International Journal of Advances in
Soft Computing and Intelligent Systems (IJASCIS)
2023, Vol 02, Issue 02, 77-85

Science Transactions © 2023

Original Paper

A STUDY OF LATENCY AWARE CONTROLLER PLACEMENT PROBLEM IN SDN

Mili Dhar[a,*], Saikat Majumder[b]

[a] School of Computing Science and Engineering, Galgotias University, Greater Noida, India, milidhar28@gmail.com

[b] Indian Institution of Technology Kanpur, Uttar Pradesh, India, majumdersaikat19015@gmail.com

ABSTRACT

In fast and rapidly growing technology where data handling is more important, a new promising paradigm has come into the picture called Software-Defined Network (SDN). A SDN is a software-based programmable network that detaches the control and data plane to overcome the shortcomings of traditional networks. This separation provides many advantages like network virtualization, flexibility, management, and so on. Apart from the advantages given by the SDN, it also brings some issues. The controller placement problem (CPP) is one of them. Depending on the controller location network performances can vary. Putting a controller in any of the accessible locations is also not a good idea, as it only increases the overhead delay. Hence, selecting an appropriate location to shorten the latency is a challenging task. Thus, in this article, we talk about the importance of latency in SDN and study some of the latency-aware techniques developed by other researchers to solve the CPP. These solutions have been divided into two categories according to the application scenarios which are data-center networks and Wide Area Networks (WANs). We've categorized latencies in the control plane and presented their mathematical formulation. We have also presented a comprehensive study in this review. Lastly, we outlined potential areas for future research that researchers can delve into further.

KEYWORDS

Software Defined Networking, Controller Placement Problem, Latency

1. INTRODUCTION

The concept of a software-defined network presents a fresh and promising approach aimed at tackling the obstacles encountered by conventional networks. SDN is a programmable network whose main idea is to decouple the data and control plane. This decoupled architecture offers various advantages for network flexibility and management. Traditional networks are very hard to manage and time-consuming to update/change the rules of networking devices for the administrators whereas SDN provides reliability, flexibility, and programmability [1]. Network administrators can easily write their protocols, and update or change the policies by using the common programming languages, no need to change the data plane or enhance the devices in the data plane. The functionalities of the data plane involve forwarding packets to their destination. The control functionalities of SDN are centralized into one or multiple controllers that are responsible for taking the routing decisions for the forwarding devices (switches, routers). Thus, the controller(s) acts as the brain of the network that also maintains the global network view [2]. Each switch forwards the packets by making a match of the IP header in the routing table. If there is a miss to look into the destination IP header, the switch will ask controller what to do with the newly arrived packet otherwise the switch will directly deliver the packet to its destination without controller intervention. When forwarding devices do not have any idea what to do with the packet, it will communicate with the controller. This switch to controller communication is done through an open interface. The most commonly used interface is known as OpenFlow [3].

Therefore, latency is an important factor in SDN for its decoupled architecture. Selecting a controller location far from the switches will increase the latency which affects network performance. It is enough to control a small-sized network by a single controller whereas multiple controllers are needed for large-size networks or Wide Area Networks (WANs). Thus, it is important to give more attention to finding the required number of controllers and their locations. The challenge of determining both the count and positioning of controllers is widely referred to as the Controller Placement Problem (CPP) in SDN.

Figure 1 shows the basic architecture of multi-controller SDN. It has three layers which are the data plane, control plane, and application layer.

