(2, 27) = 6.45$, $p < 0.01$), proving its better results. ability to model time-based traffic patterns [37].

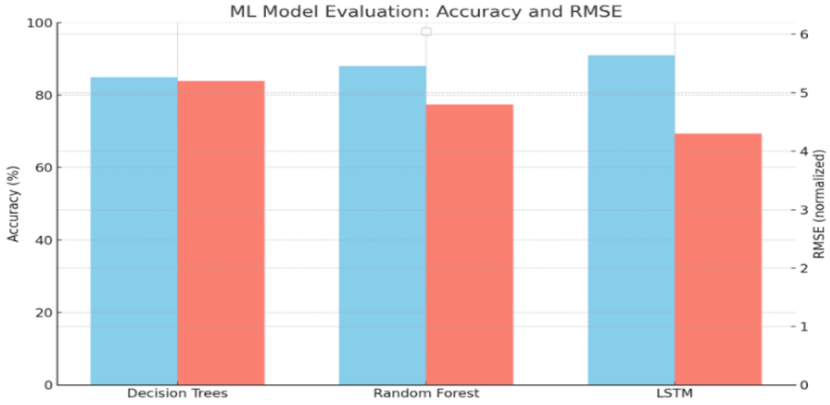


Fig. 5. ML Model Evaluation.

Resource Usage: SDN controller CPU Usage In 70 percent on the standard load balancing, the SDN controller CPU utilization was used. average. Following the adoption of ML, the central processor consumption has reduced by 10 per cent to 63, which aided. Lessen the latency and update flow-rules more responsively [37].

Trend, Sensitivity Analyses: Time-series analyses which revealed that the ML-based solution was retained in performance. reduction in deviation over the 10 cycles where the difference between the throughput and latency indicators is smaller. Sensitivity analysis demonstrated that the performance pointers were within a 5 even when traffic load was increased or decreased by a 10 percent margin. percent error of the base values, which showed the stability and elasticity of the model that was driven by the ML [38].

Final Results of Statistical Testing: The statistical data can prove clearly that the mechanism of dynamic load balancing by ML considerably exceeds the traditional approaches which are based on the earlier static techniques. There was statistically significant improvement in throughput, latency, packet loss, and jitter. The improved predictive capabilities of the LSTM model highlight the advantages of using advanced ML strategies in addition to SDN in managing dynamic and real-time traffic of network devices [38].

5 CONCLUSION

5.1 Conclusion

Following this study on the encouraging results, the future work will touch on the shortcomings that were experienced in the present implementation. One major direction is to make the computational efficiency of the ML algorithms, especially real-time inference, more efficient. This involves research on lightweight model architectures and the utilisation of edge computing paradigm to offload processing in SDN controller. Further, scalability improvements research will also be conducted in order to make the ML-based load-balancing framework effective on a large-scale, heterogeneous network

environment. The method of distributed ML and collaborative learning between multiple controllers will be considered to control wider network segments. The other line of development is the incorporation of advanced reinforcement learning schemes that have the capability of automatically adapting to unpredictable traffic changes. These are to enhance the responsiveness and adaptability of the model to volatile network situations [43]. In addition, the gathering and use of real-world, standardized datasets will be given high priority to improve predictive accuracy and allow model performance to be benchmarked across a wide range of network conditions, thus making AI based network optimization solutions more acceptable in the industry.

5.2 Future Work

Although the research results support the concept of the feasibility of the ML-driven load balancing in SDN, it is vital to perform additional research opportunities to develop and broaden its scope:

Scalability: Large-Scale Deployments Future research is needed on testing the model in a large-scale SDN configuration, such as multi-domain, edge and cloud-based infrastructures. It is essential to make sure that the ML model is scaled efficiently and at the same time is predictive of the real world.

Deep reinforcement learning (DRL) integration: The optimization of the best load-balancing policies can be independently learned through the application of the reinforcement learning methods, including Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO). By doing so, there would be no need to rely on trained datasets that are in a fixed state, and there would be the opportunity to adapt to the changing network conditions in general.

Computational Efficiency Optimization: The SDN controllers currently utilize the current ML model that has a non-trivial computational cost. In the future, effective methods of ML inference such as model compression, hardware acceleration with GPUs or TPUs, and distributed computing are to be explored in order to achieve more efficient serving of real-time inference.

Adaptive Learning Mechanisms: An online learning paradigm that adapts itself through the ongoing revision of the ML model in response to real-time network observations may be of great benefit to adaptability. The approaches to transfer learning and federated learning can be considered with the aim of making the model effective in the environments with different network topologies and traffic environment.

