Latency-Aware Placement of Software Defined Network Controllers

Raphael Abreu
MídiaCom Lab
Institute of Computing
Universidade Federal Fluminense - UFF
Niterói, Brazil
raphael.abreu@midiacom.uff.br

Célio Albuquerque
MídiaCom Lab
Institute of Computing
Universidade Federal Fluminense - UFF
Niterói, Brazil
celio@midiacom.uff.br

Débora Muchaluat-Saade
MídiaCom Lab
Institute of Computing
Universidade Federal Fluminense - UFF
Niterói, Brazil
debora@midiacom.uff.br

***Abstract*—The placement of multiple controllers over a distributed software defined network is challenging, to minimize maximum latency between controllers and switches. One way to solve this problem is to use the K-means algorithm to partition the original network into K subnets and position one controller at the center of every subnet. However, the value of K must be known in advance and this information may not be available to network administrators. On the other hand, latency between SDN controllers and network nodes can be known on account of agreements to QoS (Quality of Service) guarantees. Thus, in this work, we propose LAIP (Latency-Aware Incremental Partitioning), a K-means-based algorithm for placement of multiple controllers that takes into account Latency requirements to decide the number of partitions and controllers to be inserted in the network. Experiments show that LAIP has considerable performance compared to other methods. Also, results are presented on several network topologies to identify the number of controllers to be positioned to meet maximum latency constraints.**

***Index Terms*—SDN, K-means, Controller Placement, Latency**

I. Introduction

Internet-based services have become increasingly complex. Billions of communicating devices and the increasing adoption of video streaming services are often accompanied by challenges for computer networks. The traditional network can be considered inflexible for the adoption of innovation [1] since it is composed of specialized hardware (routers) that needs to be configured independently. With this in mind, network technologies are being proposed to accelerate innovation. One emerging technology is Software Defined Networking (SDN) [2].

SDN is expected to improve network performance by enabling intelligent management of its services. A fundamental concept of SDN is to decouple the data plane from the network control plane. In SDN networks, specialized switches only deal with forwarding packets and routing decision is made by a special network element called Controller. So switches can focus on performing packet forwarding at maximum speed, while the controller is in charge of defining the behavior of the switches. SDN can accelerate innovation by enabling network management in a flexible and programmable way [2].

In initial SDN proposals, the management of switches was done by only one controller. Using a single centralizing element may not be feasible on large-scale SDN networks [3]. Unlike allocating a single controller, having multiple controllers for the network is beneficial for network reliability, scalability and performance. Therefore, most SDN implementations currently focus on allowing multiple controllers in different locations to control the entire network. As an example, the prevalent SDN protocol, *OpenFlow* [4] allows for definition of multiple controllers.

Wide Area SDN Networks present a challenging scenario for allocating SDN controllers. Because of the large geographic distances between nodes, they contain links with significant delays. An initial proposal would be to allocate a controller for each node of the network. However, in this context, it is important to note that the allocation of controllers must be optimized, since the number of controllers used in the network directly impacts the operational expenses. So in these networks, it is interesting that an SDN controller can be allocated on a node to supply its network (i.e. within its area) and that it can also serve neighboring nodes.

In Wide Area SDN Networks, the distance between a node and the controller generates considerable latency and this strongly impacts network performance [3]. Thus, in SDN network planning, a central question is: where should controllers be positioned to ensure SDN network performance? This problem is known as the controller placement problem and was raised by Heller et al. [3]. This problem is NP-Difficult and to be solved for networks with large numbers of nodes, it requires an approximate algorithm. This problem can be modeled as a clustering problem called K-center and a known implementation that solves the K-center is the K-means algorithm [5].

The controller placement problem is similar to a clustering problem. Given this, versions of K-means can be used to solve it. In [6] an adaptation of K-means is presented to solve the controller placement problem. The authors propose an algorithm in which the user can provide a quantity of K controllers to partition the network. Afterward, the algorithm proceeds to incrementally partition the network into K-subnets and to position controllers on each new partition.

Proposals in this area focus on partitioning the network into K subnets and placing a controller in the center of each subnet [7], [6]. However, predetermining the number of controllers to be placed in the network is not a trivial task. Often the network hardware provisioning decision is related to QoS (Quality of Service). Applications such as video streaming or high-performance cloud computing are particularly sensitive to network delays [8]. For such applications, the maximum latency threshold for a controller may be defined by QoS agreements between provider and client.

Therefore, in this work, we propose an algorithm for controller placement in SDN networks that does not need prior information on the number of controllers to be allocated. In essence, the algorithm begins partitioning the network and placing controllers incrementally until an upper latency threshold is reached. Thus, the algorithm is expected to help network administrators obtain, in addition to the placement of their networked controllers, the number of controllers to meet a latency requirement.

The rest of the paper is organized as follows. In Section II, we introduce the problem of controller placement in SDN and we present related works in this area. In Section III, the algorithm is presented in its conceptual form. Section IV presents the implementation of the algorithm and its execution in network topologies. Section V presents results from experiments for evaluation. Finally, Section VI concludes the article listing contributions and future work.