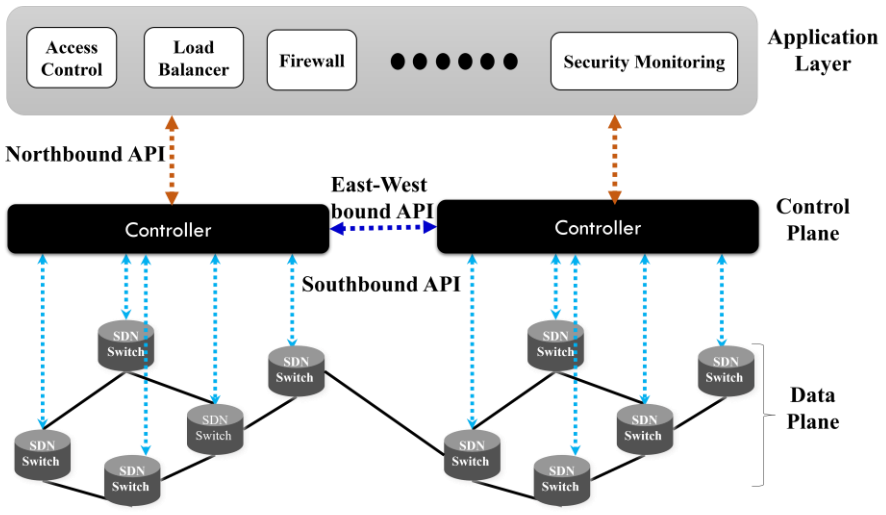


Figure 1. Multi-controller SDN architecture

The Data plane consists of those devices that are responsible for forwarding packets to their destination. Multiple controllers are employed in the control plane to manage the data plane by generating routing rules and policies for the forwarding devices. Network services are run on the application layer. SDN application layer consists of SDN applications (such as load balancer, firewall, and access control) which are nothing but a programmable software implemented on the controller. There are three kinds of application programming interfaces (API) in SDN i.e., northbound, southbound, and east-west bound API. The interface between the SDN application layer and the control plane is called Northbound API. The interface between the data plane and the control plane is known as the southbound API. For multiple controller placement, controllers exchange information with each other using an API called east-west bound API.

1.1. Controller Placement Problem

CPP is one of the primary research issues in SDN caused by the separation of the data and control plane. In this article, we only focused on the wired network topologies of data-centers and WANs. Data-center networks are basically small-size networks that particularly have higher density and limited space. The emergence of Big Data in data-centers arise the requirement for high network scaling and network capacity [4]. On the other hand, WANs interconnect multiple data-centers or local area networks (LANs) over geographically distributed locations. WANs characteristics are large traffic, high link costs, and long distance.

Various research approaches have been proposed to solve the CPP depending on the various objectives, for example latency, reliability, cost, energy, etc. In this article, we only focus on the study of latency-aware solutions proposed by different researchers. There are various types of latencies to be considered to solve the CPP i.e., switch to controller latency L_{SC}, controller to controller latency L_{CC}, processing L_{proc}, queueing L_Q, and transmission latency L_T. L_T is fixed for a particular device, and L_Q is negligible when the network is unobstructed. Thus, most of the solutions only considered L_{SC}, L_{CC}, and L_{proc} where both L_{SC}, L_{CC} are determined by the distances between them, and L_{proc} depends on the load and capacity of the controller.

1.1.1 Switch to Controller Latency

There are two types of latency between L_{SC}. These are average L_{SC} and maximum L_{SC} which we can see in Figure 2.

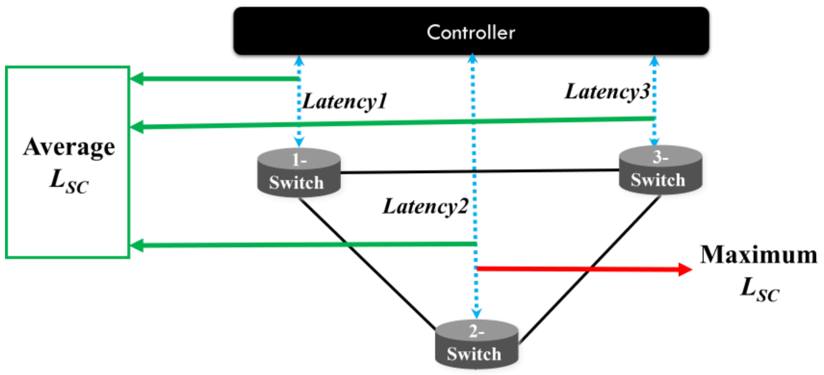


Figure 2. Types of switch to controller latency

Average L_{SC} is the result of the average value of L_{SC} latencies of a network.

