A Greedy Approach for Controller Placement in Software-Defined Networks for Multiple Controllers

Stavroula Lalou
Department of Digital Systems
University of Piraeus
Piraeus, Greece
slalou@unipi.gr

Georgios Spathoulas
Dept. of Inform. Sec. and Comm. Techn.
NTNU
Gjøvik, Norway
georgios.spathoulas@ntnu.no

Sokratis Katsikas
Dept. of Inform. Sec. and Comm. Techn.
NTNU
Gjøvik, Norway
sokratis.katsikas@ntnu.no

***Abstract*—The Controller Placement Problem (CPP) addresses the strategic positioning of Software Defined Networking (SDN) controllers within a network, crucial for efficient network management. It impacts network performance in latency, reliability, scalability, and resource usage. SDN architecture, separating control and data planes, enhances scalability and programmability compared to traditional architectures. Determining the optimal number and placement of controllers is a key challenge, with network latency being a primary performance factor. This study proposes a heuristic greedy algorithm to minimize end-to-end latency and reduce maximum latency between controllers and switches, aiming to mitigate controller queuing delay. Ultimately, deploying controllers in SDN-wide networks seeks to minimize maximum latency between controllers and switches.**

Keywords-Software Defined Networks, multiple controllers, contoller placement, latency.

I. Introduction

The rise of Software-Defined Networking (SDN) has been prompted by the escalating demands for new networks and the expansion of Internet coverage. As contemporary requirements surpass the limitations of traditional networks, SDN emerges as a promising paradigm. It addresses these challenges by separating the control and data planes, enabling the programmability of network configuration. The pivotal factor influencing SDN deployment and applications is the strategic placement of controllers. Within the SDN architecture, the use of single or multiple controllers is instrumental in achieving programmable, flexible, and scalable configurations. In the current SDN landscape, the employment of multiple controllers has become imperative. Recent developments have introduced various solutions aimed at enhancing scalability and optimizing controller placement selection. This study delves into the CPP by leveraging objective optimization with proposed algorithms.

Interaction time between the controller and the switch is an important parameter in locating the controllers. The proposed approach is a greedy algorithm for controller placement in a network. The algorithm aims to minimize the total cost of connecting nodes to controllers.

SDN is a network architecture that separates the control plane from the data plane. The control plane is responsible for making decisions and configuring the network, while the data plane is responsible for forwarding network traffic.

Multiple controllers are often used in SDN networks to distribute the control plane workload and improve network scalability. However, in a multi-controller SDN network, additional delays can be introduced due to communication between controllers.

Transmission delay between controllers can cause delays in network decision making, leading to reduced network performance and efficiency. In a multi-controller SDN network, each controller must have up-to-date information about the network topology and state, which is communicated through inter-controller messaging. Excessive transmission delay between controllers and switches can lead to increased packet delay, jitter, and even packet loss, leading to reduced network performance and efficiency.

The study aims to contribute to the understanding of relevant open problems in the realm of controller placement. By utilizing objective optimization algorithms, the research seeks to propose effective solutions that address the challenges associated with placing controllers optimally in SDN environments. As the field of SDN continues to evolve, identifying and addressing these challenges becomes crucial for the advancement and sustainability of programmable and scalable network configurations.

The contributions of this paper are:

- An effective solution that addresses the challenges associated with placing controllers optimally in SDN environments with multiple controllers.
- An implementation with greedy algorithm that finds the location that is closest to the most switches that do not have assigned controllers. The algorithm places a controller in that location, assigns each switch within the controller's coverage to that controller and calculates the latency between each switch and its assigned controller.
- A heuristic greedy algorithm designed to address the Controller Placement Problem. It aims to minimize end-to-end latency and reduce maximum latency between controllers and switches. By proposing a specific algorithmic approach, the study contributes to the practical implementation of controller placement strategies within SDN environments.

The remaining of the paper is structured as follows: In Section II, we briefly review necessary background knowledge on CPP in SDN and CPP approaches. In Section III, we discuss related work. In Section IV, we present our heuristic approach for latency. In Section V, we present the experimental setup that we used for evaluating the performance of the proposal and we discuss the results. Finally, Section VI summarizes our conclusions.

