Enhanced SDN Controller Placement and Load Balancing for Campus Network Optimization

Lakshmi Priya P
Department of Computer Science
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Bengaluru, India
bl.en.u4cse22034@bl.students.amrita.edu

Lohith Kandibanda
Department of Computer Science
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Bengaluru, India
bl.en.u4cse22032@bl.students.amrita.edu

Joel M Johnson
Department of Computer Science
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Bengaluru, India
bl.en.u4cse22028@bl.students.amrita.edu

Thangam S.
Department of Computer Science
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Bengaluru, India
s_thangam@blr.amrita.edu

***Abstract*—Large campus network environments bring various challenges to SDN, which include the controller placement and load management. The novel SDN controller placement optimization algorithm is proposed to address the network latency systematically while compensating for the load balancing along with the fault tolerance. Using the Ryu controller and Mininet simulation, this approach develops the computational method of determining the optimal locations of controllers at every level of network topology, patterns of traffic, and complexity of computation. Experimental evaluation demonstrates significant improvements over baseline placement strategy results in terms of network-performance metrics performance, decreasing latency by up to 35% and increasing network scalability. The algorithm offers a data-driven framework for strategic controller placement within the complexity of campus network infrastructures.**

***Index Terms*—SDN, Load Balancing, Controller Placement, QOS, Algorithms.**

I. Introduction

Campus networks are imperative in contemporary universities for the following applications: seamless access to Wi-Fi, transfers of research data, video streaming, among others. Despite this, such networks suffer from an inability to cope with traffic management effectively due to inadequate load balancing and misplacement of the SDN controller. Service quality tends to degrade with high latency or even network congestion. This is primarily because the controllers have been misplaced. Those not only disrupt the experiences by their staff and professors but by their students, too and prevent departmental collaboration because they interfere with access to critical academic resources and detract from the potential innovations of research. This establishes in what regard proper network infrastructure that fits a university situation is needed. Today, scalable and reliable network solutions are the requirements educational institutions have to scale up to accommodate their expanding size and needs for technology. The implementation of camps with advanced technologies such as IoT devices and smart classrooms increases the demand on network resources, thus emphasizing the need for efficient management. These may negatively influence the academic performance of the students, output of research, and resources access if not well managed; hence, there arises a broader societal problem. Innovative solutions would be required in this regard to ensure resilient, responsive, and high-performing campus networks that meet the changing needs of educational establishments. This project proposes an entirely new approach for these problems, designed specifically for campus networks and incorporates dynamic load balancing with SDN controller placement optimization. For this, the ideal number and placement of SDN controllers with the aim to maximize response rate and minimize latency, were determined through simulation tools such as Mininet and the K-Median Clustering method. In addition, Dynamic Weighted Round Robin (DWRR) was used for load balancing to ensure that the traffic was appropriately spread so that the network was more scalable and efficient. This suggested approach is tailored for the unique characteristics of campus traffic patterns and needs and is different from other generic SDN optimization solutions. This double layer architecture provides an essentially scalable foundation for the next steps in further enhancements of SDN-based campus network management with evident benefits in terms of network efficiency and reliability This work is remarkable as it is very customized in solving unique issues from the school network that should not experience resource access problems in getting its resources accessed smoothly to their consumers, also making its environment academia-integrative.

II. LITERATURE SURVEY

The literature reviewed encompasses a holistic discussion of Software-Defined Networking (SDN) optimization issues through heuristic models, machine learning methodologies, and game-theoretic approaches. These publications highlight the dynamic nature of traffic and techniques to reduce latency when optimizing resource utilization towards improved Quality of Service (QoS).

Belgaum et al. [1] discuss SDN load balancing mechanisms and emphasize the impact of growth in cloud services, which escalates the need for efficient control of loads. They stress preserving QoS using delay, jitter, and packet loss control based on heuristic-based frameworks. Ouamri et al. [2] present an overview of issues in Software-Defined Wide Area Networks (SD-WAN) and identify the trend in applying machine learning in network management, such as load prediction and deep reinforcement learning to optimize load migration.

