An Efficient Controller Placement Algorithm using Clustering in Software Defined Networks

Joshua Jacob, Sumedha Shinde, and Narayan D.G.

KLE Technological University, Hubballi, Karnataka, India

https://doi.org/10.26636/jtit.2023.4.1371

Abstract — Software defined networking (SDN) is an emerging network paradigm that separates the control plane from data plane and ensures programmable network management. In SDN, the control plane is responsible for decision-making, while packet forwarding is handled by the data plane based on flow entries defined by the control plane. The placement of controllers is an important research issue that significantly impacts the performance of SDN. In this work, we utilize clustering techniques to group networks into multiple clusters and propose an algorithm for optimal controller placement within each cluster. The evaluation involves the use of the Mininet emulator with POX as the SDN controller. By employing the silhouette score, we determine the optimal number of controllers for various topologies. Additionally, to enhance network performance, we employ the meeting point algorithm to calculate the best location for placing the controller within each cluster. The proposed approach is compared with existing works in terms of throughput, delay, and jitter using six topologies from the Internet Zoo dataset.

Keywords — clustering, controller placement, PAM, K-means++, silhouette score, SDN

1. Introduction

Software defined networking (SDN) is an emerging concept that differs from traditional networking by separating the control plane from the data plane [1]. In SDN, the control plane comprises specialized controllers that act as an intelligent core, while the data plane consists of simplified switches responsible for packet forwarding. This decoupling enables the network to be programmable directly, offering numerous advantages, such as streamlined network management, improved network efficiency, and support for future network advancements. Additionally, SDN introduces an application layer which interacts with the data plane through the control plane using northbound APIs. This integration allows for network control and the provision of network services. The versatility and benefits of SDN have led to its widespread adoption in various domains, including large data centers, cloud computing environments, Internet of Things (IoT) applications, and wireless mesh networks.

OpenFlow serves as a communication interface that represents SDN. Initially, Open-Flow operated on the assumption of a single controller for simplicity. However, as networks expand, this approach can encounter scalability and performance challenges. To overcome these issues, several multi-controller approaches have been proposed. One significant challenge that arises when using multiple controllers is known as the controller placement problem [2]. This problem revolves around determining the optimal placement of controllers within an SDN-enabled network and allocating the associated switches to achieve specific objectives. These objectives may include reducing latency, enhancing reliability, or improving energy efficiency. The aim of controller placement solutions is to identify optimal methods for connecting controllers and their corresponding switches in an SDN-enabled network. The literature offers various clustering algorithms that address the controller placement problem (CPP).

Recently, many clustering-based algorithms have been introduced to address the issue of CPP. The objective of these approaches is to determine the ideal number of clusters and identify suitable locations within each cluster for placing controllers. In the present study, we leverage clustering algorithms to group networks into multiple clusters and present an algorithm aimed at achieving optimal controller placement within each cluster. The contributions of this work are listed below:

- replication of a standard Internet Zoo topology on Mininet,
- segregation of the topology into clusters, based on clustering algorithms,

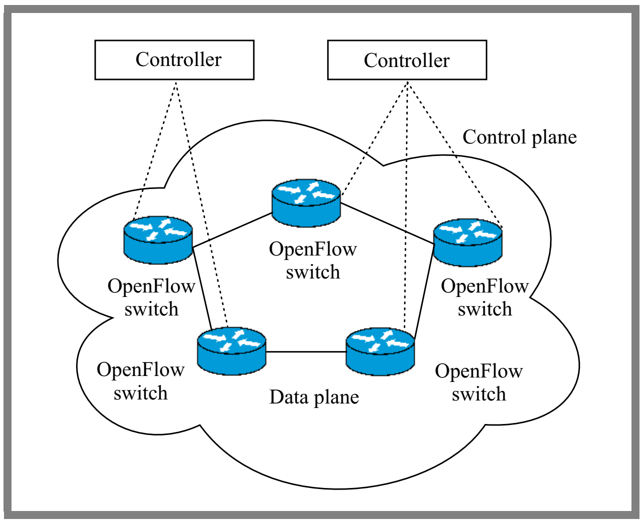

Fig. 1. Controller placement problem.

- use of the silhouette score to determine the optimal number of controllers needed for the topology,
- computation of the controller's optimal placement by relying on a meeting point algorithm to improve throughput, delay, and jitter.

The structure of the paper is as follows. A literature survey and a summary of related works are presented in Section 2. The proposed technique and the algorithms applied to come up with the answer are described in Section 3. The experimental design, as well as the findings and analysis, are covered in Section 4. Section 5 discusses the findings and the work that will be done in the future.

2. Background Study

In this section, we discuss the controller placement problem encountered in SDNs and then the related research performed in that specific field.

2.1. Controller Placement Problem in SDN

SDN offers a centralized perspective of a network by segregating the control plane from the data plane. This approach of preserving resource management is optimized by having a general network view. However, implementation of an SDN with a single physical controller causes problems with reliability and scalability. Both can be solved by a physically decentralized but semantically centralized SDN controller, with the design thereof shown in Fig. 1. Along with its advantages, distributed SDN also suffers from a difficulty in determining the quantity and in positioning the controllers needed in the network. This issue is known as the controller placement problem (CPP).