II. Related Works

Heller et al. [3] pioneered the research on controller placement and highlighted the importance of minimizing latency between switches and controller. The main goal is to position the ideal number of controllers in the best locations of the SDN network infrastructure to achieve better performance. For this, the authors formulated the problem, which is similar to the problem of facility location [9] and proposed forms of minimization. One way is to minimize the worst-case latency, which is to position the controller so that the highest latency between this controller and a node in your network is minimized. This problem is solved by methods that solve the K-median and K-center problems. The authors propose a brute force algorithm to reach optimal controller locations, where all potential locations were evaluated.

A mathematical model for optimal controller placement is proposed in [10]. In this model, the authors take into account the ideal number of controllers, where the controllers should be placed, and the type of controller. The authors assemble simulations and try to optimize several problems in a time range of 30 hours. As the authors try to minimize an NP-difficult problem, their simulations show that only a few problems can be computed on time.

The work in [11] proposed the problem of capacitated controller placement. That is, in addition to trying to minimize latency, they also considered limited server capacity and loading dynamic traffic from switches to be served by controllers. The authors propose a greedy K-center algorithm to solve the problem.

Due to the high computational cost, heuristic strategies can be employed to solve the controller placement problem. In [12], POCO (Pareto-based Optimal COntroller placement) is presented, a framework to find optimal controller placement such that the connectivity between switches and controller is maximized by also taking into account the capacity of the controller. First, the authors propose a brute-force algorithm, but it is only valid for small networks. Then the authors propose heuristics to solve the problem of controller placement in large networks. For that, they use an algorithm based on the Pareto Simulated Annealing [13]. In this work, the authors focus on maximizing the resilience of the network. Where they considered the placement of controllers in a dynamic SDN network, in which there are variations of latency between controllers and their switches.

In [1] the problem of placing multiple controllers in SDN is addressed using an algorithm based on the Louvain heuristic method for detecting communities. The proposed placement method can adjust the number of nodes within communities to meet controller capability requirements. The authors focus on minimizing the cost of deploying within communities and maximizing network resilience across communities.

The work in [14] addresses the placement of controllers dynamically, depending on the current number of flows. The authors propose an Integer Linear Program formulation and two heuristic algorithms to solve the problem. The algorithms need to run on OpenFlow switches to obtain metrics that are used to decide where to place controllers. In order to begin the algorithm, the authors assume that the network operator has to deploy or has already deployed servers that can act as controllers at particular locations throughout the network. In this work, the authors seek to minimize flow setup time and the sum of path costs from that controller to the unassigned switches. Other metrics, such as maximum latency between node and controllers are not addressed by their work.

To reduce the problem size, in [15], a controller placement strategy is proposed, where a large SDN network is partitioned into small networks. In that paper, the authors also consider the problem of controller placement based on their capacity. The algorithm proposed by the authors provides a better result for the controller placement compared to [12]. However, in that work, the user must know in advance the number and capacity of controllers that he wants to place.

In [6], the optimized K-means algorithm for network partitioning and controller placement is presented. The algorithm seeks to minimize latency between switches and controllers. The authors adapt K-means to perform incremental network partitioning, from 1 to K. An advantage of this implementation is that it does not have the random-choice component. Since for each network partitioning, the algorithm will run the K-means until it converges and will always reach the center of this subnet. The authors perform experiments that show that the optimized K-means algorithm provides the best result in comparison to the normal algorithm based on K-means. In this work, the value of K must also be supplied by the user before the start of the algorithm.

The controller placement problem in SDN is a well researched topic. As it can be observed by most works found, the strategies vary from finding optimal solutions or using heuristics. Recently, works that aim to partition the network, especially using K-means, are of growing interest since it provides a fast response for the controller placement problem. However in works such as [15] and [6], a known number of controllers is required to position them on the network. However, it may not be easy to determine the number of controllers. Since the amount and capacity of the hardware in the network often depends on service provider agreements with users. Therefore, in this work, we propose an algorithm that partitions the network incrementally and positions controllers without prior knowledge of the number of controllers. This work extends the works on K-means based controller placement, especially the proposal in [6], to allow for incremental partitioning of the network until the desired latency requirement is reached.

III. Algorithm Proposal

The maximum latency offered, i.e., the highest latency between a node and a controller, directly impacts network performance [3]. Thus, a network administrator must gauge how many controllers are required to achieve a given maximum latency requirement. To achieve this, the network administrator can choose to run the Optimized K-means algorithm of [6] several times, and after the completion of each round, test the maximum latency obtained on the network. However, this makes the controller placement process more costly since the algorithm presented in [6] needs a known value of K for each iteration. Therefore, to find the desired maximum latency, one approach would be to perform a linear search with n attempts, requiring $\sum_{n=1}^{k}$ executions of the algorithm to obtain the network with K controllers. Thus, increasing the computational cost of the process. In this context, an algorithm to position SDN controllers taking into account the maximum latency is essential to avoid unnecessary rework to measure the number of controllers to be positioned in the network.

