



AN OPTIMAL ROUTING MODEL FOR SDN NETWORKS TO MINIMIZE DELAY

Mohammad Mohammadi Anaei^{1*}, Maleeha Hashmipour²

¹Master of Science in Software Engineering, Islamic Azad University, Tehran Science and Research Branch Tehran, Iran.

²Faculty Member, Department of Computer Engineering, Islamic Azad University, Kerman, Iran

Corresponding author: Mohammad Mohammadi Anaei

ABSTRACT

This study proposes an optimal routing model for Software-Defined Networking (SDN) aimed at minimizing network delays. Proper placement of controllers can reduce traffic load, leading to faster data processing and lower data transmission latency. To achieve this, the Whale Optimization Algorithm (WOA) was utilized. The approach leverages information from the controllers about connected switches to enhance network throughput, while also reducing resource usage and improving scalability by minimizing delays. WOA is applied to optimize scheduling and routing within SDN networks, targeting improved energy efficiency among devices based on Wireless Sensor Networks (WSNs). Simulation results demonstrate that the proposed algorithm outperforms other existing methods in terms of key metrics such as authentication key breaking time, network resource consumption, load balancing, delay reduction, and maximum link utilization. Additionally, the algorithm can select the optimal path according to user priorities, providing a comprehensive performance evaluation. Results further indicate that the algorithm performs particularly well in larger network topologies, such as the “Internet Zoo Topology,” and reduces the number of controllers required by strategically placing them in optimal locations.

Keywords: SDN networks, optimal routing, delay minimization, Whale Optimization Algorithm (WOA).

INTRODUCTION

Software-Defined Networks (SDNs) represent a modern approach to computer network architecture, where network intelligence is logically centralized in software controllers, and the hardware serves primarily as simple packet-forwarding devices that can be programmed through an open interface [1]. SDNs aim to enhance network intelligence by transferring

control of data from hardware switches and routers to virtualized software layers. By employing a centralized software controller, SDNs provide capabilities such as programmability, scalability, flexibility, automation, intelligence, and software-driven network development [2]. Compared to traditional networks, SDNs improve flexibility and reduce the complexity of managing network flows [3]. Although most SDN research has focused on wired networks, only a few studies have explored its benefits in wireless domains [4]. To address this, Software-Defined Wireless Sensor Networks (SD-WSNs) have been proposed, extending SDN advantages to wireless networks [5]. Unlike distributed wireless control, an SDN controller can optimize wireless network performance—including energy consumption, delay, and traffic flows—based on a global network perspective. Wireless nodes, however, cannot fully separate the data and control planes; while they are not directly connected to the SDN controller, they still need to receive routing instructions from it. As a result, each node maintains a local routing table for control flows [6,7]. Several studies have explored SDN-based routing in wireless sensor networks. In [8], a MapReduce-based Dijkstra algorithm was proposed for QoS routing in SDN-based WSNs, modeling the network as a graph. Source-destination pairs were evaluated to calculate minimal path distances, and the Dijkstra algorithm was applied to determine optimal shortest paths. In [9], SDN-based routing was applied to multi-hop WSNs, where a dedicated controller monitors the entire network and selects optimal paths based on node energy and hop count.

Maintaining a global view requires exchanging multiple control packets, increasing bandwidth usage and addressing node failures due to energy depletion. Other approaches have focused on clustering and energy efficiency. The WARIS clustering algorithm was introduced in [10] for large-scale SD-WSNs, improving cluster formation through energy-aware cluster head (CH) selection, reclustering, and CH rotation. In [11], a congestion control mechanism was used to achieve better QoS, employing an OpenFlow-based Active Network Management system to collect link and connection status from all nodes. A green routing protocol for IoT based on SD-WSN architecture was proposed in [12], while [13] introduced a centralized routing protocol with a sink node for large-scale SD-WSNs. In [14], SDN-enabled UAVs were used to create a dynamically managed virtual topology for WSN nodes. A secure three-layer SD-WSN architecture (SeC-SDWSN) was proposed in [15], enhancing network quality and security while improving performance by 5%, PDR by 7.8%, and energy efficiency by 16%. In [16], CAPONE, a cluster-based message control system for software-defined aerial ad-hoc networks, allowed UAV controllers to predict node information without transmitting control messages.

To fully exploit SDN in wireless networks, it is essential to balance the benefits and communication costs of SDN control among network components. This study proposes an optimal routing approach designed to reduce the number of nodes involved in flow management within SDN-based wireless networks. SDN plays a key role in ensuring network performance regarding traffic and task response delay. Acting as a mediator, SDN controllers maintain stable flows and balance traffic load, which also enhances network security.

Traffic engineering algorithms based on user authentication and SDN are applied over data center networks. Effective controller placement and load distribution are crucial, as precise placement can reduce traffic load, accelerate data processing, and minimize transmission delays. To achieve this, the Whale Optimization Algorithm (WOA) is employed. This research seeks a method that utilizes controller-switch information to maximize throughput, minimize resource usage, and enhance network scalability by reducing latency.

