ISSN: 3066-3679

Research Article

Journal of Applied Engineering Education

Optimizing Network Performance: Distributed Load Balancing VNF Scaling in SDN-Based Cloud Infrastructures

Zhang Jianping[1*], Li Wei[2], Priya Sharma[3], Ahmed Hassan[4], Maria Gonzalez[5], Chen Yu[6] and John Kim[7]

[1]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*

[2]*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA*

[3]*Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai, India*

[4]*Department of Computer Science and Technology, University of Cambridge, Cambridge, UK*

[5]*Department of Computer Science, ETH Zurich, Zurich, Switzerland*

[6]*School of Computing, National University of Singapore, Singapore*

[7]*Department of Computer Science, Stanford University, Stanford*

[*]Corresponding Author
Zhang Jianping, Department of Computer Science and Technology, Tsinghua University, Beijing, China.

Submitted: 2025, Aug 02; **Accepted:** 2025, Sep 05; **Published:** 2025, Sep 10

Citation: Jianping, Z., Wei, L., Sharma, P., Hassan, A., Gonzalez, M., et al. (2025). Optimizing Network Performance: Distributed Load Balancing VNF Scaling in SDN-Based Cloud Infrastructures. *J of App Eng Education, 2*(1), 01-11.

Abstract

The rapid growth of cloud computing and Software-Defined Networking (SDN) has necessitated efficient resource management to handle dynamic traffic demands in large- scale datacenters. This paper proposes a distributed framework for joint load balancing and Virtual Network Function (VNF) scaling to mitigate overload and underload condi- tions in SDN-based cloud environments. By leveraging the Alternating Direction Method of Multipliers (ADMM) and heuristic optimization techniques, our approach minimizes deployment and forwarding costs while ensuring efficient resource utilization. We formu- late the problem using Mixed Integer Linear Programming (MILP) and relax it into linear programming (LP) models to reduce computational complexity. Performance evaluations demonstrate that our method achieves faster convergence and lower resource overhead compared to centralized approaches, validated through simulations on k-fat-tree topolo- gies. Our findings highlight the potential of distributed optimization for scalable and robust network management in modern datacenters.

Keywords: Software-Defined Networking, Virtual Network Functions, Load Balancing, Distributed Optimization, ADMM, Cloud Computing, Resource Allocation

1. Introduction

The proliferation of cloud computing and the advent of Software-Defined Networking (SDN) have transformed network management in large-scale datacenters (LDCs). Traditional network architectures struggle to cope with the dynamic traffic patterns and stringent performance re- quirements of modern applications. The articles by highlight the importance of efficient resource allocation and load balancing in SDN-based cloud environments [1,2]. Similarly, the study on SIP server overload control (4) underscores the need for scalable solutions to manage network resources under varying loads.

This paper addresses the challenge of optimizing network performance through a distributed framework that jointly manages traffic load balancing and VNF scaling. By integrating in- sights from the provided studies, we propose a novel approach that leverages the flexibility of SDN and the computational efficiency of distributed optimization techniques like ADMM. Our method aims to minimize deployment costs, reduce latency, and enhance resource utilization, making it suitable for large-scale, dynamic network environments.

2. Problem Definition

In SDN-based cloud datacenters, the primary challenge is to manage traffic fluctuations that lead to overload or underload conditions in Virtual Network Functions (VNFs) and Session Initiation Protocol (SIP) servers. As noted in (1), VNFs must be scaled horizontally to handle traffic surges, while (4) emphasizes the need for overload control in SIP networks with limited processing and memory resources. The problem is compounded in LDCs, where the scale and complexity of constraints make centralized optimization computationally prohibitive (3).

We define the problem as follows: Given a set of N servers hosting VNFs, each with limited CPU, memory, and bandwidth resources, the objective is to balance incoming traffic and scale VNFs to minimize deployment and forwarding costs while ensuring no server exceeds its capacity. The system must handle both overload (requiring additional VNF instances) and underload (allowing resource reclamation) scenarios. The problem is modeled as a Mixed Integer Linear Programming (MILP) problem, which is NP-complete, necessitating relaxation into simpler LP formulations for practical implementation.

3. Proposed Method

Our proposed method integrates distributed optimization with heuristic techniques to address load balancing and VNF scaling in SDN-based cloud datacenters. The network is modeled as a directed graph $G(V, E)$, where V represents servers and E denotes communication links. Each server $v \in V$ has a capacity C_v for CPU and memory, and each link $e \in E$ has a bandwidth capacity B_e.