Based on this proposal, in this paper we present the LAIP algorithm (**L**atency-**a**Ware **I**ncremental **P**artitioning). The algorithm performs network partitioning and controller placement incrementally until the desired maximum latency is reached. LAIP consists of the following operations applied in a network topology G to meet a maximum latency requirement X:

```
Input: $G$ with latency information between nodes.
       $X$, such that $X > 0$.
1: Select-centers
2: Position-centroids
3: Compute LatMax
4: Repeat steps 1,2 and 3 while LatMax $\geq$ X
```

Algorithm 1: LAIP(G)

To start the algorithm, an existing network topology with latency information between nodes is required. This process will be described in Section IV. Note that in this work we call *center* a given node that is chosen to initiate an iteration of the K-means. We call *centroid* a node that is chosen as the center of the subnet after each K-means iteration. That is, at the end of the algorithm, the centroids are the controllers of each subnet.

The Select-centers operation is similar to the algorithm presented by [6]. In the operation, a node is found to be the center of a new destination subnet in which G will be partitioned. The destination subnet is found firstly by computing the node that has the maximum distance to its centroid. This node is selected as the center of this new subnet. The nodes which are closer to this new centroid, than any other centroid on the network, will be assigned to him. At the first iteration of the algorithm, the network does not have any centroids. Thus, a node will be randomly selected to perform K-means until convergence and find the initial centroid of network G. It is important to notice that, although the initial node chosen is random, K-means will always obtain the same initial centroid of G.

The Position-centroids operation consists of finding centroids for each subnet. At the start of the operation, the centers will be initialized by a list of previous centroid nodes plus the new node chosen in the select-centers operation. After that, the K-means algorithm is run until convergence and at the end, we get the centroids of all the subnets of G.

The Compute LatMax operation performs the maximum node latency calculation for its centroids and stores the highest latency found. The steps of the algorithm are repeated while $LatMax$ is above an upper limit X which represents the maximum latency to be obtained.

LAIP will execute partitions while the maximum network latency is above an upper limit. Once this limit is reached the algorithm ends and returns the number of controllers inserted, their positions and their corresponding subnets. Hence, only K executions of the algorithm will be necessary to obtain the number of controllers needed to meet a latency requirement defined in X.

IV. Implementation

The code that implements our proposal can be accessed in GitHub [1]. LAIP has as input a network topology file in *gml* format. Topologies with latitude and longitude information were used. With this it is possible to obtain the distance between the nodes using the *Haversine* formula [16]. Then, distance information was used to estimate latency using the following formula:

$$Latency = \frac{distance(m)}{2 \times 10^8 (m/s)}$$

Finally, the latency is inserted in the topology as the weight between nodes. With this information, the lowest latency between nodes is computed using the Dijkstra [17] algorithm. Having this information in the topology, LAIP can execute and begin partitioning the network and placing controllers.