RESEARCH METHODOLOGY

Proposed Algorithm:

In this study, the Whale Optimization Algorithm (WOA) is employed to achieve optimal scheduling and routing in SDN networks, aiming to enhance energy efficiency among WSN-based devices. WOA is inspired by the hunting and encircling behavior of whales. The algorithm starts with an initial population of randomly generated solutions. During each iteration, search agents update their positions based on either a randomly selected agent or the current best solution.

The parameter a decreases linearly from 2 to 0 to balance exploration and exploitation. When $|A| > 1$, the update relies on a randomly chosen search agent, whereas when $|A| < 1$, the update is guided by the best solution found so far. Depending on the probability parameter p , WOA switches between spiral and circular movements, mimicking the whales' hunting behavior. The algorithm terminates once the stopping criteria are satisfied [17].

WOA assumes that the current best candidate represents either the target prey or a solution near the optimum. After identifying the best search agent, the remaining agents adjust their positions relative to it. This adaptive behavior is expressed mathematically through the following equations:

$$(1) \quad \vec{D} = \left| \vec{C} \cdot \vec{X}^*(t) - \vec{X}(t) \right|$$

$$(2) \quad \vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D}$$

In these equations, t denotes the current iteration, A and C are coefficient vectors, X^* represents the position vector of the current best solution, X is the position vector of a search agent, $||$ denotes the absolute value, and $\cdot$ represents element-wise multiplication. It should be noted that X^* must be updated in each iteration if a better solution is found.

The vectors A and C are computed as follows:

$$(3) \quad \vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a}$$

$$(4) \quad \vec{C} = 2 \cdot \vec{r}$$

The parameter a decrease linearly from 2 to 0 over the course of iterations, applying to both the exploration and exploitation phases, and r is a random vector with values in the range $[0, 1]$.

The pseudocode of the WOA algorithm is shown below. From a theoretical perspective, WOA can be considered a global optimization method, as it incorporates both exploration and exploitation capabilities. Additionally, the spiral mechanism defines a search space around the current best solution, allowing other agents to leverage the best historical information within the neighborhood. The adaptive variation of the search vector A enables WOA to gradually transition between exploration and exploitation: when $(|A| \geq 1)$, the iteration focuses on exploration, while $(|A| < 1)$ corresponds to exploitation. It is noteworthy that WOA has only two main tunable internal parameters, A and C [17].

Initialize the whales population $X_i (i = 1, 2, \dots, n)$

Calculate the fitness of each search agent

```

X*=the best search agent
While (t < maximum number of iterations)
  For each search agent
    Update a, A, C, l, and p
    if1(p<0.5)
      if2(|A|< 1)
        Update the position of the current search agent by the Eq.(2.1)
      else if2(|A|≥1)
        Select a random search agent (Xrand)
        Update the position of the current search agent by the Eq. (2.8)
      end if2
    else if1(p0.5)
      Update the position of the current search by the Eq. (2.5)
    end if1
  end for
  Check if any search agent goes beyond the search space and amend it
  Calculate the fitness of each search agent
  Update X* if there is a better solution
  t=t+1
end while
return X*

```

For each task, there is an associated metric measured in dollars per hour. The fitness of each solution in the proposed algorithm is calculated using equations (5) through (39).

$$(5) \quad C_{exe}(M)_j = \sum_k w_{kj} \quad \forall M(k) = j$$

$$(6) \quad C_{tx}(M)_j = \sum_{k1 \in T} \sum_{k2 \in T} d_{M(k1), M(k2)} e_{k1, k2} \quad , \quad \forall M(k1) = j \text{ and } M(k2) \neq j$$

$$(7) \quad C_{total}(M)_j = C_{exe}(M)_j + C_{tx}(M)_j$$

$$(8) \quad cost(M) = \max(C_{total}(M)_j)$$

$$(9) \quad Minimize(Cost(M))$$

The symbol $C_{exe}(M)_j$ denotes the **total execution cost** of all tasks assigned to controller PC_j , calculated according to equation (5). This value is obtained by summing the weights of all nodes (i.e., the execution cost of task k on the controller) corresponding to the workloads assigned to each controller in mapping M .

Additionally, $C_{exe}(M)_j$ also represents the total access cost (including transfer costs) between tasks assigned to controller PC_j and tasks that are not assigned to any resource in mapping M , computed according to equation (6). This cost is calculated as the product of the output file size (edge weight $e_{(k1, k2)}$) of tasks $k1 \in T$ and $k2 \in T$ and the communication cost from the controller, assuming that $k1$ is mapped to the same controller as $k2$. The average data communication cost between two devices is given by $d_{(M(k1), M(k2))}$. Communication costs are only incurred if a file dependency exists between two tasks ($e_{(k1, k2)} > 0$); for two or more tasks executed on the same controller, the communication cost is zero.

For a given mapping M , the total cost of controller PC_j is calculated according to equation (7) as the sum of the execution and access costs. When the total cost across all controllers is estimated, the maximum cost among all resources is minimized according to equation (9). This approach ensures that tasks are not all mapped to a single device and achieves a balanced cost distribution among controllers.