The performance of the network can be greatly influenced by the configuration of the controllers, as they are responsible for generating forwarding rules and implementing them on the switches. In order to increase network performance and ensure high availability of the network, several controllers are necessary. The quantity of controllers that must be deployed and their locations are two different components of the CPP challenge. When determining the required number of units, there is a trade-off between performance and deployment costs. By deploying numerous controllers, the network becomes robust to controller failures while simultaneously reducing latency. The device pool and the mapping between a switch and a controller are fixed in the CPP solutions. Furthermore, an effective switch migration strategy should take switch migration cost into account as well.

2.2. Related Work

In [1], an in-depth analysis of the SDN controller placement issue is presented. Generally, delay, dependability, and energy are the three primary categories that are dealt with in the literature. Since the control path serves as the conduit for the management and control SDN messages, dependability of SDNs is directly impacted by dependability of the control path. The initial expenses of building the network and subsequent costs of its operation and maintenance make up the majority of the SDN's price tag.

Paper [2] reviews several SDN-optimized controller placement algorithms. The static controller placement strategy is a topology-based method that is only applicable in a single topology and does not consider limits on the capacity of the controllers, rendering it inapplicable in actual networks. The dynamic controller placement method is a topology-based method that is not constrained to any topology and disregards limits on the controllers' capacity.

Density-based spatial clustering (DBSCAN) and density peak (DP), which are clustering techniques based on density in unsupervised learning, are introduced by the authors of [3]. They can locate noise samples in a potential data set and may cluster data of any shape. An improved DBSCAN and DP algorithm-based two-stage clustering method was adopted. According to the experiments, TSCM has the potential to effectively override both the manual selection of cluster centers in DP and the parameter settings in DBSCAN, in the context of DP and DBSCAN. The clustering performance is thus greatly improved.

Authors of [4] introduce a machine learning (ML) methodology to manage network traffic, focusing on forecasting the probable count of controllers to be situated within the network. This predictive mechanism is centralized in nature and is implemented as a virtual network function (VNF) within the network function virtualization (NFV) framework. Controller count prediction is based on the utilization of anticipated traffic patterns. Subsequently, the calculated number of controllers is positioned optimally within the network using the K-medoid algorithm.

In [5], a two-phase technique that links utility-based game theoretic solutions with stable matching, with the goal of reducing controller response time and control traffic overhead, is proposed. When compared to static assignment, the suggested technique can, on average, cut response times by 86% and traffic overhead by 2%. The algorithms used here are coalitional game phase procedure and stable matching phase procedure. The criteria considered in this case are the controller's worst response time and the control traffic overhead generated by switch and controller communications.

CPP is assessed in article [6] using three meta-heuristics and one clustering technique, with four different network scales. The network's entire traffic is predictable and static. Each controller uses the flow setup time as the load. Network topology is divided using the GSO algorithm. GSOCCPP demonstrates its ability to arrange the controllers so that minimal end-to-end delays and shorter run times are obtained. Additionally, GSOCCPP performs well in large network topologies in terms of memory consumption and calculation time.

The authors of [7] use the K-means++ algorithm as a traffic engineering (TE) mechanism. TE is an important network application. It is an iterative process of analyzing and classifying the traffic and predicting future traffic to avoid network congestion. Initially, controller position is identified using the K-means++ initialization policy. Then, the controller is placed at optimal locations and the traffic is captured. The captured traffic is classified as low, high and normal. Then, a greedy approach is used to distribute the traffic to different controllers. The number of controllers is given as input for the placement algorithm in order to achieve optimal placements.

Controller placement-related issues are discussed in paper [8], where they are divided into three categories. These are: design and implementation, development and application, as well as investigation and evaluation. Pareto borders are used to guide automated decision making. The benefits of integrating the SDN control plane with network management systems within the controller will be shown. The load balance between controller instances and communication delays has been optimized, allowing for dynamic traffic patterns to be taken into consideration. Finally, four strategies are demonstrated for weighting and rating individual solutions.

Article [9] discusses the method used to solve the issue of placing SDN controllers by figuring out how many SDN controllers should be placed, and when, in a wide area network, along with describing any performance and financial consequences. The authors apply the PAM algorithm to cluster the network and utilize the Johnson algorithm to identify the most optimal positions for the placement of controllers.

The authors of paper [10] propose a method for accommodating the increasing number of devices, necessitating network management strategies to be deployed. The open-source nature and programmability of network devices are facilitated by SDN and network function virtualization providing isolating the control plane and the data plane from the network. The authors argue that SDN attempts to create a programmable network. SDN employs a traffic engineering mechanism to determine the optimal number of controllers to be deployed in the network. To prevent network congestion, traffic engineering is an iterative process that reads the state of the network, analyses and categorizes the traffic, and forecasts network traffic.

In [11], the researchers present a method for multiple controller placement (MCP) that addresses both the determination of the minimum number of required controllers and their optimal placement within the network. Traditional solutions of CPP typically involve objective functions which consider various factors and constraints, such as propagation latency between switches, controllers, and inter-controllers, as well as the capacity of controllers and switches. As far as we are aware, this is the first time a modern review of CPP has been attempted. Paper [11] categorizes the notion of CPP, evaluates the solutions that are available, identifies their shortcomings and potential future applications. With this knowledge, future researchers will be better able to come up with new, creative CPP solutions.