[1]https://github.com/raphael-abreu/SDN-Kmeans-partitioning/blob/master/nb.ipynb

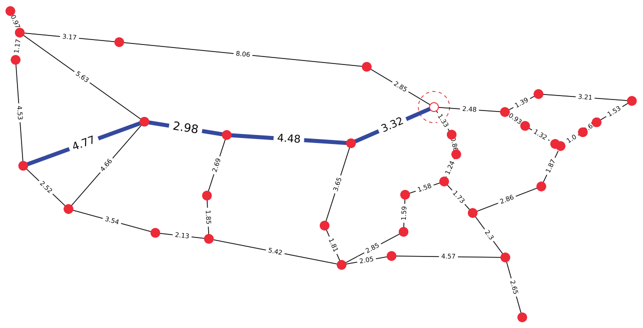


Fig. 1: OS3E with one controller. The highest latency is 15.55 ms

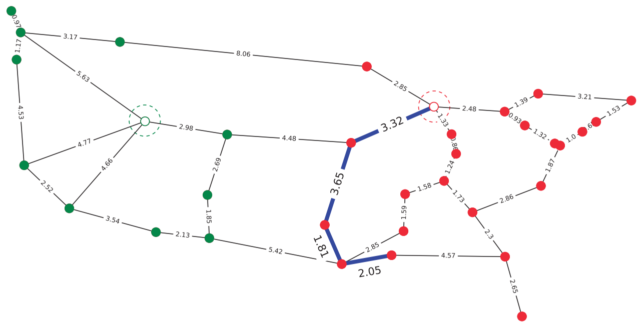


Fig. 2: OS3E with 2 controllers. The highest latency is 10.83 ms

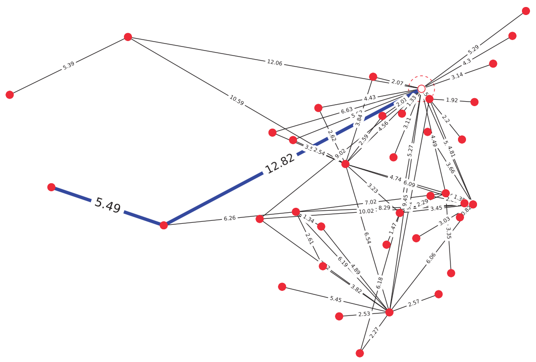


Fig. 3: Chinanet with 1 controller. The highest latency is 18.31 ms

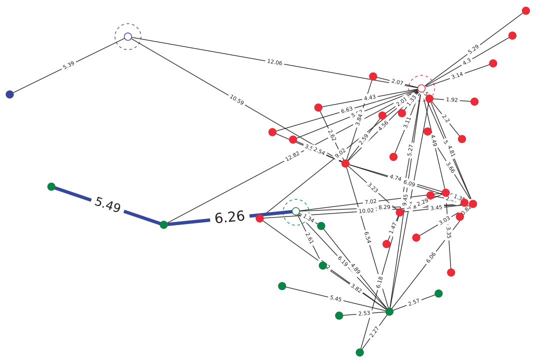


Fig. 4: Chinanet with 3 controllers. The highest latency is 11.75ms

Figures 1 and 2 present algorithm partitioning steps in the topology internet2 OS3E [18]. In the figures, the circles represent node switches. White circles with colored borders represent node switches that have become SDN controllers. The color of the circles/borders represents a subnet. The latency information is represented as weights between two nodes. It is assumed that the latency of the controller node for its network is 0. The longest (e.g., highest latency) path between a controller node and a switch is marked in blue.

Figure 1 presents the first step of the algorithm in the OS3E topology [18]. Firstly the K-means is executed for the first time to obtain the network centroid. The highest latency between a node and this centroid is computed (15.55ms) and the node that has this higher latency is saved for the next iteration of the algorithm. In Figure. 2, we can see that a new subnet has been created, represented by the green color. The algorithm used the controller node and the node found in the previous step as initial points and executed the K-means until the convergence, thus obtaining the centroids of both subnets. A new maximum latency is found for the next iteration of the algorithm. This incremental partitioning process will be repeated until it reaches the upper limit of a maximum latency defined by the user.

Figures 3 and 4 present LAIP steps in the chinanet topology [19], in order to meet a maximum latency requirement of 15ms. In Figure 3, it is possible to notice that initially a centroid (controller) is positioned in a cluster of nodes (cluster). In Figure 4, it can be seen that in addition to controllers positioned in *clusters*, it was also necessary to position a controller on a node away from the *clusters* to meet the maximum latency requirement.

Figures 5, 6 and 7 present the algorithm partition steps in the Brazilian National Research and Education Network (RNP) topology [20]. To reach a maximum latency requirement of 6ms. It took 9 controllers to achieve this maximum latency on the network. An important point to notice is that in this topology there is a wide variety of latency values between links. In this case, LAIP will partition and position controllers to create subnets on nodes that have high latency links and take advantage of low latency links to share the same controller node between them.

V. Evaluation and Results

Experiments were executed to evaluate and compare LAIP against similar methods of controller placement. In the first, we aim to compare LAIP with the standard K-means, to gauge the incremental partitioning with various runs of K-means on several topologies. In this experiment, we also identify the number of controllers required to obtain latency requirements on various topologies. Finally, another experiment is to compare LAIP against the standard K-means of Wang et al. [6] to compare the computational cost of both methods.