The pseudocode of the proposed algorithm is as follows:

Compute the average computational cost of all tasks across all controllers.

Compute the average communication cost, i.e., the size of data exchanged between controllers.

Set the node weights W_{kj} to the average computational cost.

Set the edge weights $e_{k1, k2}$ to the size of data transferred between tasks.

For all ready tasks $\{t_i\} \in T$, assign task t_i to a controller PC_j and store it in the mapping M_i .

Compute $COA(\{M_i\})$ such that the cost function is minimized.

Repeat steps 5 and 6 until all tasks are scheduled.

In this study, after simulating the wireless network (with specifications listed in Table 1) and the SDN network (specifications in Table 2), the two networks were integrated, and the simulation was implemented in MATLAB according to the proposed research methodology.

Table 1: Wireless Network Specifications

Parameters	Values
Number of Nodes	200
Initial Energy E_0	$50J$
Initial Energy of Sensor Nodes	$E_0(1 + \alpha)$
Energy for Data Aggregation E_{DA}	$5 \text{ nJ} / \text{bit} / \text{signal}$
Energy for Receiving and Transmitting E_{elec}	$5 \text{ Pj} / \text{bit}$
Reinforcement Energy for Short Distances E_{fs}	$10 \text{ Pj} / \text{bit} / \text{m}^2$
Reinforcement Energy for Long Distances E_{amp}	$0.013 \text{ Pj} / \text{bit} / \text{m}^4$
Probability P_{opt}	0.1

Table 2: SDN Network Specifications

Parameters	Values
Number of Controllers	$K = 4$
Number of Core Layer Switches	$\left(\frac{K}{2}\right)^2$
Number of Aggregation Layer Switches	$\frac{K^2}{2}$
Number of Edge Layer Switches	$\frac{K^2}{2}$
Total Number of Switches	$\frac{5K^2}{4}$
Number of Hosts	$\frac{K^2}{4}$
Number of Hosts per Controller	$\frac{K^2}{4}$

Number of Paths between Each Pair of Hosts	$\frac{K^2}{4}$
Total Number of Links	$\frac{3K^3}{4}$

Proposed Design Requirements:

In this design, it is assumed that all switches—both in the core and at the network edge—are OpenFlow-enabled and connected to a central controller. It is also assumed that the central controller can access all switches with an acceptable level of latency. Since the proposed design primarily targets WAN networks, in practical scenarios, deploying only a single central controller may not be feasible. In such cases, multiple controllers can be installed, which coordinate with each other to create a logically centralized control layer. In this scenario, maintaining database consistency among controllers becomes a separate issue beyond the scope of the present study. Supporting all switches for pod labeling is another requirement of the design. Additionally, the network must have a two-tier structure. In the network where this design is implemented, a subset of switches acts as core switches, while the remaining edge switches interconnect with them. Users connect only to the edge switches [18]. To better illustrate performance, the proposed method, based on a D2D approach, is compared with the MOPSO method described by Ho et al. (2019) [19]. The key distinction of the present study lies in the application of traffic engineering, load balancing, and delay reduction within the network. Link utilization is defined as the ratio of traffic load on a link to its capacity. The goal of the proposed system is to minimize the maximum link utilization. Reducing the peak link utilization ensures balanced load distribution and prevents congestion.

In the proposed method, the evaluation begins from the switches. Step by step, a position-based packet is sent to other controllers. The other controllers quickly receive this information, allowing them to determine their relative positions with respect to the source switch based on signal strength and direction. Each controller, after receiving position information, broadcasts its position to the other controllers, taking into account communication links and available paths across different layers. This process continues iteratively until all controllers are informed.

The primary performance metrics for evaluating the localization system are accuracy and precision. Simulations were conducted using MATLAB version 2021a on a system with a 5-core processor and 8 GB of RAM.

FINDINGS

To evaluate controller placement, the study utilizes network topology maps from the standard Internet Zoo Topology database. Specifically, the results for the “NetworkUsa” topology, which consists of 29 nodes, are presented.