Average L_{SC} can be represented mathematically using equation (1) where X_{ij} is a binary decision variable. The value of X_{ij} will be 1 if switch `$i'$ belongs to the `j' controller otherwise the value will be 0.

$$avg_{L_{SC}} = \frac{1}{|S|}\sum_{i=1}^{S}\sum_{j=1}^{C} L_{SC}^{ij} X_{ij} \tag{1}$$

Maximum L_{SC} is the maximum value of L_{SC} that a switch can have to its controller. The objective can be expressed mathematically using equation (2).

$$max_{L_{SC}} = \max_{s\epsilon S} \min_{c\epsilon C} (L_{sc}) \tag{2}$$

1.1.2 Controller to Controller Latency

From the perspective of view consistency of the network, it is required to consider L_{CC} for executing the network applications properly. The intention is to minimize the average L_{CC} which can be expressed using equation (3).

$$avg_{L_{CC}} = \frac{1}{|C|}\sum_{i=1}^{C}\sum_{j=1}^{C} L_{CC}^{ij} \tag{3}$$

1.1.3 Processing Latency

If the controller goes overloaded or the loads exceed the processing capacity of a controller then processing latency may increase. Thus, the best way to minimize this latency is to balance loads of switches among controllers which can be expressed using equation (4).

$$L_{proc} = min\left(\max_{i\epsilon C}(load_i) - \min_{j\epsilon C}(load_j)\right) \tag{4}$$

2. Existing approaches of CPP based on latency

There are basically two categories of existing networks according to the CPP application scenarios proposed by the researchers. These are the data-center network and WANs. As data-center networks are smaller in size and switches are placed closely, in that case, some exhaustive methods are generally used to solve the CPP. But, for large-size networks such as WANs where switches are distributed over geographically cannot be solved by exhaustive methods. Thus, heuristic algorithms have been developed to reduce the latency and to generate the near-optimal solution [5]. We shall describe some of those proposed solutions in detail in later sections.

2.1. CPP solution for Data-center networks

In [6], authors first introduced the CPP in SDN where they have shown the impact of multiple controllers in a network. They used 100 networks and presented the trade-off between the average L_{SC} and the maximum L_{SC}. After analyzing these networks, they have found that only one controller is sufficient for small and some medium-sized networks to handle all the switches and to meet the latency requirements. It is also found that the requirement of controllers by a network is totally topology-dependent.

In [7], the authors proposed an adaptive CPP solution for data-center networks. They used community theory for the selection of a switch placed closer to the switches of a subnet for the controller placement. A controller pool has been built based on network function virtualization that dynamically expands or shrinks because of the aggregate load changes on the controller over time. After that, an overload-avoided method was proposed that adaptively selects the number of controllers depending on the network demand. Their proposed approach minimized the L_{SC}, and balanced the loads among controllers.