In order to reduce the average propagation latency between switches and controllers, article [12] proposes a solution based on the modified density peaks clustering algorithm. A controller oversees each of the numerous subnetworks that make up the total network. To assess the effectiveness of their strategy, the authors perform their tests using the Internet 2 OS3E topology. Their controller placement approach can drastically minimize latency, according to the test results. In particular, average latency can be decreased by 10% when compared to the results obtained with the use of the optimized K-means approach and by 35% when compared to K-means.

In [13], to overcome the controller placement problem, a hierarchical K-means algorithm is introduced along with a technique to divide the expansive network into several SDN domains using a single controller. Trials relying on the Internet 2 OS3E topology are conducted to gauge the effectiveness of such an approach. According to experimental findings, the partition technique is more balanced than the optimized K-means algorithm.

By employing a complex network analysis theory, the topology of a network with the intended controller placement is conceptualized in [14] as a network consisting of multiple communities. Subsequently, a suitable location for each controller is identified within each community, thereby circumventing the need for a complex global deployment approach. To maintain a balanced distribution of controlled switches across communities, the Louvain heuristic algorithm incorporates a scale constraint factor. This factor limits the number of nodes within each community, ensuring a more equitable distribution of nodes among different communities. This method autonomously identifies partitions with community characteristics based solely on network topology.

In [15], a framework to enhance the reliability and scalability of SDN is presented. A heuristic multi-level capacitated CPP approach is used to efficiently assign switches to compatible controllers. This method aims to optimize the number of controllers, minimize delays, and evenly distribute the workload among the controllers. The allocation components are then integrated with a genetic algorithm (GA) to further improve the placement of controllers. The effectiveness of the GA-based MCCPP is evaluated and compared to that of the heuristic-based CCPP using various network topologies. The adaptive GA-based approach demonstrates convergence and achieves significantly reduced latency results.

Paper [16] introduced a method for efficient approximation near-optimal placements using cross entropy. The proposed approach was tested on five real topologies of the Internet topology Zoo dataset. The results indicated that the approach consistently achieved minimal propagation latency across different network sizes and varying numbers of controllers. Simulation results showed that the cross-entropy method consistently produced optimal or near-optimal solutions in terms of placing controllers in networks of different scales and when dealing with different controller numbers, with a margin of less than 1% from the optimal solution and up to 157 times faster than the brute force method.

In [17], the authors proposed two approaches, namely a priority-based switch placement method and an incremental controller placement approach for hybrid SDNs. Simulation results indicated that the introduced scheme significantly enhanced programmable traffic. A comparison with existing solutions revealed that, when the number of time slots was 2, the proposed approach exhibited a 12.30% and 13.14% increase in programmable traffic as compared to map-first and greedy approaches, respectively.

The multi-objective control plane dimensioning problem for hybrid SDN networks is researched in article [18]. The parameters considered are network resilience, flow latency and load balancing. The researchers have proposed optimization models for both single and multi-objective controller placements. These models incorporate the processing latency of controllers, utilizing the queuing theory, and the synchronization latency of inter-controller network state, utilizing Steiner trees as a basis.

In paper [19], a novel swarm intelligence algorithm called the naked mol-rat algorithm is presented to tackle the optimization problem of controller placement. The algorithm's performance is compared with that of existing approaches. This study specifically focuses on presenting the NMR algorithm as a solution for multi-controller placement in an SDN environment and evaluating its performance in comparison to the bat algorithm.

3. Proposed Methodology

In this section, we discuss the specific approach behind the introduced model, which includes a description of the proposed system model and the system modules. Additionally, we discuss such methods as meeting point and K-means++ algorithms.

3.1. Proposed System Model

The underlying network is described as an undirected graph made up of elements $G(S, L)$, where S denotes the set of SDN switches and L is the set of physical connections. The network will have k controllers, and a set of controllers will be deployed therein. A specific number of switches is controlled by each controller. The system model is divided into two modules as shown in Fig. 2.

The role of the network partition module is to determine the number of network partitions by analyzing node properties across different network environments. We extract the node features of the topology, such as their coordinates, connections, and numerous other aspects, after the standard topology has been chosen. Once the node attributes have been obtained, we submit them to a clustering algorithm which divides the networks into k segments and determines the optimum number of controllers k, needed for the specific topology. Additionally, we replicate the same network design using Miniedit, a Python application that offers a GUI for maintaining and building topologies.

The controller placement module's duties include determining the best places for the controllers to be placed and testing the network's functionality using predetermined criteria. Once the network is configured in Miniedit, we can connect to two hosts using Xterm and transmit traffic across an UDP channel to evaluate the network's performance. This is done by using the distributed internet traffic generator (D-ITG) command. In order to determine the best positions for the controllers, we also use the suggested controller placement method. To show the topology of our network visually, this is duplicated in Miniedit. We used the D-ITG tool to analyze the network's performance after the topology was visualized in Miniedit. Lastly, we selected the best methodology for the specified topology after comparing the performance of several methodologies.

3.2. Module Description

Two modules are used: for determining the ideal number of controllers, selecting the best clustering algorithm, and determining the ideal placement of the controllers.