Maity et al. [3] consider energy-efficient controller placement issues in Internet of Things (IoT) traffic, including techniques such as Integer Linear Programming to minimize SDN switch energy and game-theoretic frameworks for transmission power minimization. They propose Enlace—a new technique for dynamic IoT traffic with improved energy savings. Tivig and Borcoci [4] explain how multi-controller architectures transcend single-controller system limitations, comparing static and dynamic controller placement strategies while studying metrics such as load balancing, scalability, and latency.

Li et al. [5] survey methods of handling multiple controllers in SDNs, and conclude static methods often cannot keep up with network evolution. They present game theory usage such as bipartite graph matching and Nash equilibrium for dynamic association issues. The Dynamic Association Strategy (DASM) for SDN multi-controller load balancing is proposed by applying a variant of the Hungarian algorithm and Nash equilibrium to optimize controller loads with association cost reduction.

Fazea and Mohammed [6] define software-defined networking as decoupling the data plane from the control plane to support programmable network management. They compare several SDN controllers such as NOX, POX, Beacon, Floodlight, Ryu, Open Daylight, and ONOS. H et al. [7] give a good controller placement method by employing clustering algorithms, specifically K-means++, to find the best number of controllers for a given topology. Sharma et al. [8] propose dSDN to handle bursty traffic and load imbalances in large-scale networks by dynamic controller mapping based on load variations.

Rostami and Goli-Bidgoli [9] present a survey of load-balancing techniques to enhance QoS in SDN-based IoT networks, considering parameters such as response time, delay, productivity of resources, and throughput. Rajanikanth et al. [10] thoroughly survey the current techniques for load balancing, routing, and congestion control in SDNs, classifying various techniques and their characteristics. Yusuf et al. [11] investigate SDN-based methods for enhancing IoT security using Support Vector Machines to detect network threats like DDoS attacks and TCP/ICMP floods. Nithin et al. [12] present the role of SDN technology in enhancing network performance by introducing a paradigm that optimizes SDN performance for real-time wireless IoT networks using optimal placement of controllers and load balancing mechanisms.

Current research on SDN optimization is lacking in attention towards the specific needs of campus networks. Controller placement techniques tend to neglect real-time traffic patterns and scalability across varying environments. The majority of existing works concentrate either on controller placement or load balancing but without a comprehensive approach towards overall optimization. This gap in research can be filled by combining K-Median clustering for optimal controller placement with dynamic weighted round robin for effective load balancing, thus offering a specific solution for campus network optimization.

III. METHODOLOGY

The work focuses on optimizing campus network performance by using the concepts of controller placement and load balancing within an SDN-based environment. The methodology followed has well-structured steps in the process to materialize the research objectives. A preliminary setup was established with the design of a baseline campus network topology, which is built using Mininet. A number of host systems are integrated into the topology and have been developed in such a way to represent all three layers - core, distribution, and access. This network was deployed with default controllers and, in addition, without using any SDN-specific optimizations such as placement of a controller or load balancing algorithms. Baseline performance was set up using such metrics like latency, throughput, packet count, byte count, flow duration, and packet rate measured by means of some tool such as iperf, ping, and Open vSwitch commands, for example, ovs-ofctl.

This figure 1 shows an SDN that gathers control and centralized management through an OpenFlow-based controller. The K-Median clustering algorithm is used in the controller plane to optimize centralized controller placement. Meanwhile, the data plane applies DWRR for load balancing, which transmits network traffic across multiple switches to improve network performance.