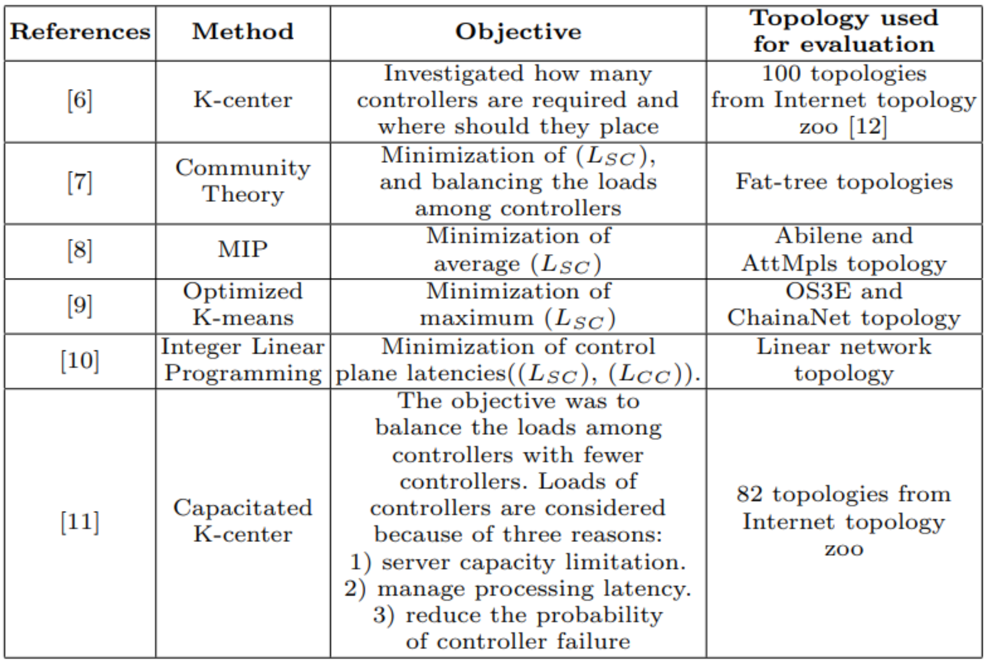
In [8], a CPP solution has been proposed by considering the dynamic traffic flows. The authors developed a combined model where controller locations and the assignments of switches to those controllers have simultaneously optimized to get the minimum average response time for dynamic traffic flows. The problem has been formulated using Mixed Integer Programming (MIP) to generate the optimal result. For comparison, two separate derivatives have been used where one was for only optimizing controller locations and another one was for optimizing switch to controller assignments.

In [9], a clustering method was proposed called the optimized K-means algorithm. Using this method, a network is divided into subnetworks by minimizing the maximum latency of a subnetwork. It focused on minimizing the maximum latency in each subnetwork instead of the whole network that further reduced the complexity of CPP.

In [10], a distributed in-band control plane SDN architecture was considered for investigating the issues of placement problems. The authors mainly focused on the importance of the communication between controllers and in this regard, they proposed two models that were implemented in controllers. The proposed structure considered all the communication of the control plane such as L_{SC} and L_{CC}. The article discussed the Pareto optimal placements and shows the accuracy of the proposed models in software-defined WAN topologies. In addition, a low-complexity algorithm has been proposed for large-size topologies to search for the approximated Pareto frontier.

In [11], the authors first considered the factor of load balancing by looking into the fact that controller overload increases the L_{proc}. Thus, they proposed a capacitated K-center algorithm that reduced the number of controllers to prevent overload while also alleviating the burden on the busiest controller. The resulting solution yields a smaller radius compared to employing dynamic scheduling or a dynamic controller provisioning strategy in K-center placement.

Table 1. A comparative analysis between existing approaches developed for data-centers.



References	Method	Objective	Topology used for evaluation
[6]	K-center	Investigated how many controllers are required and where should they place	100 topologies from Internet topology zoo [12]
[7]	Community Theory	Minimization of (L_{SC}), and balancing the loads among controllers	Fat-tree topologies
[8]	MIP	Minimization of average (L_{SC})	Abilene and AttMpls topology
[9]	Optimized K-means	Minimization of maximum (L_{SC})	OS3E and ChainaNet topology
[10]	Integer Linear Programming	Minimization of control plane latencies((L_{SC}), (L_{CC})).	Linear network topology
[11]	Capacitated K-center	The objective was to balance the loads among controllers with fewer controllers. Loads of controllers are considered because of three reasons: 1) server capacity limitation. 2) manage processing latency. 3) reduce the probability of controller failure	82 topologies from Internet topology zoo