To find the best clustering algorithm, we have taken 6 topologies from the Internet Zoo dataset, i.e. Abilene, Sanren, Grena, Spiralight, Oxford and Atmnet. We have taken 3 clustering algorithms to test them on specific topologies and to evaluate the performance of these algorithms based on the silhouette score. Algorithm 1 illustrates the K-means++ steps in the context of SDN network clustering.

Algorithm 1. K-means++ algorithm.

```
Input:
    $A = a_1, a_2, \ldots, a_n$ (set of entities required
        to be clustered in a SDN subdomain)
    $k$ (number of SDN clusters)
    $MaxIterations$ (total number of iterations)
Output:
    $CL = cl_1, cl_2, \ldots, cl_n$ (set of centroids of
        the clusters in a SDN subdomain)
    $n = n(a), a = 1, 2, \ldots, n$
        (SDN cluster labels numbers for $A$)
Begin:
1: for $a_i \in CL$ do
2:   $a_i \leftarrow a_j \in CL$ (e.g. random selection)
3: end for
4: for $a_i \in CL$ do
5:   $l(a_i) \leftarrow minDist(a_i, a_j),\ j \in \{1, \ldots, k\}$
6: end for
7:   $change \leftarrow false,\ iters \leftarrow 0$
8: Repeat
9:     for $a_i \in CL$ do $UpdateCluster(a_i)$
10:    end for
11:    for $a_i \in A$ do
12:      $l(a_i) \leftarrow minDist(a_i, a_j), j \in \{1, \ldots, k\}$
13:      if $minDistance \neq l(a_i)$ then
14:        $l(a_i) \leftarrow minDistance, change \leftarrow true$
15:      end if
16:    $iters++$
17:    end for
Until $change = true$ and $iters \leqslant MaxIterations$
```

Silhouette score. The silhouette coefficient, also referred to as the silhouette score, is a statistic used to gauge the effectiveness of a clustering method. Its value ranges from –1 to 1 and is given by:

$$\text{Silhouette score} = \frac{C - D}{\max(C, D)}\ , \tag{1}$$

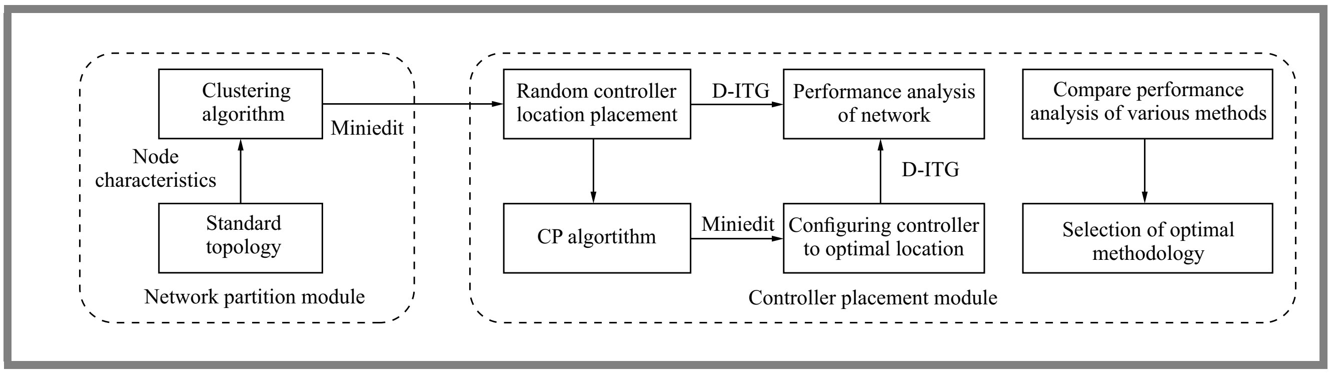
Clustering algorithm
Miniedit
Node characteristics
Standard topology
Network partition module
Random controller location placement
D-ITG
Performance analysis of network
Compare performance analysis of various methods
D-ITG
CP algortithm
Miniedit
Configuring controller to optimal location
Selection of optimal methodology
Controller placement module

Fig. 2. Proposed system model.

where C is the typical separation between each point in a cluster and D is the mean separation between each cluster. Measure "1" indicates groups that are distinct and separated by a large distance, "0" shows that there is no statistically significant difference between clusters or that they are not differentiable. Value of "—1" demonstrates that the clusters were wrongly assigned.

Algorithm 2. Controller placement using the meeting point algorithm.

```
Input:
    Location points $K = k_1, k_2, ..., k_n$ of $n$ switches at
    given location $k_i$ on the grid containing $|G|$ cells
    in a SDN subdomain.
Output:
    Optimal meeting point – a cell
    on the grid $MPA(Lat, Lon)$
Begin
1: $MPA \leftarrow NULL$
2: $MinCost \leftarrow +\infty$
3: for $g_i \in G$ do
4:     $TotalSum \leftarrow 0$
5:     for $k_j \in K$ do
6:     $TotalSum \leftarrow TotalSum + d_N(g_i, k_j)$
7: end for
8: $Cost \leftarrow Totalsum$
9:   if $Totalcost < MinCost$ then
10:    $MinCost \leftarrow Cost$
11: end if
12: $MPA \leftarrow g_i$
End
```

Once the topology is segregated into clusters, we need to optimize the controller-to-switch-to-controller distance, such that the performance of the network is enhanced. To identify the most optimal coordinates for the controllers, we use an algorithm called meeting point [20] (Algorithm 2). The meeting point algorithm is a computational method used to determine the optimal location for a group, based on individual locations. It takes into account such factors as distance and preferences to find a convenient and fair meeting point for all the parties involved. The meeting point algorithm in SDN is a routing strategy that determines the optimal point where traffic from multiple sources can converge and be efficiently processed.