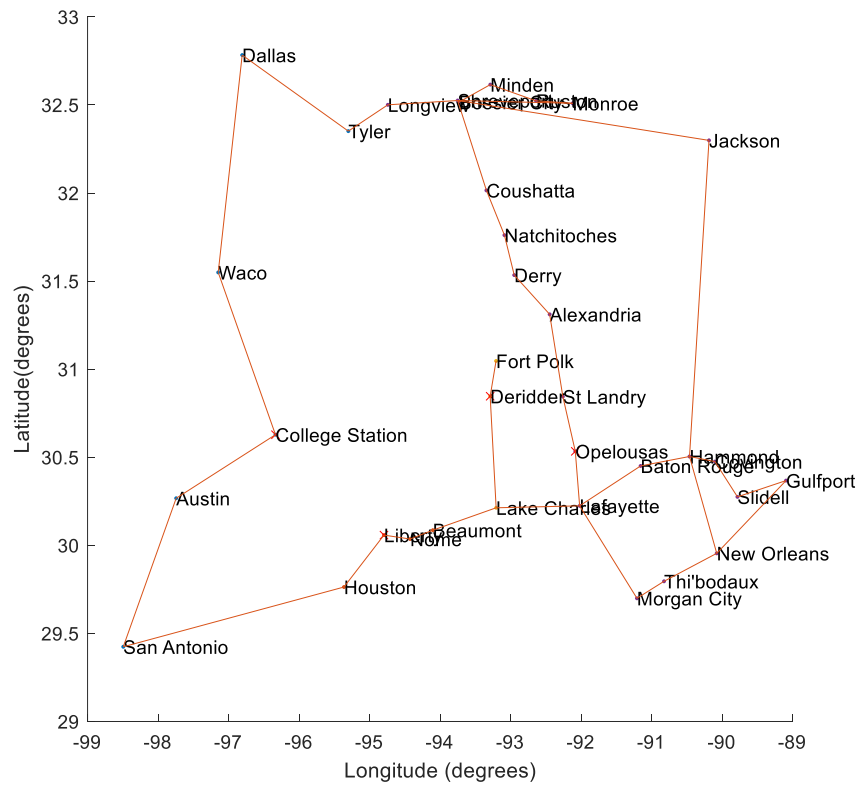


Figure 1: Core Controller Network Based on Communication Links

As shown in Figure 1, each controller initially creates a list of its neighboring controllers, recording their IDs, positions, and the task load assigned to each in a table. This process begins with each controller sending an initial message to its neighboring controllers, informing them of its position and the current task load. The effectiveness of this communication is directly influenced by the number of available paths and the bandwidth of the controller links. Each controller stores the received messages in its local vector according to the task load. Based on a priority function, the controller then determines the potential placement of each neighboring controller. This process is illustrated in Figure 2.

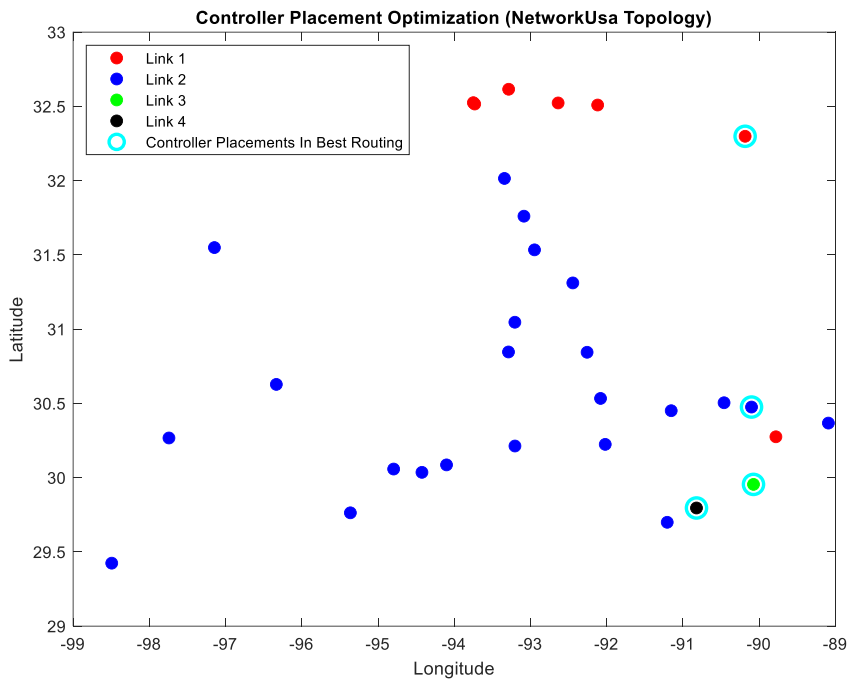


Figure 2: Evaluation of Network Links for Optimal Path Selection Based on Controller Placement

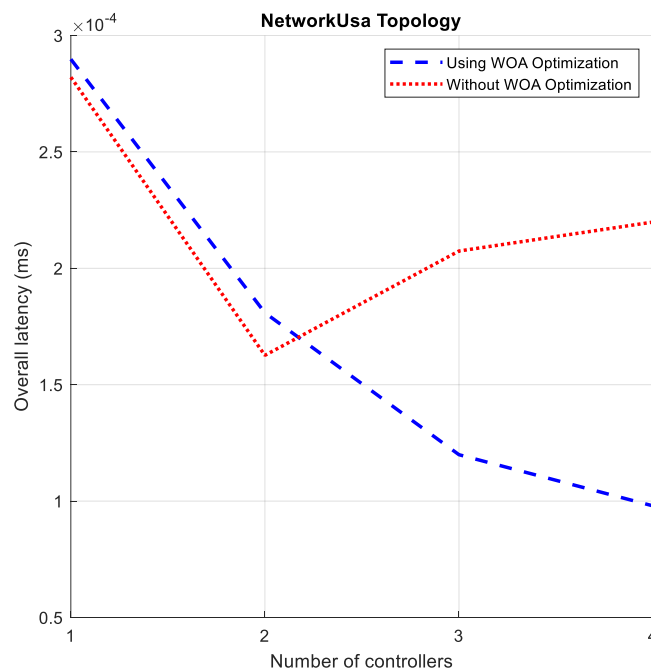


Figure 3: Overall Data Transmission Delay Before and After Applying the Whale Optimization Algorithm