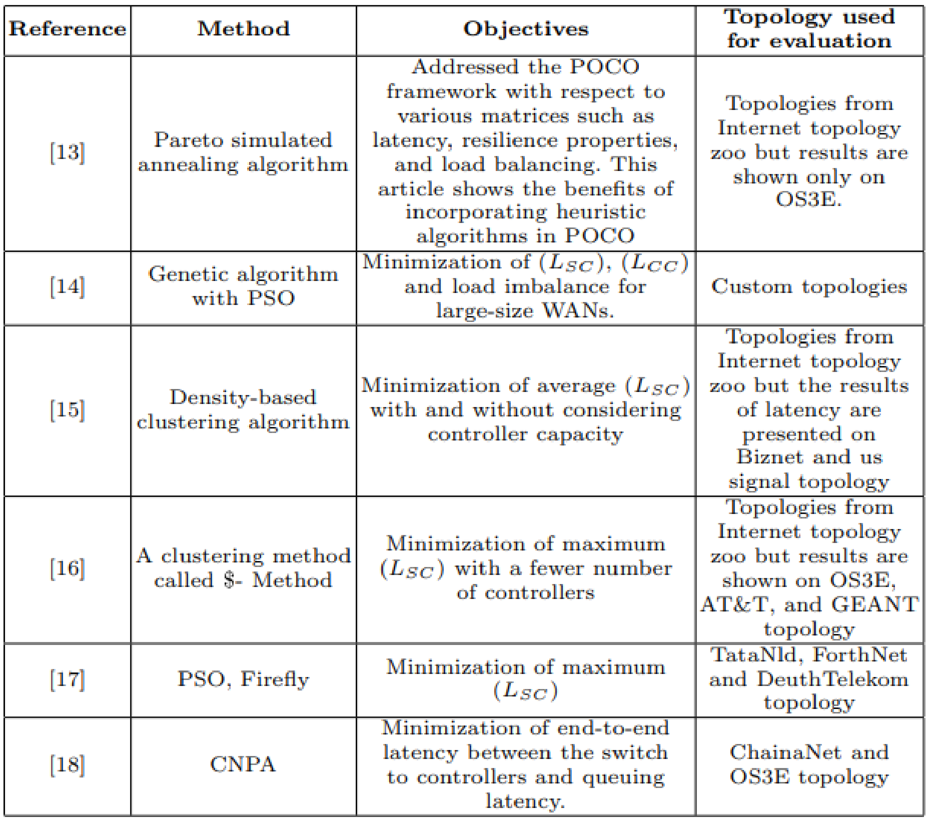
Table 1 shows a comparative analysis between existing approaches developed for data-centers networks. For modest to intermediate network sizes, it is easy to find all the deployment locations in an exhaustive way, but it is very difficult to apply those exhaustive algorithms in large-size networks, especially for dynamic networks for their limited computation time and resource constraints. Thus, in the next section, we have described some methods that can be applied in large-size networks.

2.2. CPP solution for WANs

In [13], a Pareto Optimal COntroller placement (POCO) algorithm for large-sized networks has been introduced with respect to different optimization objectives ((L_{SC}), (L_{CC}), resilience, load balance). By looking into the time and resource constraints of dynamic networks a heuristic algorithm has been extended into it. The solution might not give an accurate result but it executes faster than others.

In [14], another Pareto optimal solution has been proposed to minimize the L_{SC}, L_{CC}, and load imbalance for large-size WANs. They introduced a generic model that can further be able to develop many other optimization objectives (energy, controller migration). By maintaining generality with a larger search space, the authors proposed a Multi-Objective Genetic algorithm (MOGA) where the mutation function is based on Particle Swarm Optimization (PSO) to solve the CPP. A pre-computed global optimum position is kept for each proposed objective and then the global optimum position is selected depending on the best accordance to a parent to guide the mutation of the parent.

Table 2. A comparative analysis between existing approaches developed for WANs.