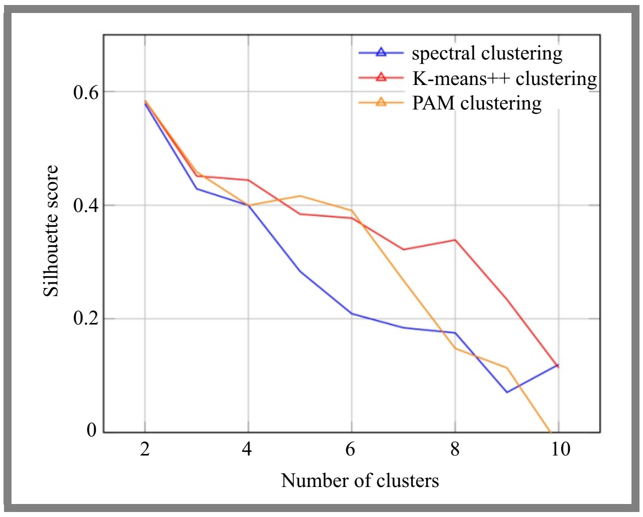

Fig. 3. Silhouette score for Abilene topology.

4. Results and Discussions

This section compares various clustering algorithms based on their silhouette score for a given topology. Furthermore, we compare the averages of throughput, latency, and jitter of the proposed solution with the outcomes generated with the use of other approaches.

4.1. Clustering Results

For evaluating the clustering algorithms and finding the optimal number of clusters, we use the silhouette coefficient as the key performance metric.

Tab. 1. Silhouette score for various topologies for $K = 2$.

Topology	No. of nodes	K-means++	Spectral	PAM
Abilene	11	0.584	0.574	0.581
Sanren	7	0.376	0.297	0.293
Grena	10	0.523	0.697	0.684
Atmnet	21	0.656	0.431	0.530
Spiralight	15	0.534	0.624	0.293
Oxford	19	0.556	0.567	0.502

Tab. 2. Controller placement results using K-means++.

Topology	K-means++ and Johnson [9]			K-means++ and meeting point		
	Avg. throughput	Avg. delay	Avg. jitter	Avg. throughput	Avg. delay	Avg. jitter
Sanreen	6.52	0.51	0.43	6.74	0.33	0.27
Abilene	7.21	0.47	0.39	7.77	0.29	0.24
Atmnet	7.68	0.44	0.42	8.32	0.34	0.39

Tab. 3. Controller placement results using spectral clustering.

Topology	Spectral and Johnson			Spectral and meeting point		
	Avg. throughput	Avg. delay	Avg. jitter	Avg. throughput	Avg. delay	Avg. jitter
Grena	7.33	0.41	0.38	7.57	0.39	0.34
Spiralight	7.67	0.46	0.33	8.05	0.41	0.29
Oxford	7.26	0.48	0.40	7.61	0.43	0.35

To find the best clustering algorithm for a small scale-based topology, we have taken 6 topologies from the Internet Zoo dataset, i.e. Abilene, Sanren, Grena, Spiralight, Oxford, and Atmnet. We have taken 3 clustering algorithms to test them on the topologies and to evaluate the performance of these algorithms based on the silhouette score. Figure 3 shows the silhouette score for the Abilene topology. The results reveal that the number of optimal clusters is 2 and that K-means++ performs better than spectral and PAM clustering approaches, for this topology. Table 1 gives the silhouette score for six topologies for $K = 2$. We can observe, from the findings concerning the six aforementioned topologies, that the K-means++ method has demonstrated to have a higher silhouette score for small scale topologies. However, we find that spectral clustering yields superior outcomes for dense and medium-sized topologies, such as Grena, Spiralight, and Oxford. This is because spectral clustering outperforms K-means++ and PAM in dense topologies. Additionally, we determine that $K = 2$ is the right number of clusters based on the silhouette score. So, using the K-means++ clustering method, we divided the topology into two groups to work on Sanren, Abilene, and Atmnet topologies. Further, by using the spectral clustering method, we divided the topology into two groups to work on Grena, Spiralight, and Oxford topologies.

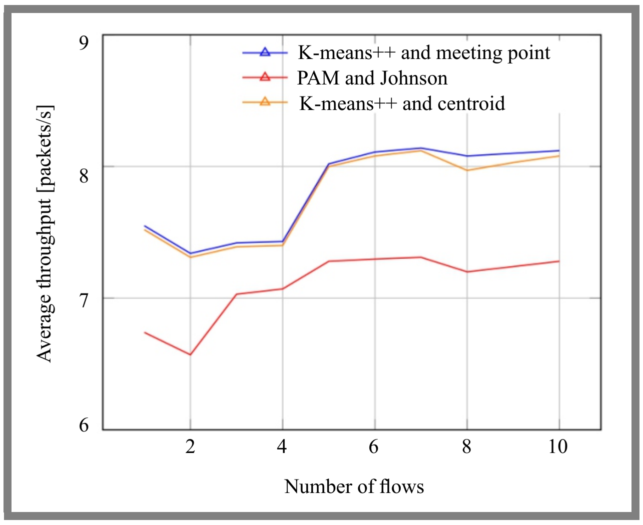

Fig. 4. Average throughput.

