Scalability and Reliability Aware SDN Controller Placement Strategies

Fetia Bannour, Sami Souihi and Abdelhamid Mellouk
LISSI/TincNetwork Research Team, University Paris-Est Créteil (UPEC), France

***Abstract*—The decoupling of control and data planes in Software-Defined Networking (SDN) brings benefits in terms of logically centralized control and application programming. But, the single point of management in physically centralized SDN architectures is a potential point of failure and a bottleneck that compromises network reliability and performance. Such centralized designs may also face scalability challenges especially in networks with a large number of hosts (e.g. IoT-like networks). To avoid such concerns, SDN control architectures are usually designed as physically distributed systems. This raises practical challenges about the best approach to decentralizing the control plane while maintaining the logically centralized network view. In particular, determining the number of controllers and locating them in the network is a hard task that should be addressed appropriately. This paper proposes two novel strategies that cover different aspects of the controller placement problem with respect to performance and reliability criteria. These strategies use two types of heuristics that are compared and assessed on large-scale topologies to provide operators with guidelines on how to find their optimal controller placement that meets their specific needs.**

I. Introduction

The rise in Virtualisation and Cloud Computing, the surge in mobile traffic patterns along with the emergence of Big Data and Internet of Things (IoT) trends revealed the difficulties faced by legacy networks in meeting the stringent requirements of modern users. These challenges are inherent to the inflexible and static nature of the Internet's traditional architecture which makes it poorly-suited to today's dynamic, interactive and hyper-scale network environments. In traditional networks, the control logic is purely distributed and most network functions are implemented in dedicated appliances. This makes the classical approach to policy enforcement and network management a hardware-centric and time-consuming task that relies on error-prone manual configuration or device-specific scripting across the heterogeneous bundle of proprietary devices.

In this context, SDN [1] emerged as a networking paradigm that promises to make the traditional architecture evolve into a scalable platform that is more responsive to changing requirements and more agile for advanced services. SDN proposes to raise the level of abstraction by decoupling software (the control plane) from hardware (the data plane) enabling their independent evolution. By breaking the vertical integration of closed devices which has been a barrier to open innovation, SDN allows to program the control logic in a high-level vendor-agnostic fashion. SDN eases the development of fine-grained, automatic and adaptive control features. It aims for network control centralization, offering improved visibility and flexibility to manage the network and optimize its resources.

Beyond the hype, there have been concerns about the widespread adoption of SDN. Studies [2], [3] on the feasibility of SDN deployments on real topologies proved that the physical centralization of the network intelligence in one software element, referred to as the SDN controller, increases the risk of a Single Point of Failure (SPOF) compromising network reliability and availability. With such resilience requirements in mind and based on use cases, these works motivated the physical decentralization of the control plane into controllers in charge of handling the network while maintaining its logically centralized view. From our perspective, in addition to meeting the reliability requirements expressed in [1]–[4], the SDN decentralization is required to achieve scalability [5]. Indeed, centralized designs may face scaling issues while struggling to match the stringent needs of IoT-like networks that witness a drastic growth in the number of connected hosts.

While the need for a distributed SDN architecture was acknowledged by the SDN community [1], [5], the best approach to designing a scalable and reliable distributed control plane is highly debatable given the challenges brought by such distributed systems. In particular, *the SDN Controller Placement Problem* investigates the required number of controllers and their appropriate locations according to specific criteria. Our controller placement optimization scheme compares two types of heuristics with low computation time. It explores their potential to handle large-scale or dynamic IoT-like environments where fast reevaluations of placement configurations are needed to adapt in real-time to frequently-changing conditions.

Outline: In this paper, we propose two context-based strategies that cover different aspects of the controller placement problem with respect to reliability and performance criteria. In Section II, we give an overview of state-of-the-art works. In Section III, we review the optimization problem and investigate the involved metrics. In Section IV, we show our heuristic approaches to tackling that problem. Section V displays and discusses the results before elaborating on future perspectives.

II. Related Work

Heller et al. [2] motivated the SDN Controller Placement Problem (CPP) and studied *how many* controllers are needed and *where* in the network they should be placed. They argued that in most topologies one controller is enough for meeting latency requirements but insufficient for achieving resilience. They treated the placement of k controllers as a variant of the *facility location problem*. Their work was extended by [3] to include resilience aspects. Hock et al. introduced the resilient Pareto-based Optimal COntroller placement (POCO) framework that generates Pareto-optimal CPP solutions with various trade-offs between quality (latency) and resilience.