Reference	Method	Objectives	Topology used for evaluation
[13]	Pareto simulated annealing algorithm	Addressed the POCO framework with respect to various matrices such as latency, resilience properties, and load balancing. This article shows the benefits of incorporating heuristic algorithms in POCO	Topologies from Internet topology zoo but results are shown only on OS3E.
[14]	Genetic algorithm with PSO	Minimization of (L_{SC}), (L_{CC}) and load imbalance for large-size WANs.	Custom topologies
[15]	Density-based clustering algorithm	Minimization of average (L_{SC}) with and without considering controller capacity	Topologies from Internet topology zoo but the results of latency are presented on Biznet and us signal topology
[16]	A clustering method called $- Method	Minimization of maximum (L_{SC}) with a fewer number of controllers	Topologies from Internet topology zoo but results are shown on OS3E, AT&T, and GEANT topology
[17]	PSO, Firefly	Minimization of maximum (L_{SC})	TataNld, ForthNet and DeuthTelekom topology
[18]	CNPA	Minimization of end-to-end latency between the switch to controllers and queuing latency.	ChainaNet and OS3E topology

In [15], the authors tried to overcome the shortcomings of heuristic approaches and proposed a clustering algorithm based on the density of network nodes. the required number of controllers is decided based on the node densities. The authors focused on minimizing the average L_{SC}. It has also been shown what will be the cluster size for capacitated controller placements.

In [16], another clustering method has been proposed that efficiently finds the controller locations for minimizing the maximum L_{SC}. The authors solved a matrix by eliminating the matrix element whose values are larger than the given maximum value. This bounded maximum value is changed in every iteration until it finds the required number of controller locations and the set of switches assigned in these controller locations. They have also shown that the solution is cost-effective as it needs a smaller number of controllers than others.

In [17], CPP has been solved using two well-defined meta-heuristic techniques considering L_{SC} as a performance metric. Authors developed PSO and firefly algorithms where L_{SC} is the fitness function that needs to be minimized.

In [18], the authors proposed a clustering-based method aiming to minimize the L_T, L_{SC}, and L_{proc} latency between the switch to controllers and the controller's queuing latency. This is the extended version of their previous work [9]. The clustering algorithm proposed in the articles [9,18] is the same. In [18], they called their clustering method CNPA (clustering-based network partitioning algorithm) and proposed a multi-controller placement method to further reduce the L_Q. The requirement of controllers is determined for each cluster and multiple controllers are placed where the density of switches is higher to reduce the controllers L_Q.

Table 2 shows a comparative analysis between existing approaches developed for WANs.

3. Future Research works

Apart from the solutions proposed for solving the CPP, there are still several open research issues that need to be considered while solving the latency-aware CPP. To motivate the readers on this topic, we described some of them and gave a direction for future study.

i) Consideration of dynamic CPP over static CPP. In the dynamic network environment, network instances are changed so frequently depending on the traffic loads. Thus, a CPP solution for a dynamic environment needs to be given more attention as a future job.

ii) Latency is a more crucial aspect of large-size networks. Many heuristic solutions have been proposed for large-size networks but those solutions are stuck on some local optimum solutions. Therefore, developing a heuristic solution is needed which will give a global solution.

iii) Most of the research only focuses on wired medium networks but, it is also equally important to give attention to wireless network topologies. Thus, a huge opportunity is there to solve the CPP for wireless network topologies.

4. CONCLUSIONS

Controller Placement is the primary decision for constructing a SDN network. For architectural behavior, CPP has become one of the hot topics in SDN. In this paper, we investigated several CPP solutions that are mainly focused on optimizing latency. At first, the concept of SDN and its architecture is elaborated briefly. Afterward, we described why latency is more important to take into consideration while solving the CPP. We have further classified the latencies of the control plane into L_{SC}, L_{SC}, and processing latency. The mathematical formulation of those latencies is also presented in this article. The proposed solutions have been divided into two categories according to the application scenarios which are data-center networks and WANs. We have also presented a comprehensive study of these latency-aware solutions and show there are still some open research issues that are essential to be overcome in the near future.