II. Background

A. The Controller Placement Problem (CPP)

The CPP [1] in Software-Defined Networking (SDN) involves strategically deploying controllers within the network, impacting various performance metrics like availability, fault tolerance, and convergence time. SDN, with its separated control and data planes, offers solutions to network challenges, making CPP a critical factor for optimizing network performance. Early research by Heller et al. [1] framed CPP as a facility placement problem, highlighting its complexity. Subsequent studies focused on minimizing propagation delay by determining optimal controller locations and quantities. Techniques like K-center and Multiobjective Optimization Controller Placement (MOCP) [8] address different aspects such as network reliability, load balance, and latency. CPP requires careful planning considering factors like network topology, latency, and controller placement to ensure optimal performance and resilience. Our approach offers a heuristic implementation of CPP, aiming to minimize the number of controllers while addressing latency concerns, ultimately enhancing network responsiveness and robustness in the face of failures.

B. General formulation of CPP

The primary network components for an SDN-enabled network are switches, controllers, and the links that join them. Therefore, the network is often modeled as an undirected graph

$$G = (SEC) \tag{1}$$

where S represents the set of switches and E is the set of physical links among those switches or controllers, and C be the controllers to be placed in the network. The switches and the controllers are represented as follows: S = s1, s2, s3, ... sn, where n denotes the number of switches in the network, and C = c1, c2, c3, ... ck, where k denotes the number of controllers located in the network. Here, Pi = p1, p2, p3, ... pk is one possible placement of the k controllers. The relationship between switches and controllers is represented by the condition that the set of controllers C is a subset of the set of switches S, denoted by $C \subseteq S$. This indicates that controllers are located within the network's switches. The shortest path between a switch $s \in S$ and a controller $c \in C$ indicates the minimum number of links (or hops) required to reach the controller from the switch [2].

C. Multiple controller placement problem

The challenge of deploying multiple controllers arises from the limitations of a single centralized controller, which struggles to keep pace with the growing demands of expanding networks and applications. Relying on a solitary controller creates a potential single-point bottleneck and failure risk, as it shoulders the entirety of control activities. Consequently, any network failure could severely impact overall network performance. Thus, the adoption of a multi-controller approach emerges as a viable solution for large-scale Software-Defined Networking (SDN), particularly concerning the scalability of the control plane. The inadequacy of deploying a single controller for managing extensive networks, advocating instead for the deployment of multiple controllers. However, effectively placing multiple controllers remains a complex task. To optimize network scalability and minimize latency, particularly in larger networks, leveraging multiple controllers to manage control plane traffic is deemed most effective. Two prevalent architectures for multi-controller setups are flat architecture and hierarchical architecture [20]. Deploying multiple controllers in a large-scale SDN environment aims to reduce latency, distribute controller workload, and optimize various network performance indicators related to controller placement. Consequently, an SDN controller can oversee multiple Network Operating Systems (NOS) [19]. While a single controller suffices for small networks like those in data centers, the adoption of multi-controller deployment is increasingly favored to bolster the scalability and stability of Wide Area Networks (WANs). Dhar et al. [21] underscore the necessity of deploying multiple controllers to maintain scalability and reliability in large SDN setups.

III. Related Work

Latency holds significant importance in Software-Defined Networking (SDN) due to the frequent interaction between switches and controllers. Existing solutions for the CPP typically prioritize minimizing both propagation delay and controller processing delay. Regarding propagation delay, CPP resembles the facility location problem.

Heller et al. [1] were pioneers in CPP research, aiming to minimize worst-case delay while proving its NP-Hard complexity. Since then, numerous researchers have proposed diverse CPP solutions, extending optimization objectives from original switch-controller delay to inter-controller delay, controller capacity, and cost considerations.

Zhu et al. [3] focus on minimizing propagation delay between switches and controllers, formulating CPP as a control plane delay minimization problem and introducing a new algorithm based on clustering and Dijkstra's algorithm. For controllers with limited capacities, Yao et al. [5] define a capacitated CPP and develop an efficient algorithm for optimizing propagation delay.

In another study [6], the authors address controller placement under dynamic network traffic by integrating the controller module placement algorithm with a dynamic flow management algorithm, albeit suitable for small-scale networks only.