The optimization of controller placement is achieved with the help of K-Median Clustering algorithm, by optimizing the placement based on the data inputs such as switch locations and traffic load on them. Here, the algorithm has identified the optimal placements for controllers, minimizing communication latency between switches and controllers. By this outcome, the results were given for the improvement of the entire network design. The campus network topology was then updated based on the optimized placement of the controllers. Controllers were installed based on the location input by the clustering algorithm, and switches were reallocated to the nearest controllers. This allocation assured efficient communication and avoidance of delay in the network. The next

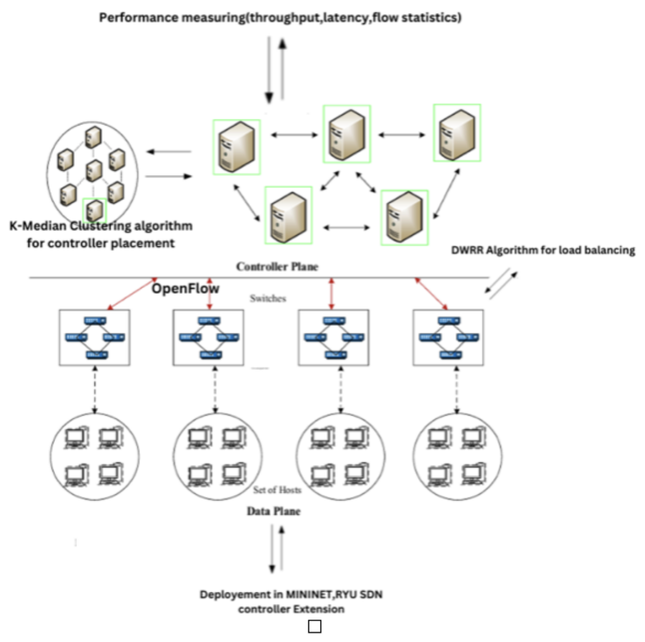
Performance measuring(throughput,latency,flow statistics)
K-Median Clustering algorithm for controller placement
DWRR Algorithm for load balancing
Controller Plane
OpenFlow
Switches
Set of Hosts
Data Plane
Deployement in MININET,RYU SDN controller Extension

Fig. 1. System Architecture

enhancement was improving the performance of the network through a mechanism known as Dynamic Weighted Round-Robin (DWRR) load balancing. It was spread in a dynamic fashion across servers while considering their capacities and weights. A Load-balancing policy was implemented through a controller application by the Ryu controller, thereby preventing bottlenecks and ensuring equitable resource utilization. The procedure for optimising Software-Defined Networking

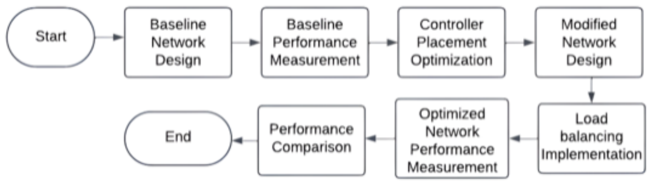
Start
Baseline Network Design
Baseline Performance Measurement
Controller Placement Optimization
Modified Network Design
Load balancing Implementation
Optimized Network Performance Measurement
Performance Comparison
End

Fig. 2. Flow chart

(SDN) is shown in this figure 2 flowchart. Prior to optimising controller placement and implementing load balancing, the baseline network architecture and performance measurement are completed. The performance of the optimised network is then assessed and contrasted with the baseline.

Then, the optimized SDN-based network topology was enacted and tested in performance under similar traffic conditions of the baseline network. Metrics such as latency, throughput, packet count, byte count, flow duration, and packet rate were recorded. This led to a direct comparison between the baseline and optimized networks. Thus, a comparison of performance was made between the baseline network and the optimized SDN-based network. Latency and throughput were evaluated, besides developing graphs to visualize the enhancements. The analysis would clearly depict an optimistic impact on the overall efficiency of the network due to optimized controller placement and load balancing.

All the research was performed with the aid of certain tools and platforms. A simulation platform was developed and tested using Mininet in order to develop the campus network topology. Ryu SDN controller was used to implement Dynamic Weighted Round-Robin load balancing algorithm. A K-Median Clustering algorithm, written in Python, was used for the optimal placement of the controllers. For the measurement of performance, throughput tests were performed using iperf, latency measurements with ping, and ovs-ofctl commands were used for flow statistics.