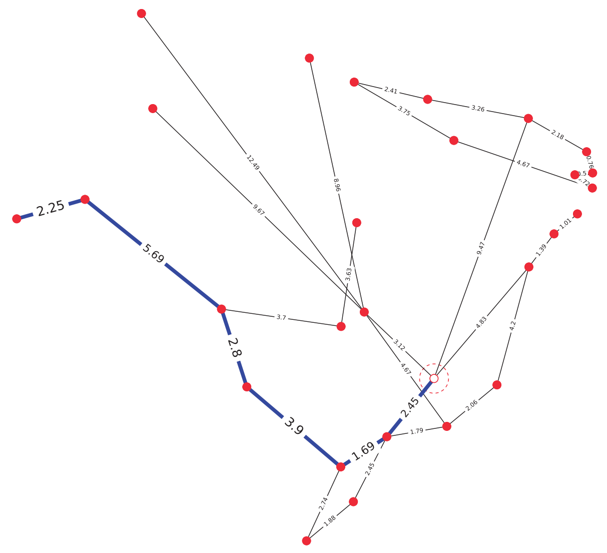


Fig. 5: RNP with 1 controller. The highest latency is 18.78 ms

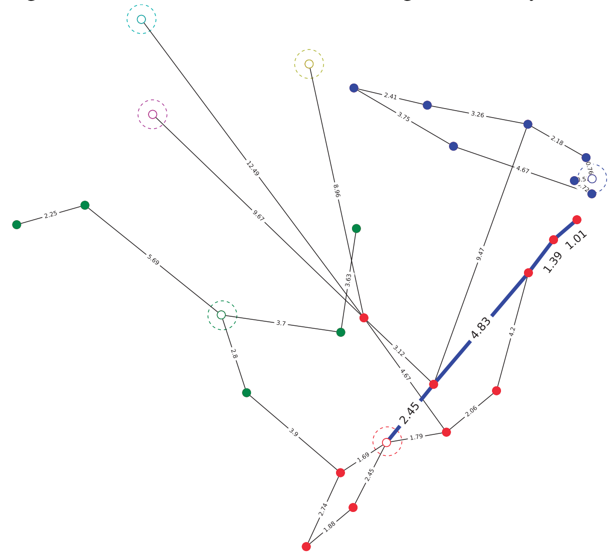


Fig. 6: RNP with 6 controllers. The highest latency is 9.68 ms

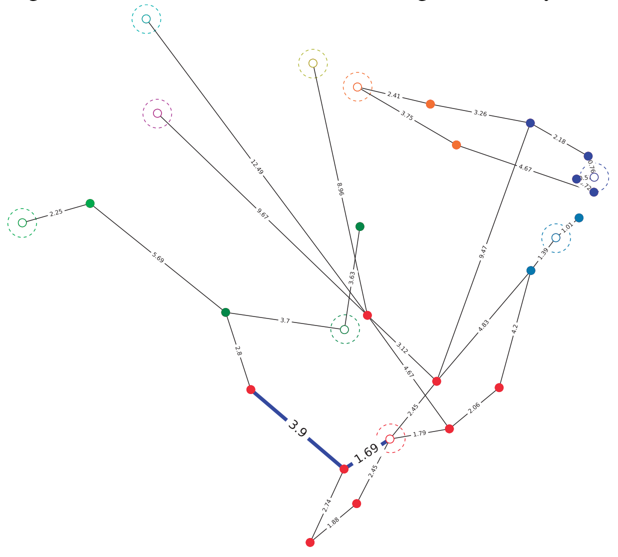


Fig. 7: RNP with 9 controllers. The highest latency is 5.59 ms

A. Comparison between LAIP and standard K-means

To evaluate the algorithm, we considered topologies in the website Topology ZOO [21] [2]. On the website, there are currently 262 topologies. Among them, 77 topologies where selected that did not contain multiple paths. In some of the remaining 77 topologies, a small fraction of nodes that do not have latitude/longitude information has been removed.

The first experiment aims at comparing the maximum latency result obtained by LAIP with the standard K-means in the 77 network topologies. In this experiment, LAIP was executed in each topology. It returned the number of controllers needed to meet the maximum latency values of 21ms, 15ms, 9ms, and 6ms. After this, the standard K-means was executed 50 times in each topology having K defined as the number of controllers found by LAIP. In Figure 8, the cumulative distribution (CDF) results for the examined topologies are presented. In the figure, the x-axis represents latency values and the y-axis represents the fraction of the topologies. While in 100% of the topologies, LAIP achieved a maximum latency below the threshold, the standard K-means often failed to achieve comparable results. Next, the execution of both algorithms in some topologies will be examined in detail.

Figure 9 presents the maximum latency CDF obtained in OS3E, Chinanet and RNP topologies for a maximum latency requirement of 6ms. For the OS3E topology, LAIP obtained 5.59ms with 5 controllers. For the Chinanet topology, LAIP got 5.49 ms with 9 controllers. For the RNP topology, LAIP obtained 5.59 ms with 8 controllers. In Figure 9, the dashed line represents the value obtained by LAIP and the solid line represents the values obtained by 100 rounds of standard K-means.

Still, in Figure 9, it can be observed that the maximum latency obtained in each round of the K-means is mostly above the maximum latency obtained by LAIP. As an example, in Figure 9 (c) it can be seen that less than 5 % of the results obtained by the standard K-means obtained a maximum latency less than or equal to 6 ms. Besides, LAIP is only executed once, since the same result is always obtained for each execution. This is due to incremental partitioning. After each partitioning, LAIP always obtains the same centroids and consequently chooses the same nodes with a maximum latency of the centroid. Therefore obtaining the same network latency response.

We use LAIP to determine how many partitions are required to meet the maximum latency requirements in various network topologies. Of the 77 topologies, 32 were removed from this evaluation because they needed only 1 controller to obtain the maximum latency of 6 ms. Figure 10 shows the results for the remaining topologies, which shows the number of controllers that must be allocated to reach a maximum latency of less than 21 ms, 15 ms, 9 ms and 6 ms in each topology.