Tanha et al. [7] concentrate on CPP within Software Defined Wide Area Network (SDWAN), introducing a clique-based approach from graph theory for polynomial-time solution derivation. In subsequent works [2], [8] CPP is formulated considering switch-to-controller delay, controller-to-controller delay, load balancing, and link utilization rate as objectives.

Wang et al. [9] identify that a critical difficulty in SDN is selecting suitable locations for controllers to reduce the latency between controllers and switches. The CPP described a few of the performance factors that were taken into consideration, including control plane overhead, latency, load imbalance, cost, and connection. They use the controller-to-node latency (propagation, queuing, and processing delay) as a crucial performance parameter.

Mamushiane et al. [10] extended and used a facility location approach known as Partition Around Medoids (PAM) with propagation latency to determine the optimum places to put SDN controllers. They suggested using the Silhouette and Gap Statistics algorithms to decide how many controllers to deploy in a wide-area network.

Rasol et al. [11] assess the Joint Latency and Reliability-aware Controller Placement (LRCP) optimization model. With the help of alternate backup channels, LRCP gives network administrators a variety of options for balancing the reliability and latency trade-offs between controllers and switches. This study suggests the Control Plan Latency (CPL) metric, the sum of average switch-to-controller latency and the average inter-controller latency, in order to evaluate the controller placements offered by LRCP and determine how effective they are in an actual controller deployment.

In each link failure state, Fan et al. [13] further take into account the number of control path reroutings and the worst-case latency between the controller and the switch. To solve the problem, they offer a heuristic approach based on particle swarm optimization. The suggested algorithm's usefulness is demonstrated by the numerical results. Additionally, it demonstrates that in the majority of link failure conditions, the suggested technique may ensure the latency and reliability of the control layer.

Fan et al. [14] present the Resilient Controller Placement (RCP) algorithm. The objective of this approach is to to minimize the average latency between all switches and the appropriate controllers in the event of a single broken link. The latency of each path is made up of the latency of the primary path plus the average of any potential backup paths that might be available in the event of a single link failure.

Chen et al. [14] present that the network is separated into many sub-networks, and the essential performance metric is the latency between the controller and switch.

Liao et al. [17] can be used to determine the latency model in this study or a reliable CPP, Singh et al [22]. suggest a Varna Based Optimization (VBO) to guarantee that it reduces the overall average latency. Their results demonstrate that the proposed VBO algorithm outperforms other effective heuristic algorithms for the Reliability-aware CPP (RCPP), such the Particle Swarm Optimization PSO. PSO and Teachning Learning-Based Optimization (TLBO) and their experimental results show that TLBO performs better than PSO for publicly accessible topologies.

In [18], reducing network latency between controllers and switches is crucial for SDN performance. The study introduces a Controller Node Partitioning Algorithm (CNPA) to minimize end-to-end latency. By partitioning the network and deploying controllers strategically, the aim is to decrease latency between controllers and switches in SDN-enabled wide-area networks.

IV. Our proposal

Our research focuses on latency which is one of the most often used performance indicators. Transmission, propagation, queuing, and processing delay make up the total latency. We evaluate latency between switch to controller latency (also known as controller-node latency) and controller-controller latency.

The proposed algorithm is a greedy algorithm for placing controllers near switches to minimize the latency between controllers and nodes. The algorithm calculates the latency between each controller and its assigned nodes. The latency is calculated as the Euclidean distance between the controller and node, plus the transmission delay. The algorithm assigns each node to the controller with the lowest latency.

The algorithm starts by initializing a costs, where each element costs[i][j] represents the cost of connecting switch i to controller j. The costs are calculated by computing the Euclidean distance between the switch and the controller. The algorithm then initializes an assigned array, where each element assigned[i] is a list of nodes assigned to controller i. The unassigned nodes are stored in a list called unassigned. The algorithm then enters a loop, where it repeatedly selects an unassigned node and assigns it to the closest available controller. The closest controller is determined by finding the controller with the smallest cost to the node's switch. If no controller is available, the node is skipped. The algorithm continues until all nodes have been assigned to a controller. The final result is a list of lists, where each inner list contains the nodes assigned to a specific controller.