While the first version of POCO was intended for small to medium sized networks, a subsequent version [4] comprised a *heuristic-based* Multi-Objective Combinatorial Optimization (MOCO) approach called Pareto Simulated Annealing (PSA) for large-scale networks. When evaluating that approach on

978-3-901882-98-2

real topologies from the Internet Topology Zoo where the network size ranges from 5 to 50 nodes, the authors only stress the diameter aspect of large-scale networks and do not assess their scalability in terms of an increased number of nodes.

Yao et al. [6] studied another variant of the CPP that includes the load on controllers in addition to latency considerations (*the capacitated k-center problem*). The authors of [4] explored in [7] the potential of specialized heuristics to solve the capacitated problem in large-scale SDNs by developing the Pareto-Capacitated k-Medoids (PCKM). Such a specialized heuristic that optimizes case-specific metrics (i.e. the average latency and the load imbalance) was compared to generic heuristics destined for arbitrary optimization purposes.

Similarly, Ahmadi et al. [8] formulated the CPP in large-scale networks as a MOCO problem and used a multi-objective heuristic called the Non-dominated Sorting Genetic Algorithm (NSGA-II) to find good and diverse approximate Pareto Optimal solutions with respect to competing criteria. Their work provided an extensive analysis of the trade-offs between many combinations of reliability and performance metrics.

III. The Controller Placement Problem

A. Problem Formulation

The *Controller Placement Problem* consists in finding the required *number* and the appropriate *locations* of controllers (among all nodes) that efficiently partition the network into domains to achieve the best trade-off between multiple criteria. We formulate it as a multiobjective optimization problem. The network is represented by a graph $G = (V, E)$ where the nodes V are the controllers and switches while the edges E are the links between nodes. Edge weights represent the shortest-path latencies between each pair of nodes. This information is stored in the *Logical Topology Map* (IV-A) where $d(s, c)$ is the latency from a switch $s \in V$ to a controller $c \in V$.

B. Placement Metrics

Performance criteria: Optimizing for control plane performance is of great importance in large-scale IoT-like networks with stringent response-time requirements and where high propagation delays lead to inconsistent and incorrect behaviors of services. In particular, the average latency (Avg-s2c-Lat) and maximum latency (Max-s2c-Lat) between controllers and their assigned switches for a placement C of k controllers among $n = |V|$ nodes are two different latency-related performance metrics introduced by [2]. Unlike the average latency (1) that evaluates the overall quality of the network performance from a controller-to-switch latency point of view while hiding single cases of unacceptably high latencies, the maximum latency (2) is useful in preventing the occurrence of such high-latency cases in placement scenarios.

$$\pi^{Avg-s2c-Lat}(C) = \frac{1}{n} \sum_{(s \in S)} \min_{(c \in C)} d(s, c) \quad (1)$$

$$\pi^{Max-s2c-Lat}(C) = \max_{(s \in S)} \min_{(c \in C)} d(s, c) \quad (2)$$

The controller capacity-awareness is another important performance factor that can be considered in our problem to avoid the chance of controller overload and prevent the related performance issues (e.g. additional delays at the controller). One possible load balancing scheme is to use the shortest-path controller-to-switch assignment method but introduce a load imbalance metric to be minimized through the controller placement optimization (3). This metric is defined by [3] as the difference between the maximum and the minimum no. of nodes n_c assigned to a controller c for a given placement C.

$$\pi^{Load-Imbalance}(C) = \max_{(c \in C)} n_c - \min_{(c \in C)} n_c \quad (3)$$

Reliability criteria: The SDN control-to-data plane separation feature brings new concerns about network reliability, a crucial requirement for operational SDNs. Notably, a key consideration should be given to improving the reliability of the SDN control plane when designing distributed platforms. That aspect of SDN reliability can be ensured by placing controllers in a reliability-aware manner that mitigates the impact of controller failures. A common mechanism for recovering from the primary controller failure is to assign the concerned orphan switches to the closest working controllers. In doing so, response-time requirements should be met to guarantee controller fault-tolerance. Indeed, the propagation latencies of orphan switches with respect to their new controllers should remain acceptable. As an indicator of reliability against controller failures, we use the maximum latency metric (to be minimized) that is computed based on the latencies between the switches and all subsets of working controllers C_1 for a placement C according to the considered controller failure scenarios F as below:

$$\pi_F^{Max-s2c-Latency}(C) = \max_{(s \in S)} \max_{(C_1 \subseteq C)} \min_{(c \in C_1)} d(s, c) \quad (4)$$