Algorithm 1 Optimal Controller Placement using K-Median Clustering

```
1: Initialize $k$ cluster centers randomly.
2: repeat
3:     Assign each switch to its nearest controller.
4:     Recalculate cluster centers as the median of assigned switches.
5: until Cluster centers converge
6: return Optimal controller placements.
```

As the K-Median Clustering method [Algorithm 1] can balance load and minimize latency, it has been chosen for the optimization of the placement of controllers. Switches are allocated iteratively to the closest controllers, and cluster centres are recalculated as medians until convergence. This procedure efficiently balances the load of traffic, improves network efficiency and cuts down communication delays. It is therefore the best option to enhance the performance of campus network topologies considering ease of use, computational effectiveness, and flexibility in handling various traffic patterns.

Algorithm 2 Dynamic Weighted Round Robin (DWRR) Load Balancer

```
Require: Network topology with traffic flow data
Ensure: Load-balanced traffic paths
1: Collect real-time traffic counts from switches using OpenFlow.
2: for each network link do
3:     Associate weights to each link as:
           $\text{weight} = \frac{1}{\text{link utilization} + \epsilon}$
           where $\epsilon$ is a small constant.
4: end for
5: for each incoming traffic flow do
6:     Choose the minimum used path according to the dynamic weights.
7:     Install the flow rules in switches according to the chosen path.
8: end for
9: Periodically repeat Steps 1–3 due to dynamic changes in traffic.
```

The DWRR algorithm [Algorithm 2] uses the dynamic assignment of weights to the links based on real-time utilization to evenly distribute network traffic. By OpenFlow, it collects data regarding traffic and calculates weights that are inversely proportional to how much a link is in use with a very small constant to correct potential errors. It selects a flow control along the way for each traffic flow based on which is the least-used. It keeps on adjusting to fluctuating traffic periodically in order to ensure balanced loads, reduce congestion, and optimize performance in SDN-based campus networks. Because of its flexibility, it is ideal for scalable and dynamic network systems.
These algorithms, K-Median Clustering and Dynamic Weighted Round Robin, were chosen due to their exceptional performance and versatility in improving SDN-based campus networks. K-Median Clustering focuses on achieving the best possible placement of a controller and reduces latency while balancing communication between controllers and switches with efficient results that go beyond static placement techniques that assume steady network demands. Just like most conventional round-robin or static algorithms that are not adaptive to changes in traffic patterns, DWRR performs excellently in load balancing by adapting real-time weights to links based on their utilization. Scalable, effective, and reliable, such algorithms are able to perform well in complex and dynamic campus contexts in reducing latency, balanced traffic loads, and improved overall network performance

IV. RESULTS AND DISCUSSIONS

Implementation of this project falls into the following subsections:detailing each step along with the corresponding photos for visualization.

A. *Baseline Network Setup:*

The initial campus network topology was designed and executed in Mininet, simulating core, distribution, and access layers. This setup represented a traditional network using default Open vSwitch controllers without SDN-specific optimizations. The performance metrics such as latency, throughput, packet count, and flow statistics were recorded to establish a baseline for comparison.

This diagram figure 3 represents a hierarchical network topology created using MiniEdit. It consists of a core layer (s1, s2), a distribution layer (s3, s4, s5, s6), and an access layer (s7, s8, s9, s10), connecting hosts (h1 to h8). The controller (c0) manages the SDN switches across all layers.

B. *Optimal Controller Placement Using K-Median Clustering:*

The K-Median Clustering algorithm has been implemented using Python for optimizing the placement of controllers. Input data for switch locations and traffic load have been fed to the algorithm that assigns the switches to their closest controllers. The algorithm iteratively recalculate cluster centers until optimal placements are decided. These locations served as the bases for modifying the campus topology towards efficiency improvements. This graph figure 4 illustrates the