A. Implementation

We use the controller-node latency and controller-controller (propagation, queuing, and processing delay) as a crucial performance parameter. We have implemented a heuristic approach based on greedy algorithm. The basic idea of this approach is to obtain the minimum number of controllers which minimizes inter nodes distances to obtain acceptable latency from nodes to their assigned controller and also between controllers. Greedy algorithm uses the Euclidean distance between nodes and controllers as the cost function to determine the best immediate solution. In the context of the controller placement problem, the greedy algorithm calculates the Euclidean distance between each unassigned node and each available controller, and assigns the node to the controller with the smallest distance. Then, the distance between switches and controllers is calculated and the nodes are allocated to the closest controller.

If the controller does not have the capacity to handle the node, the algorithm searches for the following nearest controller and carry out the same operation. This process will continue until an controller found and allocate rest of the node to the controller. The greedy solution provide by the algorithm fails when no controller can accommodate the required capacity, that case is considered as the worse case.

A greedy heuristic algorithm is a type of algorithm that makes the locally optimal choice at each stage with the hope of finding a global optimum. It is a simple and fast algorithm. In the context of the CPP in multiple software-defined networking, a greedy heuristic algorithm can be used to find a solution that minimizes the latency between controllers and switches. The algorithm would iteratively place controllers in locations that minimize the maximum distance to any switch, without considering the effect on future decisions. This approach is simple and fast, but it may not always find the optimal solution, and it may lead to higher latency between some controllers and switches. Here are the steps of the proposed approach for the CPP in SDN:

- Initialize a list of available locations for controllers.
- While there are still switches without assigned controllers:
  a. Find the location that is closest to the most switches that do not have assigned controllers.
  b. Place a controller in that location.
  c. Assign each switch within the controller's coverage to that controller.
- Calculate the latency between each switch and its assigned controller.

In our approach, switches is a list of (x, y) coordinates representing the locations of the switches, controllers is a list of (x, y) coordinates representing the locations of the controllers, and nodes is a list of nodes to be assigned to controllers. Each node has a switch attribute indicating which switch it is connected to.

The greedy controller placement function first calculates the Euclidean distance between each switch and controller and stores the distances. It then initializes a list assigned to keep track of which nodes have been assigned to which controllers.

The function then permutes the list of unassigned nodes and assigns each node to the controller with the lowest cost. The function returns the assigned list, which indicates which nodes have been assigned to which controllers.

This algorithm initializes a list of available locations for controllers, and then enters a while loop as long as there are still switches without assigned controllers. In each iteration, it finds the location that is closest to the most switches that do not have assigned controllers, places a controller in that location, and assigns each switch within the controller's coverage to that controller. It also calculates the latency between each switch and its assigned controller.

The algorithm uses a vector of Location structs to represent the available locations for controllers. Each Location struct contains the index of the location, the distance to the nearest unassigned switch, and a vector of unassigned switches that are within the controller's coverage.

The algorithm sorts the locations based on their distance and the number of unassigned switches within their coverage. It then iterates over the sorted locations and assigns a controller to each location that has not been assigned yet.

The algorithm returns a vector of Controller structs, where each Controller struct contains the index of the controller and the number of assigned switches.

The switch distances list represents the distances from each switch to a central location, and the controller coverage variable represents the coverage radius of each controller. The function returns a list of controller locations and a modified switch distances list that contains the latency between each switch and its assigned controller.

The function works by iterative placing controllers in the location that is closest to the most unassigned switches, and then assigning all switches within the controller's coverage to that controller. The function continues to place controllers until all switches have been assigned to a controller.

V. PERFORMANCE EVALUATION

A. Experimental Setup

A simulation has been conducted to assess the performance of the proposed scheme. The system on which the simulation was executed was based on a Virtual Machine (VM) with Ubuntu 22.04 OS, 16 GB of memory and OpenFlow Switches. We emulate the performance using Mininet and Ryu controller [22] component-based software defined networking framework.

We evaluated the performance of our system in terms of latency and transmission delay. These factors are crucial in Software-Defined Networking (SDN) due to the frequent communication between switches and controllers. The Controller Placement Problem in SDN often focuses on minimizing both transmission delay and controller processing delay. Transmission delay in CPP is similar to the facility location problem, where the goal is to find the best location to place the controllers to reduce the distance between the switches and the controllers. This is an important factor in ensuring efficient communication and reducing latency in SDN networks.