[2]http://www.topology-zoo.org;

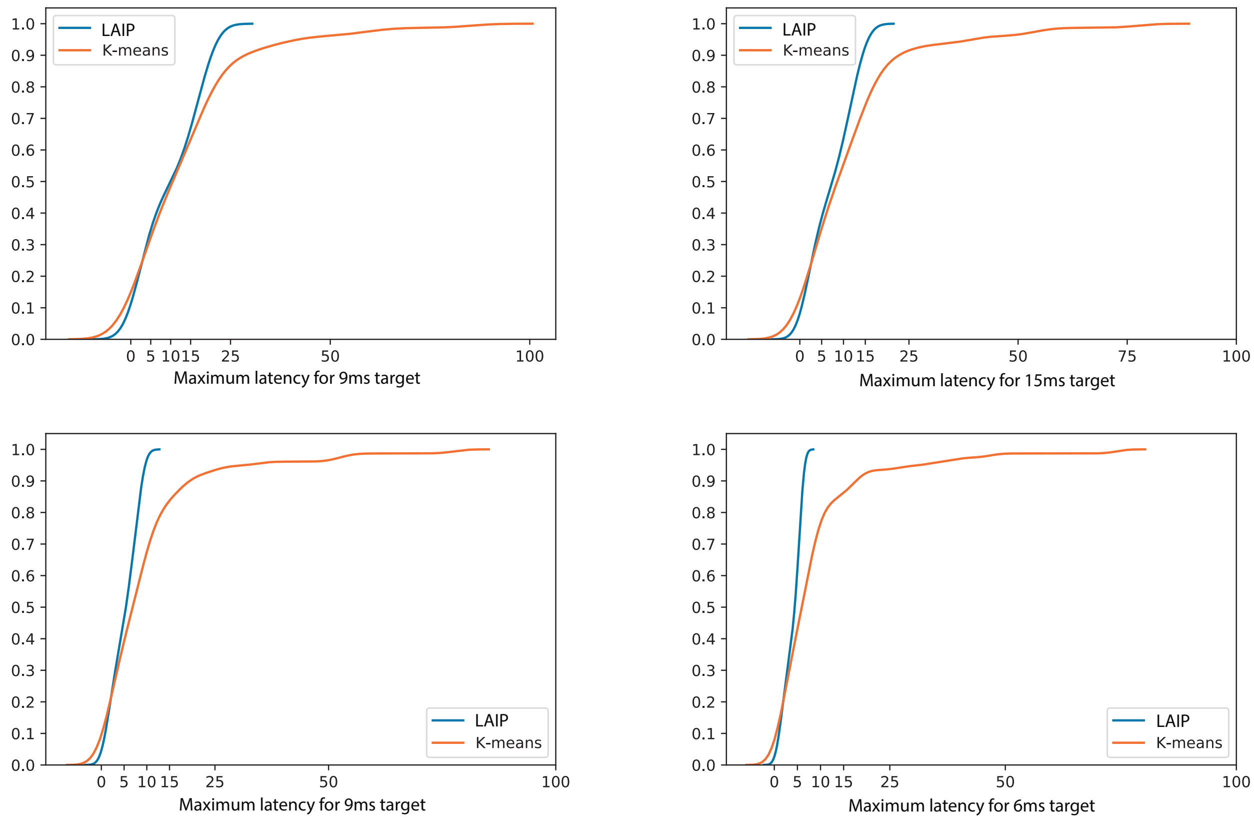


Fig. 8: CDF of maximum latencies obtained by LAIP and standard K-means in the 77 networks topologies

B. *Comparison between LAIP and Optimized K-means*

As mentioned in Section III, the Optimized K-means algorithm of [6] can be used to obtain the response of the number of controllers that are required to meet a maximum latency requirement. However, this process has a higher computational cost due to the need to define the number of K controllers in advance. In this context, the execution times of the LAIP algorithm and the optimized K-means algorithm were also examined.

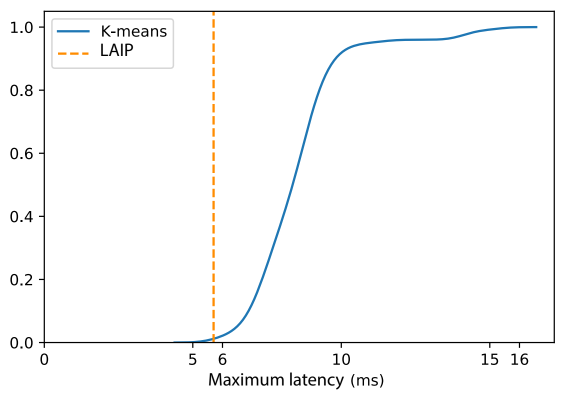
Figure 11 presents the comparative results for the Chinanet, RNP and OS3E topologies for several latency limits. In the figure, it can be observed that LAIP obtains better performance and comparison with the Optimized K-means starting at 12 ms latency in all the topologies. The results were obtained from an average of 20 runs for each experiment. The experiments were performed on a machine with the following configurations: Intel® Core^{TM} i7-4770K CPU @ 3.50GHz; 8GB RAM and Windows 10 Home 64bit Operating System.

As it can be seen in Figure 11, the computational cost in LAIP is less than the Optimized K-means. As there are smaller latency requirements, more controllers are needed to be placed. Thus, the LAIP algorithm outperforms the Optimized K-means starting at a minimum latency requirement of 12 ms.