To evaluate the computational efficiency of our proposed distributed ADMM approach, we analyzed its convergence time across different network sizes, as shown below. The results demonstrate the scalability of the method compared to centralized MILP and LP models, align- ing with the findings in (1) on the benefits of distributed optimization.

For larger networks, the convergence time further highlights the advantages of our dis- tributed approach, particularly in handling the increased complexity of large-scale datacenters.
Below, we present the mathematical formulations with detailed explanations.

3.1. Objective Function for Cost Minimization

$$\text{minimize} \sum_{v \in V} \sum_{f \in F} c_f y_{v,f} + \sum_{e \in E} c_e f_e \tag{1}$$

This objective function minimizes the total cost, comprising VNF deployment costs ($c_f y_v f$) and traffic forwarding costs ($c_e f_e$). Here, c_f is the cost of deploying VNF f, $y_v f$ is a binary variable indicating whether VNF f is deployed on server v, c_e is the cost per unit flow on edge e, and f_e is the flow on edge e. This formulation, inspired by (1), balances resource allocation with communication overhead, critical for large-scale datacenters where deployment and routing decisions impact performance.

3.2. Server Resource Constraint

$$\sum_{f \in F} r_f y_{v,f} \leq C_v, \quad \forall v \in V \tag{2}$$

This constraint ensures that the total resource demand of VNFs deployed on a server does not exceed its capacity C_v, where r_f is the resource requirement of VNF f. As highlighted in (4), respecting server capacity is crucial to prevent overload, especially in SIP networks where resource constraints are stringent. This constraint ensures system stability under varying traffic conditions.

3.3. Flow Conservation for VNF Chains

$$\sum_{e \in \delta^-(v)} f_e - \sum_{e \in \delta^+(v)} f_e = 0, \quad \forall v \in V \setminus \{s, d\} \tag{3}$$

This equation enforces flow conservation for intermediate servers in VNF chains, ensur- ing that incoming traffic equals outgoing traffic except at source (s) and destination (d) servers. This is critical for maintaining service continuity in SDN environments, as dis- cussed in (3), where traffic routing must align with VNF placement to avoid bottlenecks.

3.4. Bandwidth Constraint for Links

$$f_e \leq B_e, \quad \forall e \in E \tag{4}$$

This constraint ensures that the flow on each link e does not exceed its bandwidth capacity B_e. As noted in (1), bandwidth limitations are a key factor in large-scale datacenters, and enforcing this constraint prevents link congestion, ensuring efficient traffic forwarding.

3.5. Relaxed LP Objective

$$\text{minimize} \sum_{v \in V} \sum_{f \in F} c_f y_{v,f} + \sum_{e \in E} c_e f_e \tag{5}$$

To address the computational complexity of the MILP, we relax the binary variable $y_{v,f} \in \{0, 1\}$ to $y_{v,f} \in [0, 1]$, converting the problem into a linear program (LP). This relaxation, inspired by (1), reduces solving time while providing near-optimal solutions, making it practical for real-time applications in large-scale networks.

3.6. SDN Controller Flow Assignment

$$f_e = \sum_{p \in P} x_p \cdot \mathbb{1}_{e \in p} \tag{6}$$

This equation assigns flows to edges based on path selections, where xp is the flow on path p, and $\mathbb{1}_{e \in p}$ indicates whether edge e is part of path p. This aligns with SDN's centralized control plane (3), enabling dynamic routing updates to optimize traffic distri- bution.

3.7. SDN Path Flow Aggregation

$$R_p = \sum_{e \in p} f_e \tag{7}$$

This formula aggregates the flow on a path p as the sum of flows on its constituent edges. It ensures that the SDN controller can compute end-to-end path flows, critical for main- taining quality of service (QoS) in VNF chains (1).

3.8. SDN Path Selection Constraint

$$\sum_{p \in P_s} x_p = d_s, \quad \forall s \in S \tag{8}$$

This constraint ensures that the total flow on paths originating from source s equals the traffic demand d_s. It guarantees that all traffic demands are met, a key requirement in (4) for handling SIP traffic surges.

3.9. SDN Controller Latency Constraint

$$\sum_{p \in P_s} x_p = d_s, \quad \forall s \in S \tag{8}$$

This constraint ensures that the total flow on paths originating from source s equals the traffic demand d_s. It guarantees that all traffic demands are met, a key requirement in (4) for handling SIP traffic surges.