Among these controller failure scenarios, the worst-case scenario for a network switch would be the simultaneous failure of the $(k-1)$ closest SDN controllers. Mitigating that control plane failure scenario implies minimizing the maximum of the latencies between the switches s and their respective furthest functional controllers $C_{Fu}(s)$ as follows:

$$\pi_{F(k-1)}^{Max-s2c-Latency}(C) = \max_{(s \in S, c \in C_{Fu}(s))} d(s, c) \quad (5)$$

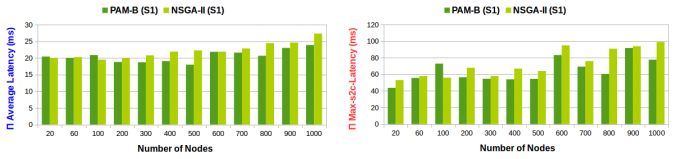
In practice, it is more common for primary controller failures to occur one at a time. Thus, reducing the impact of that controller failure scenario entails minimizing the maximum of the latencies between the switches s and their respective second closest controllers $C_{Cl}(s)$ as expressed in the following:

$$\pi_{F(1)}^{Max-s2c-Latency}(C) = \max_{(s \in S, c \in C_{Cl}(s))} d(s, c) \quad (6)$$

IV. The Proposed Method

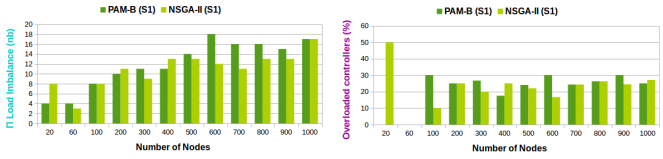
A. The Adopted Approach

We follow a two-phase approach to modeling the placement problem using a decentralized simulation framework. In the first stage, we deploy mechanisms to gather and transmit information about the network topology. This information is used by the placement algorithms we implemented in the second stage. After running a leader election scheme, followers send their neighborhood information (latency) to their leaders. Then, leaders synchronize such cluster information and build the *Global Physical Topology Map*. One leader is nominated as the *Hyper Leader* that will be in charge of running the Dijikstra shortest-path algorithm and building the *Global Log-*



(a) (b)

Fig. 1: Strategy 1: Latency-based Performance Metrics



(a) (b)

Fig. 2: Strategy 1: Load Imbalance

ical Topology Map. At the Hyper level, controller placement optimization algorithms are run based on that global view and given a number of controllers k. Placement solutions are then analyzed to find the best trade-off between the objectives.

B. Multi-criteria Placement Algorithms

To optimize the placement of k controllers, we use different algorithms: a clustering algorithm we developed based on PAM (PAM-B) and a genetic algorithm (NSGA-II). PAM (Partitioning Around Medoids) [9] is a k-Medoid method that partitions the data set of N objects (nodes) into k clusters represented by k medoids (controllers). The idea of PAM is to find the set of medoids that improve the overall quality of clustering which is measured based on the average dissimilarity of all objects to their nearest medoid. In our case, our metrics M are of equal importance making the dissimilarity function D for a given placement $C \in CP$ (the placement configurations) computed as the normalized sum of all weighted objectives O with the associated weights equal to $\frac{1}{M}$:

$$D^{PAM-B}(C) = \sum_{i \in M} (\tfrac{1}{M}) \times N(O_i) \tag{7}$$

$$where: N(O_i) = \frac{O_i(C) - \min\limits_{(C \in CP)} O_i(C)}{\max\limits_{(C \in CP)} O_i(C) - \min\limits_{(C \in CP)} O_i(C)}$$

NSGA-II used in [8] for addressing the same problem, is a popular fast and elitist genetic algorithm for multi-objective optimization. In addition to the classical genetic operators (crossover and mutation), NSGA-II uses other ranking mechanisms (non-dominated sorting and the crowding distance) for creating the next generation population of candidate solutions. The main idea of NSGA-II is to make that population evolve towards a set of non-dominated solutions (the Pareto front) offering the best trade-offs between the considered objectives.

C. Our Strategies

Strategy 1: *Performance-based metrics*

Strategy 1 solves the SDN controller placement problem based on the latency-related performance metrics shown in (1) and (2) while following the usual shortest-path controller-to-switch assignment method. We also adopt a load balancing scheme using the load imbalance metric (3) proposed by [3] and we evaluate the controller overload risk. Accordingly, the multi-objective NSGA-II is launched with these three objectives to be minimized while PAM-B minimizes the following dissimilarity function as a normalized equally-weighted sum of the three considered objectives in accordance with (7):

$$D_1^{PAM-B}(C) = Avg(N(\pi^{Avg-Latency}(C)), N(\pi^{Max-Latency}(C)), N(\pi^{Load-Imbalance}(C))) \tag{8}$$