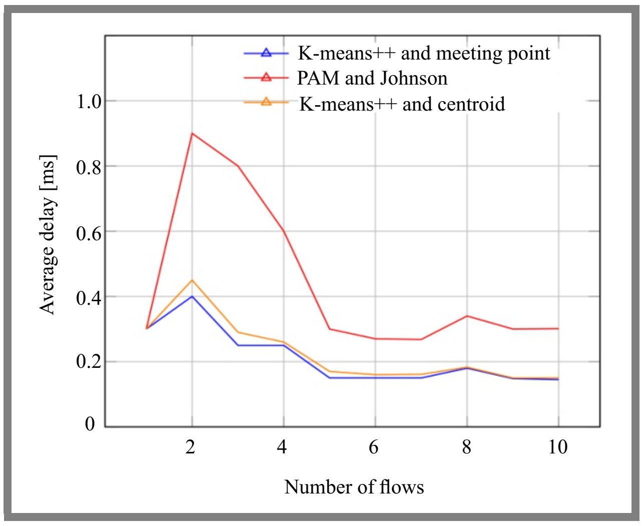

Fig. 5. Average delay.

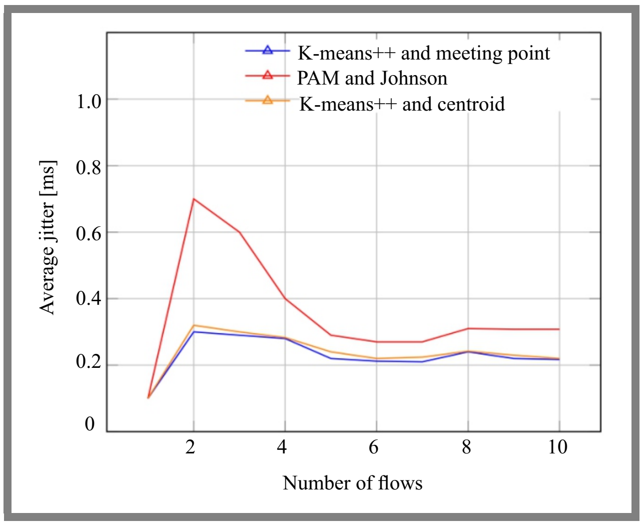

Fig. 6. Average jiiter.

4.2. Controller Placement Results

After the topology has been divided into clusters, it is necessary to optimize the controller-to-switch-to-controller distance in order to improve network performance. We employ a method called the ideal meeting point algorithm to identify the most advantageous coordinate for the controllers. We test this against the algorithm used in [9]. Any of the points whose distance is lower than the distance at the centroid will be the point with the shortest distance travelled.
The centroid can occasionally provide an incorrect position, since we simply take the average of all points and round off the value, regardless of the minimum distance covered by all the points. Undoubtedly, this will result in a point that is in the center of all the other points, which is typically the correct response. The centroid formula, however, would never allow us to know if the nearest point is always one of the input points. We double-check the neighbors of the ”losest point” to resolve this issue.
We discuss the performance of the proposed controller placement algorithms using the following QoS parameters.

Throughput is a measure of how many units of information a system can process in a given amount of time. Using the distributed internet traffic generator (D-ITG) command, we connect to two hosts using Xterm and transfer the traffic across a UDP channel in order to measure the average throughput. We test the throughput for each flow as we incrementally increase the number of hosts communicating, or the number of flows.
Figure 4 shows the results concerning the Abilene topology. From the figure it can be inferred that the average throughput of (K-means++ and meeting point) is relatively higher than the average throughput of (PAM and Johnson). This results from placing the controller at optimal locations. It is observed that for flow number 2, the average throughput decreases in all three cases. The increased number of packet-in messages to the controller can be attributed to the lack of a matching probability between the preserved flow rules and the newly arriving packets. It is further observed that for flow number 8, the average throughput decreases in all three cases. This is because after a certain time the switch reaches a hard timeout and clears its flow table leading to an additional processing delay. It can be inferred that for every 5 flows the switch reaches a hard timeout and clears its flow table leading to an additional processing delay for the Abilene topology under consideration.

Average delay is the average time taken to process n packets. We employed the D-ITG approach to inject traffic into the network and determine the average delay. By gradually incrementing the number of flows, we could evaluate the network's ability to maintain consistent performance under increasing traffic. Figure 5 shows the results concerning the Abilene topology. From the figure it can be inferred that average delay of (K-means++ and meeting point) is relatively lower than the average delay of (PAM and Johnson). This results from placing the controller at reliable locations. It is observed that at flow number 2, the average delay increases in all three cases. The reason behind the increase in the number of the controller's packet-in messages is the absence of a matching probability between the preserved flow rules and the incoming packets, leading to a lack of correspondence.