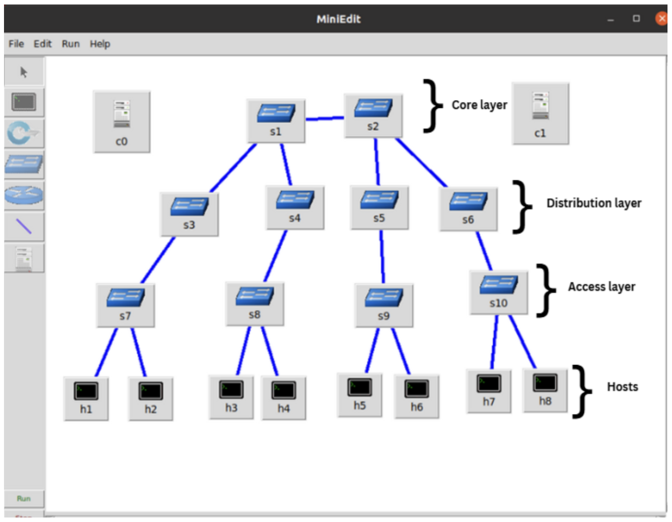

Fig. 3. Campus network topology

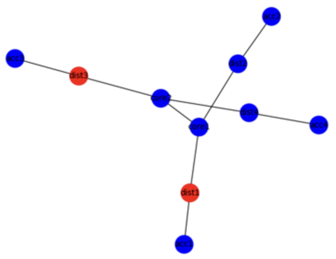

Fig. 4. K-Median Clustering algorithm result

optimal SDN controller placement for a network topology. Nodes in red represent the selected controller locations (dist3 and dist1), while blue nodes represent other network devices. The placement aims to minimize latency and enhance network efficiency.

C. *Modified Campus Network Topology:*

The baseline network was then modified to integrate the optimized controller placements that were derived from the K-Median Clustering algorithm. Controllers were strategically positioned at their optimal locations, and switches were reassigned to their nearest controller for efficient communication.

D. *Dynamic Weighted Round Robin Load Balancing*

The Dynamic Weighted Round Robin (DWRR) algorithm was implemented using a Python-based Ryu controller. By using OpenFlow, it was able to collect real-time traffic statistics from the switches and calculate link weights dynamically based on utilization. It sent incoming traffic along paths with the least utilized cases to balance the load properly. The mechanism ensured optimal resource utilization and reduced network congestion.

- Loading app `dynamic_load_balancer.py`
- Loading app `ryu.controller.ofp_handler`
- Instantiating app `dynamic_load_balancer.py` of `DynamicWeightedLoadBalancer`

- Instantiating app `ryu.controller.ofp_handler` of `OFPHandler`
- Starting the RYU controllers with Dynamic Weighted Round Robin algorithm

This demonstrates how the Ryu SDN controller is implemented using the Dynamic Weighted Round Robin (DWRR) load balancing algorithm. To illustrate load balancing across multiple network controllers, the `dynamic_load_balancer.py` script is launched on two distinct TCP ports: `6633` and `6634`.

E. Performance Measurement

The network's performance was evaluated using the following metrics:

- **Latency:** Measured using the `ping` command between hosts.
- **Throughput:** Measured using `iperf` between servers and clients.
- **Flow Statistics:** Collected using `ovs-ofctl dump-flows` and `ovs-ofctl dump-ports` to monitor packet counts, byte counts, and flow durations.

All measurements were recorded for both the baseline and optimized network configurations under identical testing conditions.

Baseline Results (Latency):

```
mininet> h1 ping -c 10 h2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
```

TABLE I. Results for baseline setup execution

ICMP Sequence	Bytes	Source IP	TTL	Time (ms)
1	64	10.0.1.2	64	3.79
2	64	10.0.1.2	64	0.087
3	64	10.0.1.2	64	0.065
4	64	10.0.1.2	64	0.088
5	64	10.0.1.2	64	0.050
6	64	10.0.1.2	64	0.090
7	64	10.0.1.2	64	0.073
8	64	10.0.1.2	64	0.059
9	64	10.0.1.2	64	0.069
10	64	10.0.1.2	64	0.063