Strategy 2: *Strategy 1 with reliability metrics*

Strategy 2 provides a placement framework that includes reliability metrics along with performance metrics. As for reliability metrics (4), users of the framework have the option of adding a reliability metric variant that tackles the worst-case controller failure scenario (5) or a variant that addresses a more common controller failure scenario (6) besides the performance metrics (1), (2) and (3). The dissimilarity functions of PAM-B for both variants are computed in accordance with:

$$D_2^{PAM-B(k-1)}(C) = Avg(N(\pi^{Avg-Latency}(C)), N(\pi^{Max-Latency}(C)), N(\pi^{Load-Imbalance}(C)), N(\pi^{Max-Latency}_{F(k-1)}(C))) \tag{9}$$

$$D_2^{PAM-B(1)}(C) = Avg(N(\pi^{Avg-Latency}(C)), N(\pi^{Max-Latency}(C)), N(\pi^{Load-Imbalance}(C)), N(\pi^{Max-Latency}_{F(1)}(C))) \tag{10}$$

V. Performance Evaluation

A. Simulation Settings

We use the JAVA-based framework Sinalgo (Simulator for Network Algorithms) for implementing our two-phase approach (IV-A) and assessing our multi-criteria placement algorithms (IV-B) based on our strategies (IV-C) and according to different simulation scenarios. In NSGA-II, the *maximum number of objective function evaluations* simulation parameter (MaxEvaluations) is used as a stopping criterion. The values of that parameter (from 20000 to 140000) depend on the number of the objectives involved in each strategy (3 to 4) and the size of the network (from 20 to 1000 nodes) in each scenario.

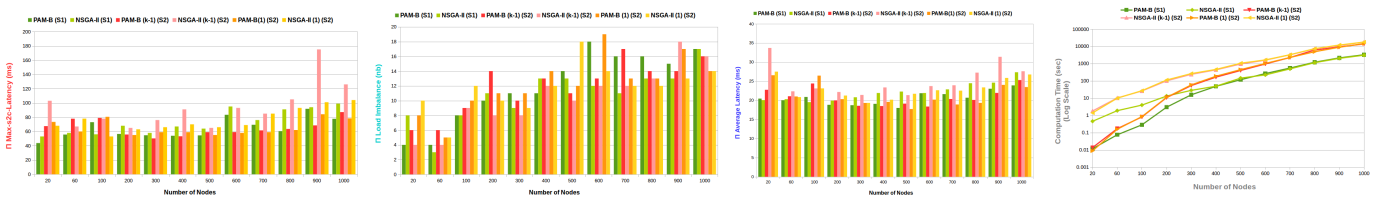
B. Simulation Results

For each strategy, given a topology, we assess the placement solutions of each algorithm. PAM-B generates a clustering solution based on the equally-weighted dissimilarity measure (7). Likewise, in NSGA-II, we consider the fairest placement solution with respect to our criteria among the generated non-dominated Pareto Optimal solutions representing the possible trade-offs between the objectives. This is achieved by selecting the Pareto solution S that best reduces the total gap between its objective values M and their respective optimal values across the set of Pareto optimal solutions P. This corresponds to the Pareto solution with the minimum value of the measure below:

$$a(S) = \sum_{i \in M} (\tfrac{1}{M}) \times \frac{O_i(S) - \min\limits_{(S \in P)} O_i(S)}{\max\limits_{(S \in P)} O_i(S) - \min\limits_{(S \in P)} O_i(S)} \tag{11}$$

Several simulation scenarios are performed following the considered strategies and using various types of topologies of different size; from **20** up to **1000** nodes. That allowed us to compare our placement strategies, analyze the performance of our optimization algorithms and study the scalability of our approach that is mainly intended for large-scale deployments.

As shown in Figure 1, PAM-B is better than NSGA-II at minimizing the Average Latency (6,68 % better on average) and the Maximum Latency (10,53 % better on average) over all scenarios. Besides targeting a certain level of latency performance, Strategy 1 considers load balancing as another ma-

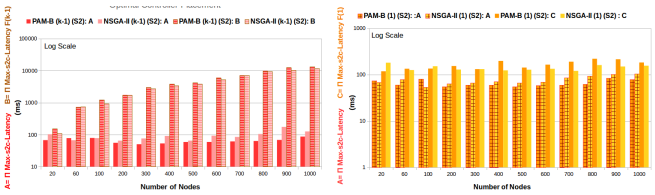


(a) (b) (c)