As shown in Figure 3, increasing the number of controllers and the link bandwidth can lead to higher transmission delays if packets are not routed through optimal paths. However, after applying the Whale Optimization Algorithm to determine the optimal controller placement, traffic management and routing are significantly improved. The data transmission rate increases before controller shutdown, and as the number of controllers grows, the packet delivery rate also rises. Because packets follow optimal paths and network load is balanced, overall transmission delay is reduced.

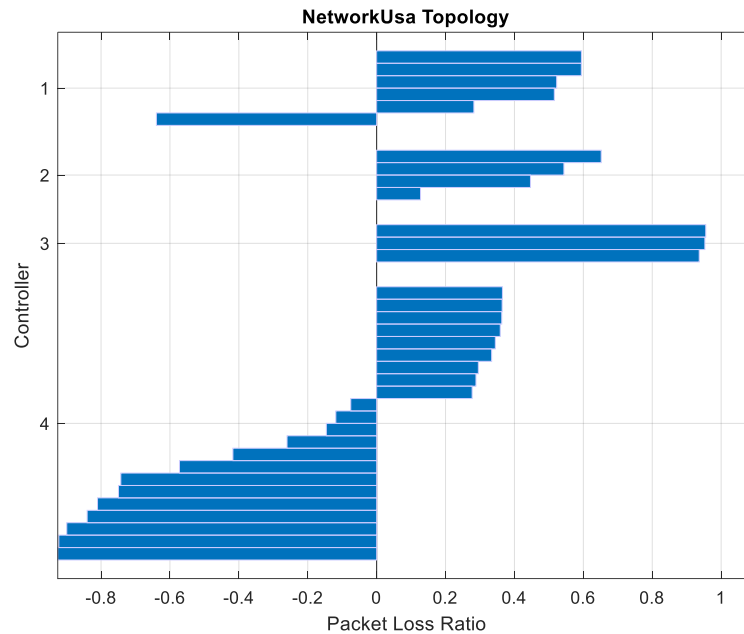


Figure 4: Comparison of Packet Loss Rate Using the Whale Optimization Algorithm

The impact of link bandwidth is considered alongside the increase in the number of controllers. Figure 4 illustrates that packet loss decreases as the number of controllers increases. As observed, applying the Whale Optimization Algorithm improves packet delivery by approximately 70% through optimal controller placement, reduced controller overhead, and better network load balancing. In other words, as the number of controllers increases from 1 to 4, the packet loss rate decreases from around 0.7 to approximately 0.21. Furthermore, it is expected that the waiting time for packet transmission between controllers will also decrease as the number of controllers increases, as shown in Figure 5.

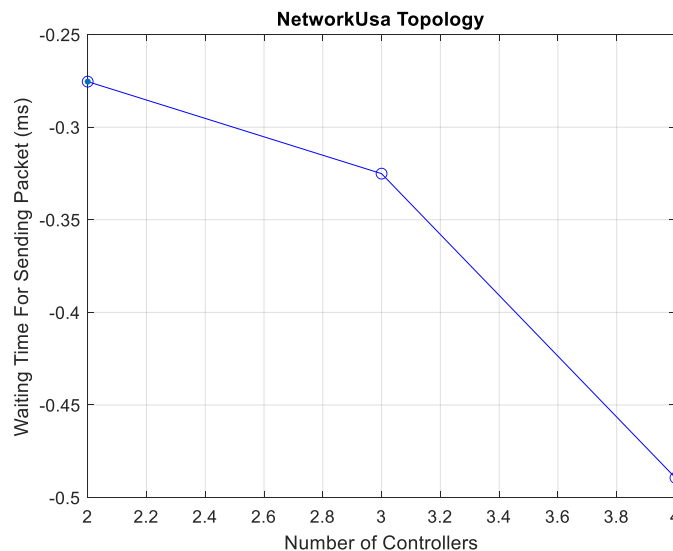


Figure 5: Packet Transmission Waiting Time

Additionally, the quality of service (QoS) for network links is also improved, as illustrated in Figure 6.