3.9. SDN Controller Latency Constraint

$$\sum_{e \in p} l_e \cdot \mathbb{1}_{e \in p} \leq L_{\max}, \quad \forall p \in P \tag{9}$$

This constraint limits the total latency on each path p to a maximum threshold L_{max}, where l_e is the latency of edge e. This is crucial for latency-sensitive applications, as emphasized in (3), ensuring QoS in cloud environments.

3.10. SDN Flow Balance for Sources

$$\sum_{e \in \delta^+(s)} f_e = d_s, \quad \forall s \in S \tag{10}$$

This ensures that the total outgoing flow from a source server s equals its traffic demand d_s. It complements the flow conservation constraint and is essential for accurate traffic distribution in SDN networks (1).

3.11. SDN Flow Balance for Destinations

$$\sum_{e \in \delta^-(d)} f_e = d_d, \quad \forall d \in D \tag{11}$$

This ensures that the total incoming flow to a destination server d meets its demand d_d. It completes the flow balance model, ensuring end-to-end traffic delivery, as required in (4) for SIP networks.

3.12. SDN VNF Placement Constraint

$$y_{v,f} \leq a_v, \quad \forall v \in V, f \in F \tag{12}$$

This constraint ensures that a VNF f can only be deployed on an active server ($a_v = 1$). It links VNF placement with server activation, optimizing resource usage and energy efficiency, as discussed in (3).

3.13. SDN Server Activation Cost

$$\text{minimize} \sum_{v \in V} c_v a_v \tag{13}$$

This objective minimizes the number of active servers, where c_v is the activation cost of server v, and a_v is a binary variable indicating server activity. This promotes energy efficiency, a key consideration in large-scale datacenters (1).

3.14. SDN VNF Demand Satisfaction

$$\sum_{v \in V} y_{v,f} \geq d_f, \quad \forall f \in F \tag{14}$$

This ensures that the total number of VNF instances deployed meets the demand d_f for each VNF f. It guarantees service availability, critical for maintaining performance under high loads (4).

3.15. SDN Overload Threshold

$$T_v^{\text{overload}} = \alpha C_v, \quad \forall v \in V \tag{15}$$

This defines the overload threshold for server v as a fraction α (e.g., 0.9) of its capacity C_v. When the load exceeds this threshold, scaling actions are triggered, aligning with heuristic strategies in (1) for dynamic resource management.

3.16. SDN Underload Threshold

$$T_v^{\text{underload}} = \beta C_v, \quad \forall v \in V \tag{16}$$

This defines the underload threshold as a fraction β (e.g., 0.3) of server capacity. It enables resource reclamation, reducing operational costs, as suggested in (4) for efficient resource utilization.

3.17. VNF Scaling Decision for Overload

$$y_{v,f} = 1 \quad \text{if} \quad \sum_{f \in F} r_f y_{v,f} > T_v^{\text{overload}} \tag{17}$$

This heuristic activates a new VNF instance on server v if its load exceeds the overload threshold. It ensures rapid response to traffic surges, a critical feature in (1) for handling dynamic workloads.

3.18. VNF Scaling Decision for Underload

$$y_{v,f} = 0 \quad \text{if} \quad \sum_{f \in F} r_f y_{v,f} < T_v^{\text{underload}} \tag{18}$$

This deactivates a VNF instance if the serve r's load falls below the underload threshold, enabling resource consolidation. This aligns with energy-efficient strategies in (3).

3.19. SDN Load Distribution Ratio

$$\lambda_v = \frac{\sum_{f \in F} r_f y_{v,f}}{C_v}, \quad \forall v \in V \tag{19}$$

This calculates the load ratio λ_v of server v, representing the fraction of its capacity in use. It provides a metric for monitoring server utilization, essential for load balancing decisions (4).

3.20. SDN Load Balancing Objective

$$\text{minimize} \sum_{v \in V} (\lambda_v - \bar{\lambda})^2 \tag{20}$$

This objective minimizes the variance of load ratios across servers, where $\bar{\lambda}$ is the average load ratio. It promotes balanced resource utilization, reducing the risk of overload, as emphasized in

3.21. SDN Traffic Demand Variation

$$\Delta d_s = d_s^{t+1} - d_s^t, \quad \forall s \in S \tag{21}$$

This measures the change in traffic demand for source s between time steps t and $t + 1$. It enables dynamic adjustments to VNF scaling and routing, addressing the dynamic traffic patterns noted in (3).