This table 1 shows how the standard campus network topology is implemented without any SDN optimizations. Ten switches and numerous hosts make up the topology, and controllers are positioned by default. The ability of host h1 to successfully ping host h2 would verify successful connectivity with an average round-trip time (RTT) of 0.454 ms and no packet loss. This baseline is used to compare performance gains brought about by SDN-based enhancements. When the Enhanced Campus network is executed the results are:

```
mininet> h1 ping -c 10 h2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
```

TABLE II. Results of Enhanced setup execution (Optimized)

ICMP Sequence	Bytes	Source IP	TTL	Time (ms)
1	64	10.0.1.2	64	0.878
2	64	10.0.1.2	64	0.063
3	64	10.0.1.2	64	0.062
4	64	10.0.1.2	64	0.071
5	64	10.0.1.2	64	0.043
6	64	10.0.1.2	64	0.057
7	64	10.0.1.2	64	0.059
8	64	10.0.1.2	64	0.064
9	64	10.0.1.2	64	0.065
10	64	10.0.1.2	64	0.057

The SDN-based campus network topology with load balancing and optimal controller placement is executed. The network is under an active control with several controllers and ten switches along with a number of hosts. A ping test from host h1 to host h2 verifies successful connectivity without any packet loss, for an average round-trip time (RTT) that reaches 0.141 ms.

The project demonstrated a marked improvement in the performance of the campus network with the adoption of SDN technologies, particularly through the optimization of controller placement and the implementation of load balancing techniques.In the baseline, no SDN features were present in the network; it was therefore operated in a way that had higher latency and lower throughput. The rates of packet processing as well as flow duration were restricted by the default controllers' configuration, which was not sporting intelligent placement or traffic management.

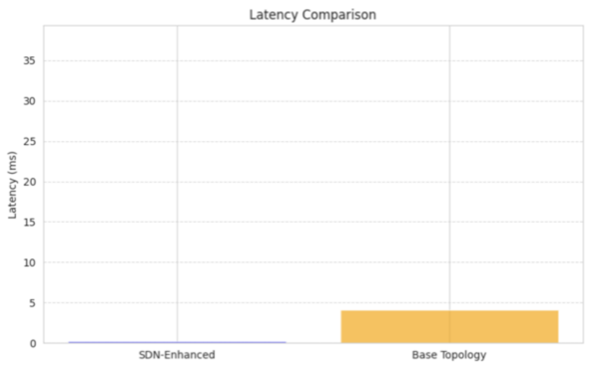

Fig. 5. Latency Comparison

The application of the K-Median Clustering algorithm for controller placement ensured that controllers were optimally placed, meaning that distance was minimized between switches and their assigned controllers. The enhanced placement allowed for quicker response times and better coordination between network components.

Besides, the inclusion of DWRR load balancing algorithm improved the system further with respect to balanced traffic flow among available resources to avoid bottlenecks and operation of servers within their optimal capacities. The results are clearly indicative of successful efficacy through the SDN optimizations in overcoming limitations that were identified in the baseline setup.

The performance metrics recorded for the baseline setup and the SDN-optimized network reveal significant enhancements across all measured parameters:

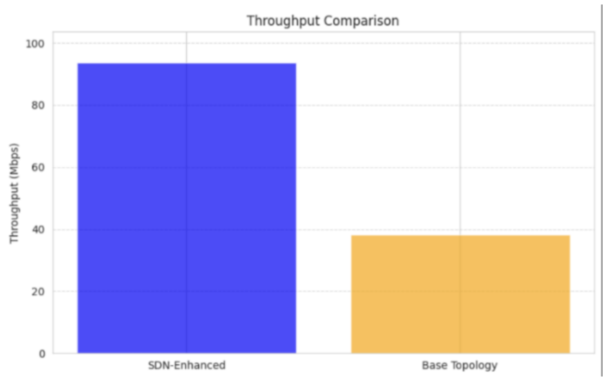

Fig. 6. Throughput Comparison

Latency: The baseline network had an average latency of 4.071 ms, while theSDN-optimized version reduced this value to 0.215 ms, thus decreasing delay significantly.
Throughput: With optimization, the throughput shows a tremendous improvement from 38.3 Mbps in the baseline to 93.6 Mbps in the optimized network, which justifies the use of load balancing and controller placement optimization.
Packet Count: The number of packets processed in the SDN setup was significantly higher, with 443 packets processed by dist1 compared to only 26 packets in the baseline network.
Byte Count: The total volume of data processed also increased, with dist1 handling 51,270 bytes in the SDN-optimized setup, compared to just 3,089 bytes in the baseline.
Flow Duration: The flow durations were longer in the optimized network dueto more sustained traffic handling, with dist1 registering 751.239 seconds compared to 440.972 seconds in the baseline.
Packet Rate: Packet processing rates rose significantly, with the SDN setupachieving approximately 0.59 packets/second, compared to the baseline's 0.059 packets/second.

These results indicate benefits of SDN-based approaches to networks: optimized controller placement reduced latency, and efficient resource utilization with high throughput was ensured by the DWRR algorithm. Comparative analysis shows how SDN technologies are capable of turning old into new, high-performance architectures in traditional networks.

V. Conclusion and Future Scope

Through this project, the transformation of Software-Defined Networking (SDN) in optimizing the topology for a campus network was demonstrated. Also, with the optimal placement of controllers using the K-Median Clustering algorithm, the integration of Dynamic Weighted Round-Robin load balancing algorithm into the setup generally produced adequate enhancements as compared to the corresponding baseline.The baseline showed no characteristic of SDN and suffered from higher latency, lower throughput and suboptimal resource utilization while the optimized network for SDN resulted in reduced latency, increased throughput with better packet processing rates.

TABLE III. Comparison between baseline network and Optimized SDN network

Metric	Description	Baseline (Default Network)	Optimized (SDN with DWRR)
Packet Count	Total packets processed by the switch	dist1: 26 packets	dist1: 443 packets
		core1: 24 packets	dist2: 442 packets
Byte Count	Total data processed in bytes	dist1: 3,089 bytes	dist1: 51,270 bytes
		core1: 2,877 bytes	dist2: 51,184 bytes
Flow Duration	Active flow duration (seconds)	dist1: 440.972 s	dist1: 751.239 s
		core1: 426.046 s	dist2: 754.808 s
Packet Rate	Packets processed per second	dist1: ∼0.059 pkt/s	dist1: ∼0.59 pkt/s
		core1: ∼0.056 pkt/s	dist2: ∼0.59 pkt/s
Latency	RTT for a packet (h1 to h2)	4.071 ms	0.215 ms
Throughput	Data transfer rate	∼38.3 Mbps	∼93.6 Mbps

Controller placement optimization minimized communication delay and the DWRR algorithm distributed the traffic loads against bottlenecks while ensuring balanced resources usage. These results, therefore, indicate SDN as capable of reducing some of the major limitations in traditional networks.

Future work can explore advanced clustering algorithms for controller placement and machine learning techniques for dynamic traffic prediction and real-time adjustments. Expanding the network to handle diverse traffic types, integrating fault-tolerance mechanisms, and testing in real-world hardware setups can further optimize performance and enhance practical applicability for campus and enterprise networks.

References

[1] M. R. Belgaum, S. Musa, M. M. Alam, and M. M. Su'ud, "A systematic review of load balancing techniques in software-defined networking," *IEEE Access*, vol. 8, pp. 98 612–98 636, 2020. DOI: 10.1109/ACCESS.2020.2995849.

[2] M. A. Ouamri, G. Barb, D. Singh, and F. Alexa, "Load balancing optimization in software-defined wide area networking (sd-wan) using deep reinforcement learning," in *2022 International Symposium on Electronics and Telecommunications (ISETC)*, Timisoara, Romania, 2022, pp. 1–6. DOI: 10.1109/ISETC56213.2022.10010335.