Average jitter refers to the mean irregular fluctuations in latency experienced by a packet flow between two systems, arising from varying delays encountered by individual packets during the transmission. To calculate the average delay, we used the D-ITG to introduce traffic into the network and calculated the average jitter. By gradually increasing the number of flows, we could assess the network's capability to sustain consistent performance under an escalating traffic load.
From Fig. 6, it can be inferred that he average jitter of (K-means++ and meeting point) is relatively lower than the average jitter of (PAM and Johnson). This is due to the selection of optimal locations for the controller.
Across six unique network topologies, we evaluate the effectiveness of the proposed approach by analyzing average throughput, average latency, and average jitter. This assessment encompasses the performance of two distinct clustering algorithms on specific topologies, as summarized in Tabs. 2 and 3. Upon analyzing the findings presented in Tab. 2, it is apparent that the synergy between the K-means++ clustering method and the meeting point algorithm has displayed better performance when compared with the outcomes achieved when using K-means++ in conjunction with the Johnson algorithm. This notable enhancement in performance is consistently observed across three diverse topologies.
It is important to highlight that the authors in [9] did not utilize spectral clustering in combination with the Johnson algorithm. In our study, we specifically assessed the performance of spectral clustering when used in conjunction with the Johnson algorithm. When analyzing Tab. 3, it becomes evident that the utilization of spectral clustering alongside the meeting point method has yielded significantly superior results compared to both spectral clustering and the Johnson algorithm, across three distinct topologies.
In summary, our research showcased that the combination of the K-means clustering approach with the Johnson or Meeting Point algorithm exhibits superior performance when dealing with topologies featuring a lower number of nodes, as evidenced in Tab. 2. Conversely, the fusion of spectral clustering with the Johnson or meeting point algorithms proves more effective for topologies comprising a larger number of nodes, as demonstrated in Tab. 3. Moreover, our results underscore the superiority of the meeting point algorithm over the Johnson in terms of performance. This discrepancy can be attributed to the following factors:

Efficiency: the meeting point algorithm has a faster run time than the Johnson approach, meaning it can solve more complex instances of the controller placement problem more quickly. The meeting point algorithm is based on a greedy approach, meaning it can solve the controller placement problem in polynomial time, typically $O(n^3)$, where n is the number of switches in the network. This makes it more efficient than the Johnson algorithm which has an exponential runtime of $O(2^n)$.

Scalability: The meeting point algorithm exhibits superior scalability in comparison with the Johnson approach. This implies its ability to manage more extensive networks containing a higher number of switches and controllers. The meeting point algorithm operates in a distributed manner, allowing convenient parallelization to accommodate larger networks. On the other hand, the Johnson algorithm operates centrally, demanding the calculation of all switch-to-switch distances before placement. This characteristic makes it challenging to effectively scale for larger networks.

Accuracy: the meeting point algorithm provides a more accurate solution than the Johnson approach, meaning it can achieve better performance in terms of network objectives. The meeting point algorithm takes into account the physical distance between switches and controllers when assigning switches to controllers. This consideration allows the meeting point algorithm to assign switches to the nearest controller, resulting in better network performance. On the other hand, the Johnson algorithm does not consider the physical distance between switches and controllers when assigning switches to controllers. Instead, it assigns switches to controllers based solely on their connectivity, which can lead to suboptimal controller placements and poorer network performance.

5. Conclusion

SDN is a new paradigm for networks that separates the control plane from the data plane and offers flexible network administration. Controller placement is an important issue in a large-scale SDN. The main aim of the controller placement process is to improve such performance metrics as average latency, average delay, and average jitter. Furthermore, the location of the controllers and effective switch-controller mapping is important.

In this work, we initially tested different clustering algorithms to identify the suitable clustering algorithm for the specific type of topology. Silhouette score was used to test the performance of the clustering algorithms. Next, we found the optimal locations of the controllers using the closest meeting point algorithm. The results obtained in the course of Mininet simulations reveal that the performance level achieved with the proposed controller placement algorithm is better than in existing works. As future work, we plan to work on a dynamic controller placement mechanism using such QoS parameters as latency and reliability. We also intend to evaluate the results in an OpenStack-based testbed environment.

References

[1] J. Lu *et al.*, "A Survey of Controller Placement Problem in Software-defined Networking", *IEEE Access*, vol. 7, pp. 24290–24307, 2019 (https://doi.org/10.1109/ACCESS.2019.2893283).

[2] S.-K. Yoon, Z. Khalib, N. Yaakob, and A. Amir, "Controller Placement Algorithms in Software Defined Network – A Review of Trends and Challenges", *MATEC Web of Conferences*, vol. 140, art. no. 01014, 2017 (https://doi.org/10.1051/matecconf/201714001014).

[3] M. Li, X. Bi, L. Wang, and X. Han, "A Method of Two-stage Clustering Learning Based on Improved DBSCAN and Density Peak Algorithm", *Computer Communications*, vol. 167, pp. 75–84, 2021 (https://doi.org/10.1016/j.comcom.2020.12.019).

[4] G. Ramya and R. Manoharan, "Traffic-aware Dynamic Controller Placement in SDN using NFV", *The Journal of Supercomputing*, vol. 79, no. 4, pp. 2082–2107, 2023 (https://doi.org/10.1007/s11227-022-04717-8).