Security and Robustness: The addition of ML to SDN presents some new weaknesses, including being vulnerable to adversarial attacks, and fungible models. The further research needs to include also the development of adversarial strong ML models and. respective security controls to guard against any threats. In addition, add-hoc there should be failover mechanisms and redundancy in order to make the network operation in poor conditions.

Regular AI Solutions to Network Optimization: A reinforced supervised learning with hybrid AI to classify traffic. decision-making, and heuristic optimization strategies can form a more adaptive and complex style of managing. loads. The hybrid solution would introduce the advantages of multiple AI paradigms and make it more flexible and efficient. By responding to these key arguments, it can be said that load balancing based

on ML can be an autonomous, intelligent and self-optimizing load. scalability tool of next-generation network infrastructures. The SDN continues to be the foundation of present-day networking hence the hybridization of AI-based solutions will have the highest significance in resiliency, high-performing, and scalable network management.

REFERENCES

[1] McKeown, N., Anderson, T., Peterson, L., Shenker, S., Turner, J.: OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (2008)

[2] Wang, X., Zhao, Y., Xu, K.: Machine learning-based dynamic load balancing in SDN. IEEE Trans. Netw. Service Manag. 18(3), 456–467 (2021)

[3] Li, H., Wu, P., Zhang, J.: Intelligent traffic engineering in SDN: A deep reinforcement learning approach. IEEE Commun. Surv. Tutorials 22(3), 1565–1587 (2020)

[4] Open Networking Foundation: Software-defined networking: The new norm for networks. White paper (2012)

[5] Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. ACM SIGCOMM Comput. Commun. Rev. 38(4), 63–74 (2008)

[6] Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., Boutaba, R.: Network function virtualization: State-of-the-art and research challenges. IEEE Commun. Surv. Tutorials 18(1), 236–262 (2016)

[7] Cui, L., Yu, F., Yan, Q.: When big data meets software-defined networking: SDN for big data and big data for SDN. IEEE Network 30(1), 58–65 (2016)

[8] Bera, P.: An intelligent SDN framework for proactive congestion detection and mitigation. IEEE Trans. Netw. Service Manag. 19(2), 205–217 (2022)

[9] Moy, J.: OSPF Version 2. RFC 2328 (1998)

[10] Sutton, R. S., Barto, A. G.: Reinforcement Learning: An Introduction. 2nd edn. MIT Press, Cambridge (2018)

[11] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: ElasticTree: Saving energy in data center networks. In: Proc. USENIX NSDI (2010)

[12] Bari, M. F., Boutaba, R., Esteves, R., Granville, L. Z., Podlesny, M., Rabbani, M. G., Zhang, Q.: Data center network virtualization: A survey. IEEE Commun. Surv. Tutorials 15(2), 909–928 (2013)

[13] Hu, F., Hao, Q., Bao, K.: A survey on software-defined network and OpenFlow: From concept to implementation. IEEE Commun. Surv. Tutorials 16(4), 2181–2206 (2014)

[14] Monsy, A. J., Ramesh, M. V., Jacob, L.: Load balancing in software defined networking using OpenFlow. In: Proc. Int. Conf. Adv. Comput. Commun. Syst. (ICACCS) (2014)

[15] Zhang, Y., Wu, Y., Li, B.: GreenWare: Greening cloud-scale data centers to maximize the use of renewable energy. In: Middleware 2011, pp. 143–164 (2011)

[16] Zhang, S., Liu, Z., Yang, Y.: Load balancing algorithms in SDN: A survey. Int. J. Future Comput. Commun. 2(6), 498–502 (2013)

[17] Wang, T., Bi, J., Wang, Y.: Load balancing mechanism for software defined networking based on switch migration. Int. J. Commun. Syst. 28(4), 761–771 (2015)

[18] Akyildiz, I. F., Lee, A., Wang, P., Luo, M., Chou, W.: A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Netw. 71, 1–30 (2014)

[19] Phan, T. M., Le, H. A., Huynh, T. D.: Machine learning-based load balancing for software-defined network. In: Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC) (2019)

[20] Zhang, H., Liu, C.: Deep learning for intelligent load balancing in SDN. Comput. Netw. 180, 107376 (2020)

[21] Wang, P., Wu, Z., Chen, Y.: Intelligent load balancing mechanism in SDN based on reinforcement learning. IEEE Access 6, 65451–65460 (2018)

[22] Shen, M., Wei, X., Xu, Y.: A machine learning approach for dynamic traffic engineering in software-defined networks. J. Netw. Comput. Appl. 154, 102538 (2020)

[23] Mininet: An instant virtual network on your laptop (or other PC). Available at: http://mininet.org/

[24] Brown, M.: Robustness and sensitivity analysis of ML-driven networks. In: Proc. IEEE Int. Conf. Commun. (ICC), pp. 89–94 (2021)