B. Results

We have created a network of 6 controllers and 8 nodes (switches). We calculated the latency between controllers to nodes and between controllers.

Controller- Node latency is the time it takes for a message to travel from a controller to a node in a network. Transmission delay, also known as latency, is the time it takes for a message to be transmitted over the physical link between the controller and the node. Figure 1 depicts the latency between each controller and its assigned nodes. The latency is calculated in seconds. The total transmission latency between controllers to nodes is 10.067 seconds, it is the sum of all the latencies between each controller and its assigned nodes as described in Table I. The transmission delay between controllers is 1.414 seconds and is lower than the controller-controller latencies. The transmission latency as shown in Figure 1 includes the

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

time it takes for the switch to receive the data from the sender, process it, and send it to the receiver.

The latency between two controllers is calculated as the Euclidean distance between the two controllers and the transmission delay because the signal has to travel from one controller to the other, and then back to the first controller. It is depicted in Figure 2. The transmission delay is also counted for the round-trip time. The Euclidean distance between two controllers increases as the controllers are placed further apart, so the latency between two controllers will also increase as they are placed further apart.

The nodes are placed at positions (3, 3), (4, 4), (3, 4), (3, 5), (4, 3), (4, 4), (5, 3), and (5, 4). The Euclidean distance between between Controller 1 and Node 1 is math.sqrt((0 - 3) ** 2 + (5 - 3) ** 2) = 3.61 units, so the latency between Controller 1 and Node 1 is 3.61.

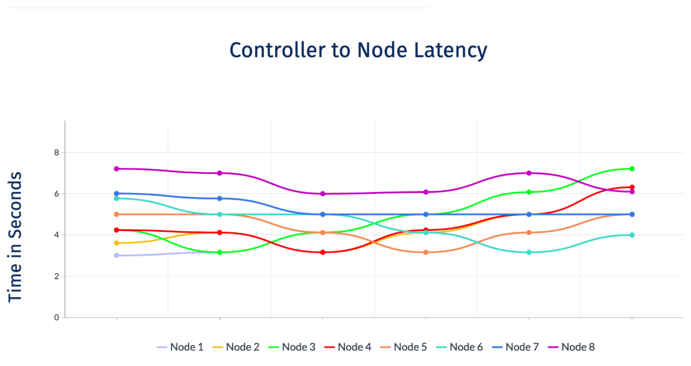

Figure 1. Controllers to Node Latency.

Controllers are placed at positions (0, 5), (10, 5), (20, 5), (30, 5), (40, 5), and (50, 5). The Euclidean distance between two adjacent controllers is 10 units, so the latency between two adjacent controllers is 10 + 2 * transmission delay.

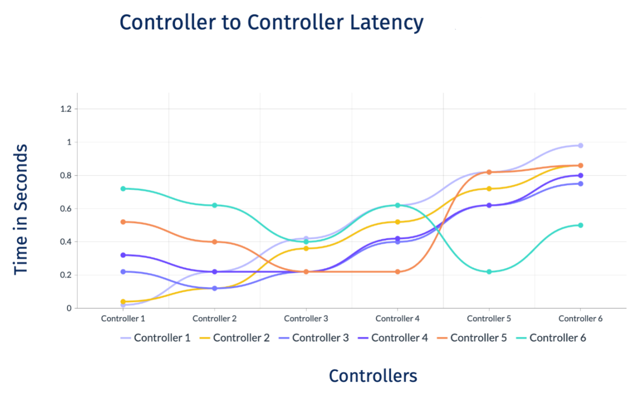

Figure 2. Controller to Controller Latency.

In our implementation, we use a nested loop to iterate over each node and controller, and find the minimum latency between the node and the controller. We then sum up these minimum latencies to get the total transmission delay. The controller with the minimum latency is assigned to the node. Finally, the total transmission delay is calculated by summing up the minimum latencies between each node and its assigned controller.

Table I shows the latency between each controller and the nodes assigned to it. The first controller has a latency of 0.21 seconds for the first node and 0.42 seconds for the second node.