[5] M. Ider and B. Barekatain, "An Enhanced AHPTOPSIS-based Load Balancing Algorithm for Switch Migration in Software-defined Networks", *The Journal of Supercomputing*, vol. 77, pp. 563–596, 2021 (https://doi.org/10.1007/s11227-020-03285-z).

[6] S. Torkamani-Azar and M. Jahanshahi, "A New GSO Based Method for SDN Controller Placement", *Computer Communications*, vol. 163, pp. 91–108, 2020 (https://doi.org/10.1016/j.comcom.2020.09.004).

[7] V. Huang, G. Chen, Q. Fu, and E. Wen, "Optimizing Controller Placement for Software-Defined Networks," in: *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Arlington, USA, pp. 224–232, 2019 (https://arxiv.org/pdf/1902.09451).

[8] O. Flauzac, E.G. Robledo, C. Gonzalez, F. Mauhourat, and F. Nolot, "SDN Architecture to Prevent Attacks with OpenFlow", in: *8th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, Reims, France, 2020 (https://doi.org/10.1109/WINCOM50532.2020.9272445).

[9] L. Mamushiane, J. Mwangama, and A.A. Lysko, "Controller Placement Optimization for Software Defined Wide Area Networks (SD-WAN)", *ITU Journal on Future and Evolving Technologies*, vol. 2, no. 1, pp. 45–66, 2021 (https://doi.org/10.52953/PUIU5171).

[10] G. Ramya and R. Manoharan, "Prediction Based Dynamic Controller Placement in SDN", *EAI Endorsed Transactions on Scalable Information Systems*, vol. 8, no. 32, 2021 (https://doi.org/10.4108/eai.27-4-2021.169420).

[11] A.K. Singh and S. Srivastava, "A Survey and Classification of Controller Placement Problem in SDN", *International Journal of Network Management*, vol. 28, no. 3, 2018 (https://doi.org/10.1002/nem.2018).

[12] Y. Qi *et al.*, "Towards Multi-controller Placement for SDN Based on Density Peaks Clustering", in: *ICC IEEE International Conference on Communications*, Shanghai, China, 2019 (https://doi.org/10.1109/ICC.2019.8761814).

[13] H. Kuang, Y. Qiu, R. Li, and X. Liu, "A Hierarchical K-means Algorithm for Controller Placement in SDN-based WAN Architecture", in: *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Changsha, China, pp. 263–267, 2018 (https://doi.org/10.1109/ICMTMA.2018.00070).

[14] W. Chen, C. Chen, X. Jiang, and L. Liu, "Multi-controller Placement towards SDN Based on Louvain Heuristic Algorithm", *IEEE Access*, vol. 6, pp. 49486–49497, 2018 (https://doi.org/10.1109/ACCESS.2018.2867931).

[15] A.A.Z. Ibrahim *et al.*, "A Modified Genetic Algorithm for Controller Placement Problem in SDN Distributed Network", in: *2021 26th IEEE Asia-Pacific Conference on Communications (APCC)*, Kuala Lumpur, Malaysia, 2021 (https://doi.org/10.1109/APCC49754.2021.9609838).

[16] J. Chen, H. Yin, C. Xiao, and D. He, "A Cross Entropy-Based Approach for Controller Placement Problem in Software Defined Network", in: *2021 International Conference on Information Technology and Biomedical Engineering (ICITBE)*, Nanchang, China, 2021 (https://doi.org/10.1109/ICITBE54178.2021.00010).

[17] T. Das and M. Gurusamy, "Multi-objective Control Plane Dimensioning in Hybrid SDN/Legacy Networks", *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2929–2942, 2021 (https://doi.org/10.1109/TNSM.2021.3066847).

[18] I. Maity, S. Misra, and C. Mandal, "SCOPE: Cost-Efficient QoS-Aware Switch and Controller Placement in Hybrid SDN", *IEEE Systems Journal*, vol. 16, no. 3, pp. 4873–4880, 2022 (https://doi.org/10.1109/JSYST.2021.3124280).

[19] A.B. Sapkota, B.B.R. Dawadi, and C.S.R. Joshi, "Multi-Controller Placement Optimization Using Naked Mole-Rat Algorithm over Software-Defined Networking Environment", *Journal of Computer Networks and Communications*, vol. 2022, art. no. 3145276, 2022 (https://doi.org/10.1155/2022/3145276).

[20] Closest Meeting Point [Online]. Available: https://www.educative.io/m/closest-meeting-point

Joshua Jacob, B.E.
School of Computer Science and Engineering
E-mail: joshuajacob1020@gmail.com
KLE Technological University, Hubballi, Karnataka, India
https://www.kletech.ac.in

Sumedha Shinde, Ph.D.
Department of Mathematics
https://orcid.org/0000-0002-4154-0025
E-mail: sumedha@kletech.ac.in
KLE Technological University, Hubballi, Karnataka, India
https://www.kletech.ac.in

Narayan D.G., Ph.D.
School of Computer Science and Engineering
https://orcid.org/0000-0002-2843-8931
E-mail: narayan_dg@kletech.ac.in
KLE Technological University, Hubballi, Karnataka, India
https://www.kletech.ac.in