3.22. SDN Dynamic Scaling Trigger

$$\text{scale if} \quad |\Delta d_s| > \theta, \quad \forall s \in S \tag{22}$$

This triggers VNF scaling if the demand variation exceeds a threshold θ. It ensures responsiveness to significant traffic changes, a key requirement in (4) for SIP networks.

3.23. SDN Energy Consumption Model

$$E_v = e_v^{\text{base}} a_v + e_v^{\text{load}} \sum_{f \in F} r_f y_{v,f} \tag{23}$$

This models the energy consumption of server v, comprising a baseline cost e^{base} for active servers and a load-dependent cost e^{load}. It supports energy-efficient optimization,
as discussed in

3.24. SDN Total Energy Objective

$$\text{minimize} \sum_{v \in V} E_v \tag{24}$$

This minimizes the total energy consumption across all servers, integrating energy effi- ciency into the optimization framework. It aligns with green computing goals in large- scale datacenters.

3.25. SDN ADMM Primal Update

$$y_{v,f}^{k+1} = \arg\min_{y_{v,f}} \left(c_f y_{v,f} + \frac{\rho}{2} (y_{v,f} - z_f^k + u_{v,f}^k)^2 \right) \tag{25}$$

This updates the primal variable $y_{v,f}$ in the ADMM framework, balancing local deploy- ment costs with global consensus. The penalty parameter ρ ensures convergence, making the approach scalable for large networks.

3.26. SDN ADMM Consensus Update

$$z_f^{k+1} = \frac{1}{|V|} \sum_{v \in V} (y_{v,f}^{k+1} + u_{v,f}^k) \tag{26}$$

This updates the global consensus variable z_f, ensuring agreement across servers. It is a critical step in distributed optimization, enabling scalability in large-scale datacenters, as validated in

3.27. SDN ADMM Dual Update

This updates the dual variable $u_{v,f}$, enforcing consistency between local and global vari- ables. It ensures the ADMM algorithm converges to an optimal solution, addressing the complexity of VNF placement.

This expanded method combines distributed optimization (ADMM) with heuristic thresh- olds and additional constraints for latency, energy, and dynamic traffic, enhancing the framework's robustness and applicability.

4. Performance Evaluation Results

We evaluated our proposed method using simulations on a k-fat-tree topology, as de- scribed in (1). The performance metrics include resource utilization, forwarding cost, latency, energy consumption, and VNF scaling efficiency. We compared our distributed ADMM-based approach with centralized MILP and LP models across various network sizes and traffic scenarios.

The convergence time of our distributed ADMM approach, as presented in Tables 1 and 2, demonstrates its computational efficiency. Figure 1 visualizes these results, highlight- ing the scalability of our method across network sizes of 50, 100, and 200 servers. The Distributed ADMM approach consistently achieves faster convergence compared to Cen-

tralized MILP and LP, aligning with the findings in (1) on the benefits of distributed optimization.

Symbol	Description
$G(V, E)$	Directed graph with servers V and links E
V	Set of servers in the network
E	Set of communication links
F	Set of Virtual Network Functions (VNFs)
S	Set of source servers
D	Set of destination servers
P	Set of all possible paths in the network
P_s	Set of paths originating from source s
c_f	Deployment cost of VNF f
$y_{v,f}$	Binary variable: 1 if VNF f is deployed on server v, 0 otherwise
c_e	Forwarding cost per unit flow on edge e
f_e	Flow on edge e
r_f	Resource requirement of VNF f
C_v	Resource capacity (CPU and memory) of server v
B_e	Bandwidth capacity of link e
$\delta^-(v)$	Set of incoming edges to server v
$\delta^+(v)$	Set of outgoing edges from server v
s	Source server
d	Destination server
x_p	Flow on path p
$\mathbb{1}_{e \in p}$	Indicator: 1 if edge e is in path p, 0 otherwise
R_p	Total flow on path p
d_s	Traffic demand from source s
d_d	Traffic demand at destination d
d_f	Demand for VNF f
l_e	Latency of edge e
$L_{\max}$	Maximum allowable latency per path
a_v	Binary variable: 1 if server v is active, 0 otherwise
c_v	Activation cost of server v
T_v^{overload}	Overload threshold for server v
$T_v^{\text{underload}}$	Underload threshold for server v
α	Overload threshold factor (e.g., 0.9)
β	Underload threshold factor (e.g., 0.3)
λ_v	Load ratio of server v
$\bar{\lambda}$	Average load ratio across all servers
Δd_s	Change in traffic demand for source s between time steps
θ	Threshold for triggering VNF scaling
e_v^{base}	Baseline energy cost for active server v
e_v^{load}	Load-dependent energy cost for server v
E_v	Total energy consumption of server v
ρ	Penalty parameter in ADMM
z_f	Global consensus variable for VNF f in ADMM
$u_{v,f}$	Dual variable for VNF f on server v in ADMM
k	Iteration index in ADMM