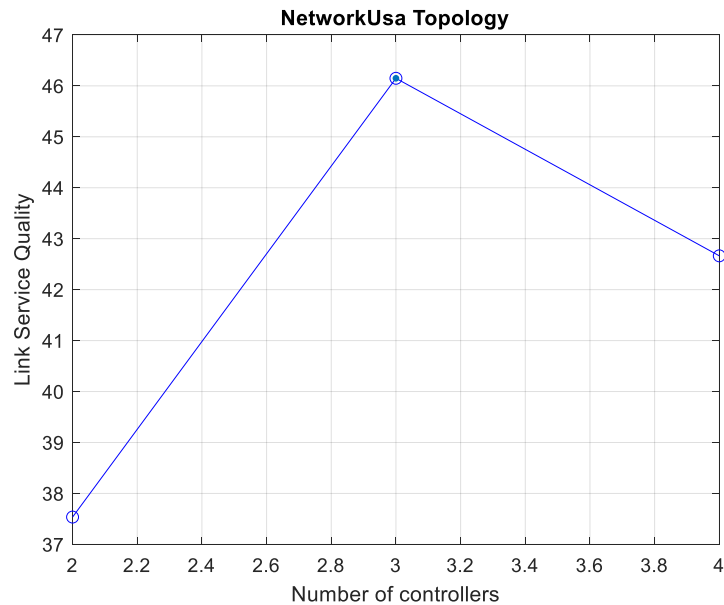


Figure 6: Link Quality of Service Variations with Respect to the Number of Controllers

Finally, the performance of the proposed controller placement method in improving SDN network efficiency is examined. A load distribution module was introduced to achieve network balance: when the transmission delay exceeds a predefined threshold within a given bandwidth, controllers establish optimal communication between network components based on the allocated demand. In this implementation, both the number of tasks and their variations are taken into account. As a result, transmission delays are reduced, leading to improved quality of service. In the implemented scenario, data traffic is injected into the specified SDN topology links, distributing traffic across all bandwidths. Figure 7 presents the network performance results for varying numbers of controllers.

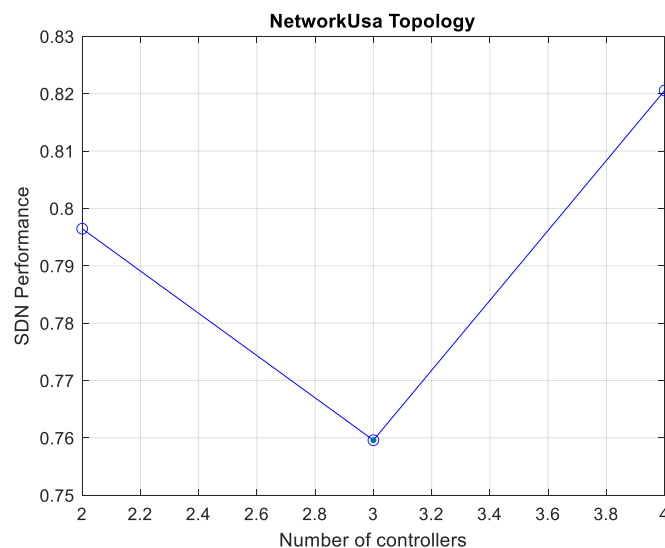


Figure 7: Network Performance After Optimal Controller Placement

Table 2 presents a comparison of the processing times of algorithms reported by Ho et al. (2019) [19]. As observed, the Whale Optimization Algorithm used in this study requires more execution time compared to other algorithms. This is because the algorithm performs numerous iterations to ultimately select the optimal solution. Additionally, the algorithm includes a path evaluation step after identifying the preferred routes, which is not addressed in the other algorithms. In essence, the proposed algorithm demands more computations and increases computational complexity at the controller level. However, with enhanced computational

capacity of hardware devices and the use of multiple distributed controllers in the SDN, the computational burden on individual controllers is reduced.

Table 3: Average Processing Time of Different Algorithms

Algorithm	MOPSO	Lee	WSP	Mira	SPF	WOA
Execution Time (seconds)	15.56	17.16	23.32	24.12	21.89	12.47

Additionally, based on the study by Ataya et al. (2019) [20], which addresses controller placement in SDN networks using a Particle Swarm Optimization (PSO) algorithm, several sample topologies from the “Internet Zoo Topology” database—listed in Table 3—were utilized to compare the performance of the proposed method with that of the approach presented in the referenced study.

Table 4: Sample Network Topologies

Topology	Number of Nodes
ARPANET 1969_12	4
MREN	6
GetNet	7
Sprint	11
NSF	13
Claranet	15
IBM	18
Oxford	20
FCCN	23
AGIS	25

In this analysis, the delay threshold was set to 2 milliseconds ($\tau=2$ ms), and the optimal number of controllers was determined for each topology. The comparison of both methods is presented in Figure 8.