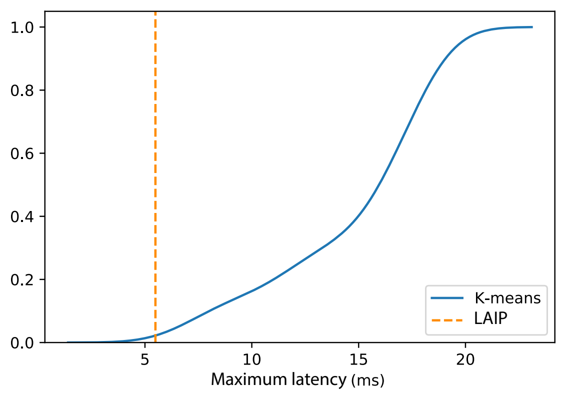
VI. Conclusion

The problem of controller placement in SDN networks is extensively researched. As seen in this work, the placement of controllers is a fundamental decision for the construction of SDN networks. In this work, we propose LAIP, an algorithm for placement of controllers based on K-means that does not need previous information of the number of controllers. With this approach, it is expected to help in the decision of where to place and about the number of required controllers to deliver the desired latency requirement. Our experiments indicate that LAIP has superior performance when compared to standard K-means and Optimized K-means. We also presented experimental results to evaluate the number of controllers needed to meet the maximum latency requirements for several network topologies.

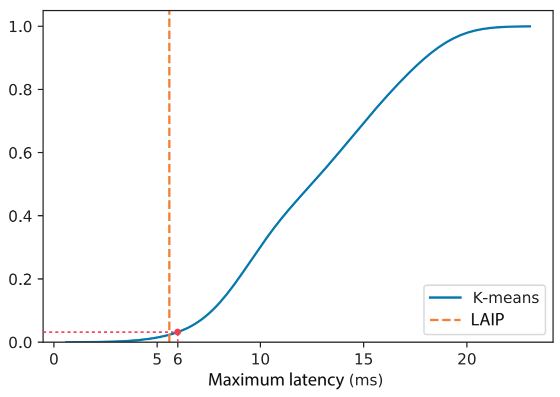
In this work, we did not consider the placement of controllers taking into account the dynamic allocation of the network. As mentioned by [7], the dynamic network allocation strategy can take a considerable time. Therefore, a promising future work is to design a controller placement algorithm that encompasses this scenario. Another future work is to investigate the usage of LAIP with other network performance metrics, such as queueing delay and throughput.



(a) OS3E



(b) Chinanet



(c) RNP

Fig. 9: CDF of maximum latencies for the topologies OS3E, Chinanet and RNP.

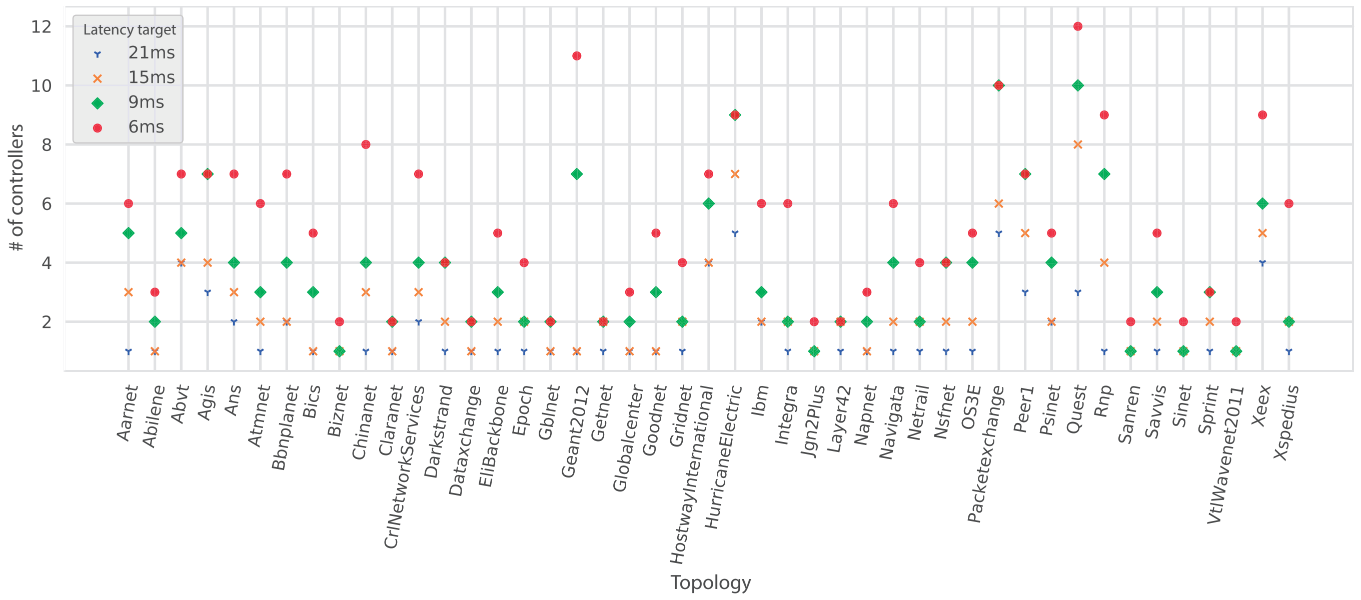
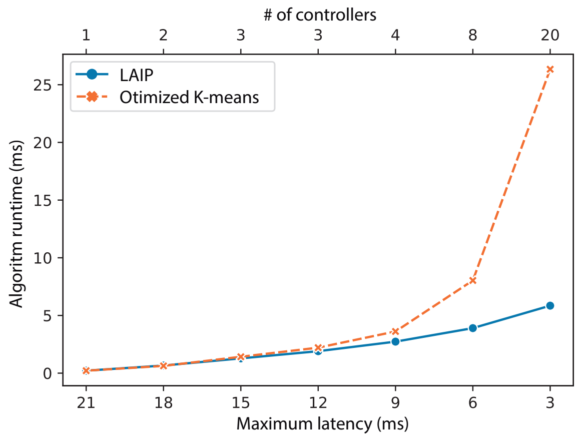
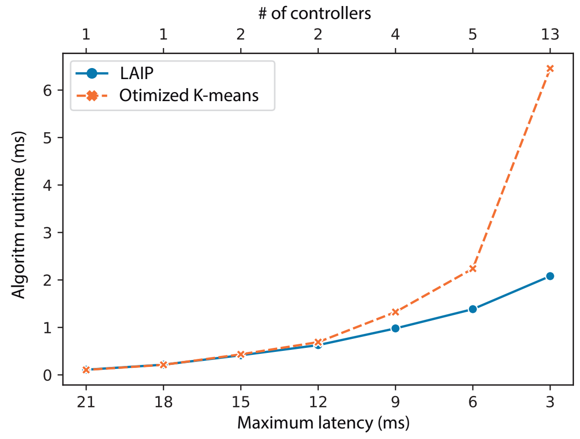