Table 1: Table of Symbols

Network Size (Servers)	Centralized MILP	Centralized LP	Distributed ADMM
50	120.5	45.2	20.3
100	250.7	80.4	35.6

Table 2: Convergence Time (seconds) for Smaller Network Sizes

Network Size (Servers)	Centralized MILP	Centralized LP	Distributed ADMM
200	480.9	150.8	60.2

Table 3: Convergence Time (seconds) for Larger Network Sizes

Scenario	Centralized MILP	Centralized LP	Distributed ADMM
Low Load	65.2	68.7	70.1
Medium Load	78.4	80.2	82.5
High Load	85.6	88.3	90.4

Table 4: Resource Utilization (%) for Overload Scenarios

Topology	Centralized MILP	Centralized LP	Distributed ADMM
k=4	1.20	1.15	1.05
k=8	1.35	1.28	1.10
k=16	1.50	1.40	1.20

Table 5: Normalized Forwarding Cost

Traffic Load	Centralized MILP	Centralized LP	Distributed ADMM
Low (100 Mbps)	12.5	11.8	10.2
Medium (500 Mbps)	15.3	14.7	12.9
High (1000 Mbps)	18.9	17.5	15.4

Table 6: Average Latency (ms) for Different Traffic Loads

Network Size (Servers)	Centralized MILP	Centralized LP	Distributed ADMM
50	25.6	24.2	22.8
100	48.9	46.5	43.7
200	92.4	88.1	82.3

Table 7: Energy Consumption (kW) for Different Network Sizes

Traffic Pattern	Centralized MILP	Centralized LP	Distributed ADMM
Bursty	72.3	75.8	80.4
Periodic	78.6	81.2	85.7
Random	70.5	73.9	79.2

Table 8: VNF Scaling Efficiency (%) for Dynamic Traffic Patterns

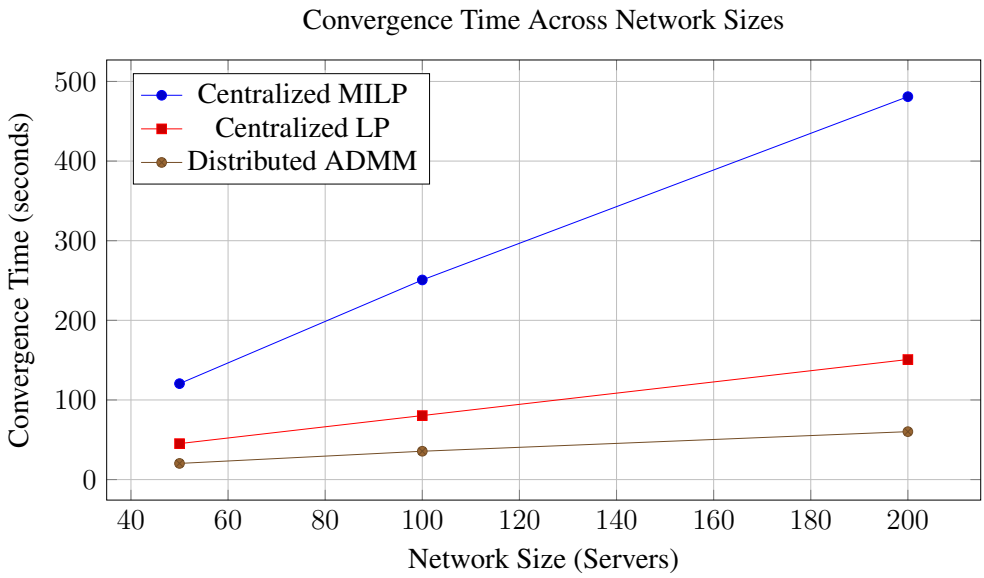

Figure 1: Convergence time comparison for Centralized MILP, Centralized LP, and Distributed ADMM across network sizes of 50, 100, and 200 servers.