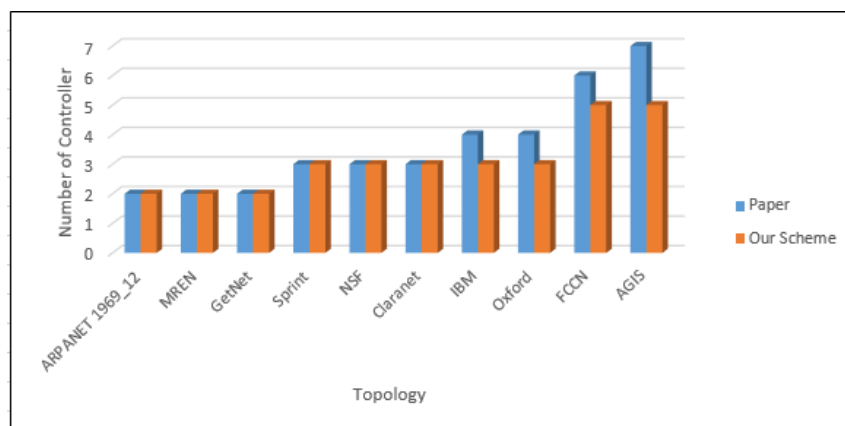


Figure 8: Controller Selection Comparison with Ataya et al. (2019) [20] Considering a Delay Threshold of 2 ms

As shown in Figure 8, the proposed method in this study, based on the Whale Optimization Algorithm, demonstrates superior performance under identical conditions and for larger

network topologies—such as IBM, Oxford, FCCN, and SGIS. By optimally placing the controllers, the method effectively manages network load and prevents the unnecessary increase in the number of controllers. Next, the impact of increasing the delay threshold is examined under the same request load conditions as before ($\lambda \in [1500, 3000]$). In this scenario, the response delay threshold for requests is doubled to 4 milliseconds. The results of this evaluation are presented in Figure 9.

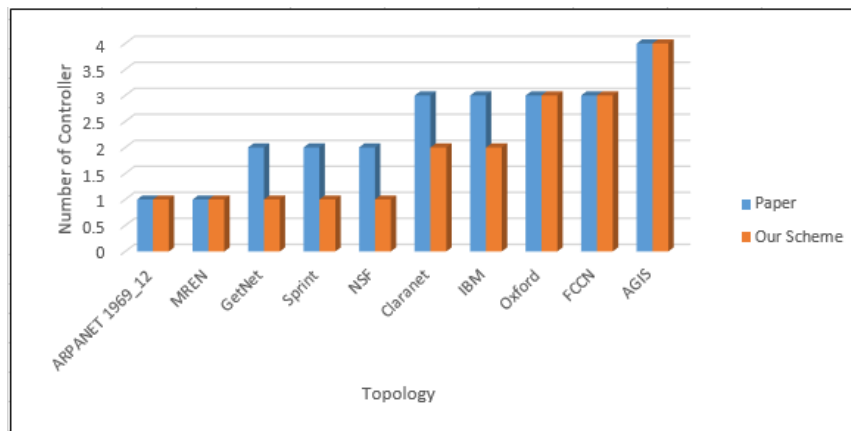


Figure 9: Controller Selection Comparison with Ataya et al. (2019) [20] Considering a Delay Threshold of 4 ms

As shown in Figure 9, the Whale Optimization Algorithm demonstrates superior performance in determining optimal controller placement and, consequently, in selecting the appropriate number of controllers compared to the Particle Swarm Optimization algorithm proposed by Ataya et al. (2019) [20]. Another observation from this comparison is that increasing the delay threshold reduces the number of controllers relative to the previous scenario. This indicates that a higher delay threshold allows each controller to manage more devices, as the network QoS remains controlled despite the increased response time.

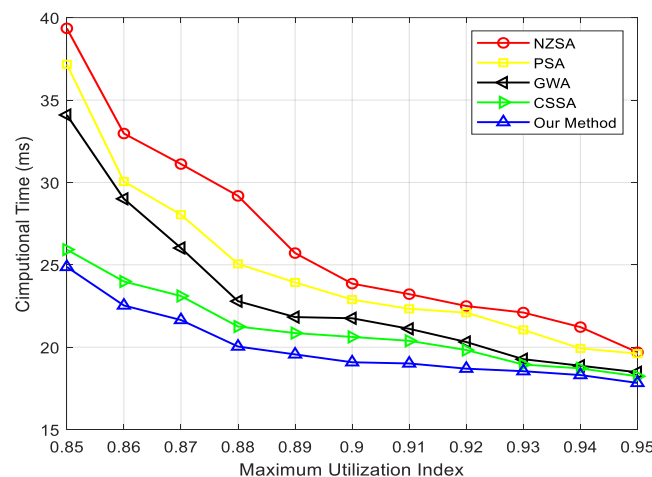


Figure 10: Computational Time Compared to Link Utilization Efficiency

Additionally, Figure 10 presents a comparison of the proposed algorithm's results against four algorithms reported in Ataya et al. (2019) [20]. The results indicate that the proposed method achieves the best link utilization efficiency across all evaluated scenarios. Another important finding is that the other algorithms sometimes fail to reach the global optimum due to local convergence, a limitation addressed by employing the Whale Optimization Algorithm in this study.