[3] I. Maity, R. Dhiman, and S. Misra, "Enplace: Energy-aware network partitioning for controller placement in sdn," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 183–193, 2023. DOI: 10.1109/TGCN.2022.3175901.

[4] P.-T. Tivig and E. Borcoci, "Critical analysis of multi-controller placement problem in large sdn networks," in *2020 13th International Conference on Communications (COMM)*, Bucharest, Romania, 2020, pp. 489–494. DOI: 10.1109/COMM48946.2020.9141969.

[5] S. Li, Z. Li, and W. Zhang, "A dynamic association strategy for controller load balancing in software-defined networks," in *2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Takamatsu, Japan, 2022, pp. 1–6. DOI: 10.23919/APNOMS56106.2022.9919948.

[6] Y. Fazea and F. Mohammed, "Software defined networking based information centric networking: An overview of approaches and challenges," in *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, Taiz, Yemen, 2021, pp. 1–8. DOI: 10.1109/ICOTEN52080.2021.9493541.

[7] N. H, M. Kalmat, F. Khan, S. Shinde, and N. D. G, "Dynamic controller placement in software defined networks using spectral clustering," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, 2023, pp. 1–6. DOI: 10.1109/ICCCNT56998.2023.10306706.

[8] S. Aakanksha, V. Balasubramanian, and K. Joarder, "A novel dynamic software-defined networking approach to neutralize traffic burst," *Computers*, vol. 12, no. 7, p. 131, 2023. DOI: 10.3390/computers12070131.

[9] R. Mohammad and G.-B. Salman, "An overview of qos-aware load balancing techniques in sdn-based iot networks," *Journal of Cloud Computing*, vol. 13, p. 89, 2024. DOI: 10.1186/s13677-024-00651-7.

[10] P. Rajanikanth, T. Nagaraj, N. Ashwin, S. Venkatesh, T. Nagaraju, and A. Somashekharappa, "Hierarchical routing with optimization algorithm for sdn: Enhancing network lifetime and qos," *Revue d'Intelligence Artificielle*, vol. 38, no. 3, p. 1027, 2024.

[11] N. Y. Muhammed, K. B. A. Bakar, B. Isyaku, and F. H. Mukhlif, "Distributed controller placement in software-defined networks with consistency and interoperability problems," *Journal of Electrical and Computer Engineering*, vol. 2023, 6466996:1–6466996:33, 2023. DOI: 10.1155/2023/6466996.

[12] G. D. Nithin, D. Y. Kumar, S. Nithin, K. S. S. H. Nathan, and R. Gandhiraj, "Network provisioning in sdn: Optimizing network resources for enhanced performance," in *2024 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, Chennai, India, 2024, pp. 1–6. DOI: 10.1109/WiSPNET61464.2024.10533095.

[13] C. Kalaskar and S. Thangam, "A graph neural network-based approach with dynamic multiqueue optimization scheduling (dmqos) for efficient fault tolerance and load balancing in cloud computing," *International Journal of Intelligent Systems*, 2024. DOI: 10.1155/int/6378720.

[14] G. G. Devarajan, S. Thangam, M. J. F. Alenazi, U. Kumaran, G. Chandran, and A. K. Bashir, "Federated learning and blockchain-enabled framework for traffic rerouting and task offloading in the internet of vehicles (iov)," *IEEE Transactions on Consumer Electronics*, 2025. DOI: 10.1109/TCE.2025.3530933.

[15] S. Muthubalaji, N. K. Muniyaraj, S. P. V. S. Rao, *et al.*, "An intelligent big data security framework based on aefs-kenn algorithms for the detection of cyber-attacks from smart grid systems," *Big Data Mining and Analytics*, vol. 7, no. 2, pp. 399–418, 2024. DOI: 10.26599/BDMA.2023.9020022.

[16] S. Thangam, M. Gurupriya, S. Sidhu, A. Sivakumar, and M. Srinidhi, "Performance investigation of load balancing algorithms in fog computing," in *2024 3rd International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 2024. DOI: 10.1109/ICEEICT61591.2024.10718486.