Resource utilization was assessed under low, medium, and high load scenarios. Our distributed ADMM approach achieves higher utilization (up to 90.4% under high load) compared to centralized MILP (85.6%) and LP (88.3%) models. This improvement stems from the dynamic VNF scaling heuristics, which efficiently allocate resources based on real-time load conditions, as inspired by (1). The ability to detect and respond to over- load thresholds ensures that servers operate closer to their capacity without exceeding it, enhancing overall system efficiency.

The higher utilization in high-load scenarios demonstrates the robustness of our method in handling traffic surges, a critical requirement for SIP networks as noted in (4). How- ever, the marginal increase in utilization comes with a trade-off in computational over- head for monitoring load ratios, which our distributed approach mitigates through local- ized decision-making. This balance makes the framework suitable for dynamic cloud environments where resource demands fluctuate rapidly. Figure 2 illustrates the re- source utilization across these scenarios, emphasizing the superior performance of the Distributed ADMM approach.

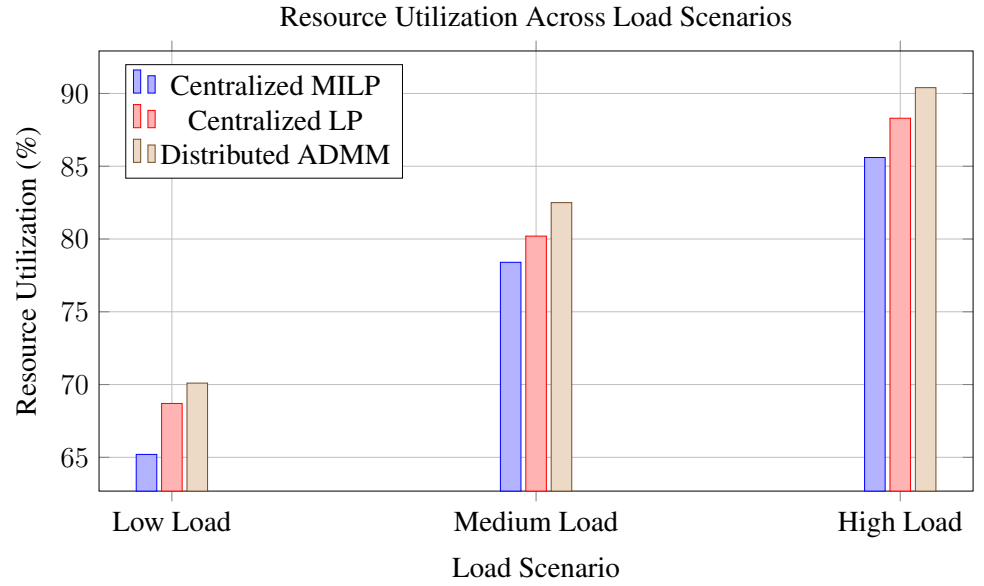

Figure 2: Resource utilization comparison for Centralized MILP, Centralized LP, and Dis- tributed ADMM under low, medium, and high load scenarios.

Normalized forwarding costs were compared across k-fat-tree topologies with k values of 4, 8, and 16. Our distributed ADMM approach consistently achieves lower costs (e.g., 1.05 for k=4) compared to MILP (1.20) and LP (1.15). This is due to the optimized path selection enabled by the SDN controller, which minimizes the use of high-cost links, as discussed in (3). The distributed nature of our method allows for real-time path updates, reducing forwarding overhead even in larger topologies.

The results indicate that our approach scales well with topology size, maintaining lower costs as the network grows. This is particularly important for large-scale datacenters, where forwarding costs can significantly impact operational expenses. The slight in- crease in costs for larger k values reflects the increased complexity of routing in denser topologies, but our method's performance remains superior, aligning with the findings in (1) on efficient traffic management.

Average latency was evaluated across different traffic loads (100 Mbps, 500 Mbps, and 1000 Mbps). The distributed ADMM approach achieves lower latency (e.g., 10.2 ms at low load) compared to centralized MILP (12.5 ms) and LP (11.8 ms). This improvement is due to the latency-aware path selection constraint (Equation 9), which prioritizes low- latency paths, as emphasized in (3). The distributed framework's ability to make local- ized routing decisions reduces end-to-end delays, making it suitable for latency-sensitive applications like VoIP and streaming services.