TABLE I
CONTROLLER-NODE TRANSMISSION DELAY IN SECONDS

Controllers	**Node 1**	**Node 2**	**Node 3**	**Node 4**	**Node 5**	**Node 6**	**Node 7**	**Node 8**
Controller 1	0.21	0.42	0.44	0.42	0.43	0.44	0.46	0.43
Controller 2	0.48	0.21	0.33	0.33	0.30	0.33	0.36	0.34
Controller 3	0.59	0.43	0.21	0.22	0.22	0.21	0.22	0.22
Controller 4	0.57	0.46	0.32	0.21	0.21	0.22	0.22	0.21
Controller 5	0.53	0.41	0.32	0.33	0.21	0.22	0.22	0.21
Controller 6	0.63	0.48	0.36	0.35	0.34	0.21	0.22	0.21

Table II calculates the average controller latency. The delay is calculated as the sum of the transmission delays. According to the results, in an SDN network with multiple controllers, the latency numbers are low.

TABLE II
AVERAGE TRANSMISSION DELAY PER CONTROLLER

Controllers	**Average Transmission Delay per Controller**
	Average Latency
Controller 1	0.122 Seconds
Controller 2	0.118 Seconds
Controller 3	0.133 Seconds
Controller 4	0.132 Seconds
Controller 5	0.124 Seconds
Controller 6	0.115 Seconds

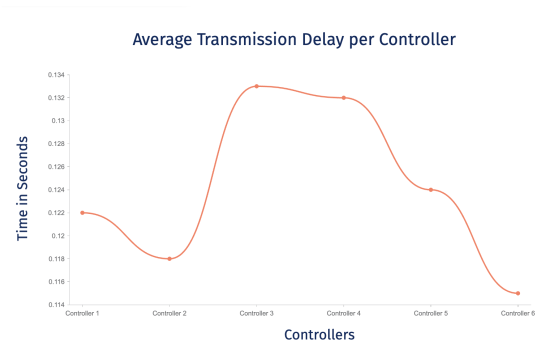

Figure 3. Transmission Delay per Controller.

The latency numbers represent the time it takes for a controller to send a message to another controller. The latency numbers are calculated as the average of the time it takes for the sender to send the message and the time it takes for the receiver to receive the message.

The Controller-Controller latencies show the latency between each pair of controllers. The latencies are relatively small and the switch is able to transmit data quickly between them. The controller-controller latencies represented in Figure 2 are the time it takes for the switch to process the data.

These latencies are lower than the total transmission latencies because the switch only needs to process the data once, not for each controller sending a message to another controller.

Transmission delay per controller is illustrated in Figure 3. The transmission delay per controller is obtaining by dividing the total transmission delay by the number of controllers. The transmission delay per controller is calculated by taking the average of the latencies for each controller, which can be done by summing up the latencies for each controller and dividing by the number of nodes.

The total transmission latency between controllers and nodes is higher than the controller-controller latencies because the switch needs to perform additional processing tasks. It includes the time it takes for the switch to receive the data from the sender, process it, and send it to the receiver.

VI. Conclusion

The idea of CPP in SDN is to adapt the facility location problem concepts to find the best location to place the controllers in the network, in order to reduce the transmission delay and improve the overall performance of the network. This implementation presents a greedy algorithm designed to optimize controller placement within a network, with the aim of minimizing latency between controllers to nodes and controllers to controllers and transmission delay.

Experimental results demonstrate the efficiency of the implementation, achieving near-optimal solutions within the CPP framework. The observed latencies between controllers and nodes, as well as between controllers themselves, remain low, affirming the success of the controller placement strategy. Furthermore, our proposal minimizes transmission delay between controllers, which is critical to ensuring high network performance and efficiency. According to our results, our proposal ensures high network performance, scalability, and efficiency by minimizing transmission delay between controllers and between controllers and switches.

Future work could investigate the impact of different network topologies on controller placement. This could provide insights into the algorithm's performance in different network configurations.

Acknowledgment

This work has been partly supported by the University of Piraeus Research Center.

References

[1] B. Heller, R. Sherwood and N. McKeown, ”The controller placement problem”, Proc. of the First Workshop on Hot Topics in Software Defined Networks, HotSDN ’12, ACM, New York, USA, pp. 7–12, 2012, doi:http://doi.acm.org/10.1145/2342441.2342444.