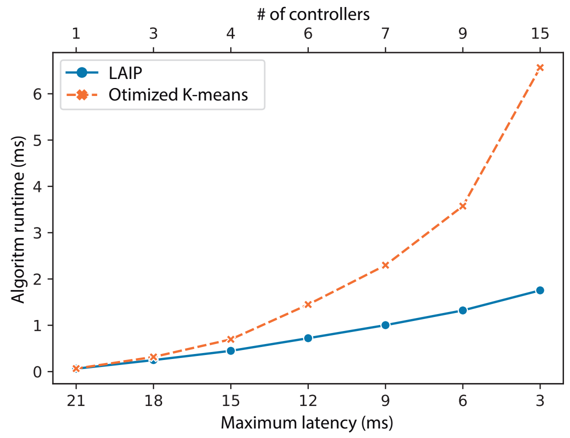
Fig. 10: Required amount of controllers to meet maximum latency of 21ms, 15ms, 9ms e 6ms.



(a) Chinanet



(b) OS3E



(c) RNP

Fig. 11: Run-time comparison between LAIP and Optimized K-means for the Chinanet, RNP and OS3E topologies.

References

[1] W. Chen, C. Chen, X. Jiang, and L. Liu, "Multi-controller placement towards sdn based on louvain heuristic algorithm," *IEEE Access*, vol. 6, pp. 49 486–49 497, 2018.

[2] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[4] 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.

[5] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

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

[7] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The sdn controller placement problem for wan," in *Communications in China (ICCC), 2014 IEEE/CIC International Conference on*. IEEE, 2014, pp. 220–224.

[8] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand, and L. S. Leong, "Realizing the quality of service (qos) in software-defined networking (sdn) based cloud infrastructure," in *2014 2nd International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2014, pp. 505–510.

[9] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *Journal of the ACM (JACM)*, vol. 48, no. 2, pp. 274–296, 2001.

[10] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE communications letters*, vol. 19, no. 1, pp. 30–33, 2015.

[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. 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.

[13] P. Czyżak and A. Jaszkiewicz, "Pareto simulated annealing," in *Multiple Criteria Decision Making*. Springer, 1997, pp. 297–307.

[14] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks." in *CNSM*, 2013, pp. 18–25.

[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 networkings," *Computer Networks*, vol. 112, pp. 24–35, 2017.

[16] C. C. Robusto, "The cosine-haversine formula," *The American Mathematical Monthly*, vol. 64, no. 1, pp. 38–40, 1957.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] Internet2, "Layer 2 services," 2019. [Online]. Available: https://www.internet2.edu/products-services/advanced-networking/layer-2-services

[19] China Telecom, "Chinanet network," 2019. [Online]. Available: https://www.ctamericas.com/resource/chinanet-network-map__trashed/chinanet-network-map-2/

[20] RNP, "Ipê network — rnp," 2019. [Online]. Available: https://www.rnp.br/en/services/connectivity/ipe-network

[21] 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.