DISCUSSION AND CONCLUSION

The simulation results indicate that the proposed algorithm outperforms other algorithms in terms of key metrics, including authentication key breaking time, network resource consumption, load balancing, delay reduction, and maximum link utilization. Furthermore, the proposed algorithm can select the optimal path based on user priority, providing a comprehensive performance evaluation metric. Analysis of the results shows that the algorithm performs particularly well as the network topologies from the “Internet Zoo Topology” become larger. By placing controllers at optimal locations, the algorithm reduces the number of controllers required in the network. The factors influencing the number of controllers include the allowable delay threshold for responding to requests and the volume of incoming requests. To maintain network QoS, the number of controllers must be chosen to efficiently monitor and manage network load. Security is also addressed through authentication and key verification between devices, protecting the network at the source, along the transmission path, and at the destination. By reducing the transmission path and minimizing delays, the proposed approach also contributes to enhanced network security. Initially, the Whale Optimization Algorithm and routing were used to optimize link weights, selecting the shortest path among all available paths. Finally, the path with the highest comprehensive evaluation value was chosen.

One limitation of the current study is that service quality is ensured solely through routing management and mapping flows to existing paths. To improve this, a queuing mechanism could be incorporated, giving reserved tunnels for multimedia traffic higher priority in switch queues. However, this raises questions about how to adapt the cost formulation for paths under these new conditions and how to redistribute network load. Additionally, the proposed method does not include any control over incoming flows into the network.

REFERENCES

1. Sayeed MA, Kumar R, Sharma V. Efficient data management and control over WSNs using SDN-enabled aerial networks. *International Journal of Communication Systems*. 2020 Jan 10;33(1): e4170.
2. Ouhab A, Abreu T, Slimani H, Mellouk A. Energy-efficient clustering and routing algorithm for large-scale SDN-based IoT monitoring. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)* 2020 Jun 7 (pp. 1-6). IEEE.
3. B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.
4. I. T. Haque and N. Abu-Ghazaleh, “Wireless software defined networking: A survey and taxonomy,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 2016.
5. H. Mostafaei and M. Menth, “Software-defined wireless sensor networks: A survey,” *Journal of Network and Computer Applications*, vol.119, pp. 42–56, 2018.
6. Liu Q, Ozcelebi T, Cheng L, Kuipers F, Lukkien J. Cluflow: Cluster-based flow management in software-defined wireless sensor networks. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)* 2019 Apr 15 (pp. 1-8). IEEE.
7. "Ryu SDN Framework," [Online]. Available: <https://osrg.github.io/ryu/>
8. Buzachis A, Galletta A, Celesti A, Villari M. An innovative MapReduce-based approach of Dijkstra's algorithm for SDN routing in hybrid cloud, edge and IoT scenarios. *Lect Notes Comput Sci*. 2018; 11116:185-198.
9. Wang J, Miao Y, Zhou P, Hossain MS, Rahman SMM. A software defined network routing in wireless multihop network. *J Netw Comput Appl*. 2017; 85:76-83.

10. Shah SBH, Chen Z, Yin F, Ahmad A. Water rippling shaped clustering strategy for efficient performance of software define wireless sensor networks. *Peer-to-Peer Netw Appl.* 2017;12(2):371-380.
11. Letswamotse BB, Malekian R, Chen C-Y, Modieginyane KM. Software defined wireless sensor networks and efficient congestion control. *IET Networks.* 2018;7(6):460-464.
12. Kumar N, Vidyarthi DP. A green routing algorithm for IoT enabled software defined wireless sensor network. *IEEE Sens J.* 2018; 18:9449-9460.
13. Al-Janabi TA, Al-Raweshidy HS. A centralised routing protocol with a scheduled mobile sink-based AI for large scale I-IoT. *IEEE Sens J.* 2018; 18:10248-10261.
14. Sayeed MA, Kumar R, Sharma V. Efficient data management and control over WSNs using SDN-enabled aerial networks. *International Journal of Communication Systems.* 2020 Jan 10;33(1): e4170.
15. Vishnu VM, Manjunath P. SeC-SDWSN: Secure cluster-based SDWSN environment for QoS guaranteed routing in three-tier architecture. *International Journal of Communication Systems.* 2019 Sep 25;32(14): e4020.
16. Cumino P, Maciel K, Tavares T, Oliveira H, Rosário D, Cerqueira E. Cluster-Based Control Plane Messages Management in Software-Defined Flying Ad-Hoc Network. *Sensors.* 2020 Jan;20(1):67.
17. Jadhav, A. R., & Shankar, T. (2017). Whale Optimization Based Energy-Efficient Cluster Head Selection Algorithm for Wireless Sensor Networks. *arXiv preprint arXiv:1711.09389.*
18. Regencia, J. E. T., & Yu, W. E. S. (2021). Latency and throughput advantage of leaf-enforced quality of service in software-defined networking for large traffic flows. In *Intelligent Computing* (pp. 606-623). Springer, Cham.
19. Hou, X., Wu, M., & Zhao, M. (2019). An optimization routing algorithm based on segment routing in software-defined networks. *Sensors, 19*(1), 49.
20. Ateya, A. A., Muthanna, A., Vybornova, A., Algarni, A. D., Abuarqoub, A., Koucheryavy, Y., & Koucheryavy, A. (2019). Chaotic salp swarm algorithm for SDN multi-controller networks. *Engineering Science and Technology, an International Journal, 22*(4), 1001-1012.