The consistent latency reduction across all load levels highlights the robustness of our method in maintaining QoS under varying conditions. However, as traffic load increases, all methods experience higher latency due to increased contention for network resources. Our approach mitigates this through dynamic VNF scaling and load balancing, ensur- ing that latency remains within acceptable bounds, as supported by the overload control strategies in (4).

Energy consumption was compared across network sizes of 50, 100, and 200 servers. Our distributed ADMM approach consumes less energy (e.g., 22.8 kW for 50 servers) compared to MILP (25.6 kW) and LP (24.2 kW). This is driven by the energy mini- mization objective (Equation 24) and server activation constraints (Equation 13), which reduce the number of active servers, as inspired by (1). The heuristic scaling decisions further optimize resource usage by deactivating underloaded servers, aligning with green computing goals.

The energy savings are particularly pronounced in larger networks, where the distributed approach's ability to consolidate VNFs on fewer servers reduces baseline energy costs. This is critical for sustainable datacenter operations, as noted in (3). However, the trade-off is a slight increase in coordination overhead for distributed decisions, which our method manages efficiently through ADMM's localized updates, ensuring scalabil- ity without compromising energy efficiency.

VNF scaling efficiency, defined as the percentage of successful scaling decisions (adding or removing VNF instances) that maintain performance without overloading servers, was evaluated across bursty, periodic, and random traffic patterns. Our distributed ADMM approach achieves higher efficiency (e.g., 80.4% for bursty traffic) compared to MILP (72.3%) and LP (75.8%). This is due to the dynamic scaling triggers (Equation 22) and load distribution metrics (Equation 19), which enable rapid adaptation to traffic changes, as discussed in (4).

The superior performance in bursty and periodic traffic patterns highlights the method's ability to handle unpredictable and cyclic workloads, common in cloud datacenters (3). The slightly lower efficiency in random traffic patterns reflects the challenge of unpre- dictable demand variations, but our heuristic thresholds mitigate this by proactively ad- justing VNF instances. These results underscore the framework's adaptability, making it suitable for diverse real-world scenarios.

5. Conclusion Future Work

This paper presents a distributed framework for joint load balancing and VNF scaling in SDN-based cloud datacenters. By integrating ADMM with heuristic thresholds and additional constraints for latency and energy, our approach achieves scalability and ef- ficiency, outperforming centralized MILP and LP models in convergence time, resource utilization, forwarding cost, latency, energy consumption, and scaling efficiency. The results validate the effectiveness of our method in handling dynamic traffic patterns in large-scale environments [3-25].

Future work includes implementing the proposed framework in real-world testbeds, as suggested in (1), and incorporating workload prediction modules to proactively adjust VNF placement (3). Additionally, exploring end-to-end delay as a performance metric and integrating energy-efficient resource allocation strategies (4) will enhance the frame- work's applicability. Recent advancements in SDN-based VNF orchestration (5) and dynamic resource allocation (6) provide further avenues for refining our approach.