REFERENCES

[1] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, “Software-defined net working (sdn): a survey,” Security and communication networks, vol. 9, no. 18, pp.
5803–5833, 2016.

[2] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, “Flowvisor: A network virtualization layer,” OpenFlow Switch Consortium, Tech. Rep, vol. 1, p. 132, 2009.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” ACM SIGCOMM computer communication review, vol. 38, no. 2, pp. 69–74, 2008.

[4] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, and M. Keshtgary, “A survey on sdn, the future of networking,” Journal of Advanced Computer Science & Technology, vol. 3, no. 2, pp. 232–248, 2014.

[5] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, “A survey of controller placement problem in software-defined networking,” IEEE Access, vol. 7, pp. 24 290–24 307, 2019.

[6] B. Heller, R. Sherwood, and N. McKeown, “The controller placement problem,” ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, pp. 473–478, 2012.

[7] W. Liu, Y. Wang, J. Zhang, H. Liao, Z. Liang, and X. Liu, “Aamcon: an adaptively distributed sdn controller in data center networks,” Frontiers of Computer Science, vol. 14, pp. 146–161, 2020.

[8] M. He, A. Basta, A. Blenk, and W. Kellerer, “Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows,” in 2017 IEEE International Conference on Communications(ICC). IEEE, 2017, pp. 1–7.

[9] 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,” in 2016 IEEE International Conference on Communications (ICC). IEEE, 2016, pp. 1–6.

[10] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, “The role of the inter-controller consensus in the placement of distributed sdn controllers,” Computer Communications, vol. 113, pp. 1–13, 2017.

[11] G. Yao, J. Bi, Y. Li, and L. Guo, “On the capacitated controller placement problem in software defined networks,” IEEE communications letters, vol. 18, no. 8, pp. 1339–1342, 2014.

[12] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The internet topology zoo,” IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765–1775, 2011.

[13] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, “Heuristic approaches to the controller placement problem in large scale sdn networks,” IEEE Transactions on Network and Service Management, vol. 12, no. 1, pp. 4–17, 2015.

[14] L. Liao and V. C. Leung, “Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in sdns,” in 2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN).
IEEE, 2017, pp. 1–6.

[15] 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 network ings,” Computer Networks, vol. 112, pp. 24–35,

[16] M. Dhar, B. K. Bhattacharyya, M. Kanti Debbarma, and S. Debbarma, “A new optimization technique to solve the latency aware controller placement problem in software defined networks,” Transactions on Emerging Telecommunications Technologies, vol. 32, no. 10, p. e4316, 2021

[17] K. S. Sahoo, S. Sahoo, A. Sarkar, B. Sahoo, and R. Dash, “On the placement of controllers for designing a wide area software defined networks,” in TENCON 2017-2017 IEEE Region 10 Conference. IEEE, 2017, pp. 3123–3128.

[18] G. Wang, Y. Zhao, J. Huang, and Y. Wu, “An effective approach to controller placement in software defined wide area networks,” IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 344–355, 2017

Authors

Mili Dhar is currently working as an Assistant Professor of the Computer Science Department in Galgotias University, Greater Noida, India. She has completed B.Tech, M.tech and Ph.D. from the Tripura Institute of technology (TIT) Narsingarh, National Institute of Technology Arunachal Pradesh (NIT AP), and National Institute of Technology Agartala (NIT Agartala) in 2016, 2018 and 2023 repectively. Her research interest includes Controller Placement Problem in Software Defined Network, Wireless sensor network, network optimization and Cryptography

Saikat Majumder is currently working as a PhD scholar of the Electrical Engineering Department in Indian Institute of Technology (IIT) Kanpur, India. He has completed B.Tech from the North Eastern Regional Institute of Science and Technology (NERIST) Arunachal Pradesh, India, in 2018 and M.Tech from Indian Institute of Engineering Science and Technology (IIEST) Shibpur, India, in 2021. His research interest includes Communications, Robotics, Reinforcement learning, Control system.