Fig. 3: Strategy 2: Performance Metrics

Fig. 4: Computation Time

jor performance factor. But, minimizing the Load Imbalance metric (2(a)) does not eradicate the risk of controller overload as illustrated by Figure 2(b) that depicts the % of overloaded controllers over all scenarios. For example, when the network size is equal to 800 (2(b)) and the no. of controllers is equal to 80, both PAM-B and NSGA-II produced configurations where 21 (26,25%) of these controllers are overloaded. Strategy 2 involves reliability metrics in addition to the performance metrics of Strategy 1 (IV-C). Figure 5 compares, for each variant of Strategy 2 according to all placement solutions, the values of the Maximum Latency in the failure free case with that of the Maximum Latency in the considered failure case. It shows that PAM-B and NSGA-II perform in a quite similar fashion when optimizing these metrics. Figure 3 studies



(a) (b)

Fig. 5: Strategy 2: Reliability Metrics: (Maximum Latencies in Failure free & Failure case scenarios)

the performance cost of involving reliability criteria in the controller placement. Surprisingly, optimizing for reliability metrics did not severely impact performance metrics like the Maximum Latency (in the failure free case) (3(a)) whose values remained acceptable and comparable to that in Strategy 1 except for a few placement scenarios that were in most cases produced by NSGA-II. Likewise, similar trends are observed across Strategy 1 and 2 for each of the Load Imbalance (3(b)) and the Average Latency (3(c)) performance metrics of the obtained placement configurations. When the network size is equal to 1000 (3(c)), PAM-B(k-1) (respectively PAM-B(1)) according to Strategy 2 produced a configuration where the value of the Average Latency metric is equal to 25,3ms (respectively 23,5ms) compared to 24ms for PAM-B according to Strategy 1. In the same scenario, NSGA-II(k-1) (respectively NSGA-II(1)) according to Strategy2 generated a configuration with an Average Latency value equal to 27,6ms (respectively 26,8ms) against 27,3ms for NSGA-II with respect to Strategy1.

C. Discussion

Assessing PAM-B and NSGA-II over our strategies revealed that in most scenarios PAM-B outperforms NSGA-II in terms of the quality of final solutions. PAM-B gives a balanced trade-off between performance and reliability metrics and also stable results over both strategies whereas the performance of NSGA-II with respect to these metrics is sometimes unpredictable and dependent on the strategy. The runtime comparison between PAM-B and NSGA-II (Figure 4) showed similar trends over the scenarios. Their computational complexities are indeed close and respectively equal to $O(k(n-k)^2)$ (k is the no. of medoid clusters (controllers) and n is the no. of objects (the network size)) and $O(MN^2)$ (M is the no. of objectives and N is the population size). It is worth noting that the computation time of PAM can be improved. CLARA [9], a sampling-based variant of PAM, is recommended for large data sets. Its complexity is $O(ks^2 + k(n-k))$, where k is the no. of controllers, n is the network size and s is the sample size.

VI. Conclusion

We studied the controller placement optimization problem in large-scale IoT-like SDNs. Multiple performance and reliability metrics were considered based on the context. In these strategies, two heuristics were put forward to find high-quality approximate solutions in a reasonable computation time: A *Clustering approach* and a *Genetic approach*. Our results showed the potential of clustering methods in delivering proper solutions that achieve balanced trade-offs among the competing performance and reliability criteria. The challenge of determining the number and locations of controllers is one particular aspect of the distributed SDN control problem. The second key aspect calling for further investigation is the knowledge sharing challenge facing these logically-centralized platforms. Given the correlation between placing controllers and modeling the traffic among them, it becomes essential to reassess the placement parameters after studying the consistency models used for SDN inter-controller communication.

References

[1] F. J. Ros and P. M. Ruiz, "On reliable controller placements in software-defined networks," *Comput. Commun.*, vol. 77, pp. 41–51, Mar. 2016.

[2] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop HotSDN*. ACM, 2012, pp. 7–12.

[3] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in *ITC*, 9 2013.

[4] 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 TNSM*, vol. 12, no. 1, pp. 4–17, 2015.

[5] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-scale dynamic controller placement," *IEEE TNSM*, pp. 63–76, March 2017.

[6] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," in *IEEE Commun. Lett.*, 2014.

[7] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia, "Specialized heuristics for the controller placement problem in large scale sdn networks," in *Proc. ITC*, Sept 2015, pp. 210–218.

[8] V.Ahmadi, A.Jalili, S.M.Khor, and M.Keshtgari, "A hybrid nsga-ii for solving multiobjective controller placement in sdn," in *KBEI*, 2015.

[9] K. Shraddha and M. Emmanuel, "Review and comparative study of clustering techniques," *IJCSIT*, pp. 805–812, 2014.