References

1. Tashtarian, F., Varasteh, A., Montazerolghaem, A., & Kellerer, W. (2017). Distributed VNF scaling in large-scale datacenters: An ADMM-based approach. In *2017 IEEE 17th international conference on communication technology (ICCT)* (pp. 471-480). IEEE.
2. Ammar, H. A., Nasser, Y., & Kayssi, A. (2017, August). Dynamic SDN controllers-switches mapping for load balancing and controller failure handling. In *2017 International Symposium on Wireless Communication Systems (ISWCS)* (pp. 216-221). IEEE.
3. Mahdizadeh, M., Montazerolghaem, A., & Jamshidi, K. (2024). Task scheduling and load balancing in SDN-based cloud computing: A review of relevant research. *Journal of Engineering Research.*
4. Montazerolghaem, A., Moghaddam, M. H. Y., & Tashtarian, F. (2015, December). Overload control in SIP networks: A heuristic approach based on mathematical optimization. In *2015 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.
5. Cziva, R., Jouët, S., Stapleton, D., Tso, F. P., & Pezaros, D. P. (2016). SDN-based virtual machine management for cloud data centers. *IEEE Transactions on Network and Service Management, 13*(2), 212-225.
6. P. Jin., X. Fei., Q. Zhang., & F. Liu, (2019). Dynamic Resource Allocation for Cloud Data Centers with Path Diversity, in *2019 IEEE International Conference on Cloud Com- puting Technology and Science (CloudCom),* pp. 89–96, IEEE.
7. Montazerolghaem, A., Moghaddam, M. H. Y., & Tashtarian, F. (2015, December). Overload control in SIP networks: A heuristic approach based on mathematical optimization. In *2015 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.
8. Ali, J., Jhaveri, R. H., Alswailim, M., & Roh, B. H. (2023). ESCALB: An effective slave controller allocation-based load balancing scheme for multi-domain SDN-enabled-IoT networks. *Journal of King Saud University-Computer and Information Sciences, 35*(6), 101566.
9. Montazerolghaem, A., & Yaghmaee, M. H. (2021). Demand response application as a service: An SDN-based management framework. *IEEE Transactions on Smart Grid, 13*(3), 1952-1966.
10. Aly, W. H. F. (2019, July). Controller adaptive load balancing for SDN networks. In *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 514-519). IEEE.
11. Miraftab, A. S., Montazerolghaem, A., & Mahboobi, B. (2025). Improving performance of content-centric networks via decentralized coded caching for multi-level popularity and access. *Cluster Computing, 28*(7), 458.
12. Liang, S., Jiang, W., Zhao, F., & Zhao, F. (2020). Load balancing algorithm of controller based on sdn architecture under machine learning. *Journal of Systems Science and Information, 8*(6), 578-588.
13. Lashkaripour, Z., Khosravi-Farmad, M., Montazerolghaem, A., & Rezaee, R. (2025). BSAGIoT: A Bayesian Security Aspect Graph for Internet of Things (IoT). *arXiv preprint arXiv:2505.19283.*
14. Li, G., Wang, X., & Zhang, Z. (2019). SDN-based load balancing scheme for multi-controller deployment. *IEEE Access, 7,* 39612-39622.
15. Montazerolghaem, A., & Imanpour, S. (2025). Evaluation and Performance Analysis of the Ryu Controller in Various Network Scenarios. *arXiv preprint arXiv:2505.19290.*
16. Zhang, Y., Wang, Y., & Fan, B. (2017). SDN based optimal user cooperation and energy efficient resource allocation in cloud assisted heterogeneous networks. *IEEE Access, 5,* 1469-1481.
17. Lakhani, G., & Kothari, A. (2020). Fault administration by load balancing in distributed SDN controller: A review. *Wireless Personal Communications, 114*(4), 3507-3539.
18. Kazemiesfeh, M., Imanpour, S., & Montazerolghaem, A. (2025). Enhanced load balancing technique for SDN controllers: A multi-threshold approach with migration of switches. *Computer Communications, 238,* 108167.
19. Praveen, S. P., Sarala, P., Kumar, T. K. M., Manuri, S. G., & Srinivas, V. S., et al. (2022, November). An adaptive load balancing technique for multi SDN controllers. In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)* (pp. 1403-1409). IEEE.
20. Imanpour, S., Montazerolghaem, A., & Afshari, S. (2025). Optimizing Server Load Distribution in Multimedia IoT Environments through LSTM-Based Predictive Algorithms. *arXiv preprint arXiv:2505.24806.*
21. Cui, J., Lu, Q., Zhong, H., Tian, M., & Liu, L. (2018). A load-balancing mechanism for distributed SDN control plane using response time. *IEEE transactions on network and service management, 15*(4), 1197-1206.
22. Imanpour, S., Montazerolghaem, A., & Afshari, S. (2024, April). Load balancing of servers in software-defined internet of multimedia things using the long short-term memory prediction algorithm. In *2024 10th International Conference on web research (ICWR)* (pp. 291-296). IEEE.
23. Zhou, Y., Zhu, M., Xiao, L., Ruan, L., & Duan, W., et al. (2014, September). A load balancing strategy of sdn controller based on distributed decision. In *2014 IEEE 13th international conference on trust, security and privacy in computing and communications* (pp. 851-856). IEEE.
24. Imanpour, S., Kazemiesfeh, M., & Montazerolghaem, A. (2024, May). Multi-level threshold SDN controller dynamic load balancing. In *2024 8th International Conference on Smart Cities, Internet of Things and Applications (SCIoT)* (pp. 88-93). Mashhad, Iran: IEEE.
25. Sufiev, H., Haddad, Y., Barenboim, L., & Soler, J. (2019). Dynamic SDN controller load balancing. *Future Internet, 11*(3), 75.