[2] G. Schütz and J. A. Martins, ”A comprehensive approach for optimizing controller placement in Software-Defined Networks”, pp. 199, 2020, doi: https://doi.org/10.1016/j.comcom.2020.05.008.

[3] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, ”A K-means-based network partition algorithm for controller placement in software defined network”, IEEE International Conference on Communications, Kuala Lumpur, pp. 1–6, 2016.

[4] L. Zhu, R. Chai and Q. Chen, ”Control plane delay minimization based SDN controller placement scheme”, Proc. 9th International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, pp. 1–6, 2017.

[5] G. Yao, J. Bi, Y. Li and L. Guo, ”On the capacitated controller placement problem in software defined networks”, IEEE Commun. Lett. 18, pp. 1339–1342, 2014.

[6] M. T . I. ul Huque, W. Si, G. Jourjon and V. Gramoli, ”Large-scale dynamic controller placement”, IEEE Trans. Netw. Serv. Manag. 14, pp. 63–76, 2017.

[7] M. Tanha, D. Sajjadi, R. Ruby and J. Pan, ”Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs”, IEEE Trans. Netw. Serv. Manag. 15, pp. 991–1005, 2018.

[8] B. Zhang, X. Wang and M. Huang, ”Multi-objective optimization controller placement problem in internet-oriented software defined network”, Comput. Commun. 123, pp. 24–35, 2018.

[9] G. Wang, Y. Zhao, J. Huang, and Y. Wu, ”An effective approach to controller placement in software defined wide area networks,” IEEE Trans. Netw. Serv. Manag., pp. 344-355, 2017.

[10] L. Mamushiane, J. Mwangama and A. A. Lysko, ”Controller placement optimization for Software Defined Wide Area Networks (SDWAN)”, pp. 45-66, 2021.

[11] K. A. Rasol and J. Domingo-Pascual, ”Evaluation of joint controller placement for latency and reliability-aware control plane”, Eighth International Conference on Software Defined Systems (SDS) IEEE, pp. 1-7, 2021.

[12] E. Borcoci, R. Badea, S. Georgica Obreja, and M. Vochin, ”On multi-controller placement optimization in software defined networking-based wans”, (ICN 2015), pp. 273, 2015.

[13] Z. Fan et al., ”A multi-controller placement strategy based on delay and reliability optimization in SDN”. Proc. 28th Wireless and Optical Communications Conference (WOCC), IEEE, 2019.

[14] Y. Fan et al., ”Latency-aware reliable controller placements in SDNs”, Proc. Communications and Networking: 11th EAI International Conference, (ChinaCom 2016 Chongqing), pp. 152-162, China, September 24–26, 2016.

[15] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, ”Pareto-optimal resilient controller placement in sdn-based core networks”, Teletraffic Congress (ITC), 25th International IEEE, pp. 1–9, 2013.

[16] W. Chen, C. Chen, X. Jiang and L. Liu, ”Multi-controller placement towards SDN based on Louvain heuristic algorithm”, IEEE Access, 2018.

[17] J. Liao, H. Sun, J. Wang, Q. Qi, K .Li and T. Li ”Density cluster based approach for controller placement problem in large-scale software defined networkings”, 2017.

[18] G. Wang, Y. Zhao, J. Huang and Y. Wu, ”An effective approach to controller placement in software defined wide area networks”, IEEE Trans Netw Serv Manag, pp. 344-355, 2017.

[19] T. Hu, Z. Guo, P. Yi, T. Baker and J. Lan, ”Multi-controller based software-defined networking: a survey”, IEEE Access, 2018.

[20] M. Dhar, A. Debnath, B. K. Bhattacharyya, M. K. Debbarma and S. Debbarma, ”A comprehensive study of different objectives and solutions of controller placement problem in software-defined networks”, Trans. Emerg. Telecommun. Technol., pp. 33, 2022.

[21] ”Ryu omponent-based software”, [Online]. Available from: ”https://ryu-sdn.org/”, (Retrieved May 2024).

[22] K. Singh, Saurabh, S. Srivastava,”Varna-based Optimization: A New Method for Solving Global Optimization”, International Journal of Intelligent Systems and Applications, pp. 1-15, 2018.