The Role of the Inter-Controller Consensus in the Placement of Distributed SDN Controllers

Tianzhu Zhang, Paolo Giaccone, Andrea Bianco, Samuele De Domenico
Dept. Electronics and Telecommunications, Politecnico di Torino, Italy

Abstract

We consider a distributed Software Defined Networking (SDN) architecture adopting a cluster of controllers to improve network scalability and reliability. Different from previous works that focused solely on the control traffic exchanged between controllers and switches, we additionally consider the control traffic exchanged among the controllers (e.g., the control traffic to keep the shared data structures synchronized). We advocate a careful placement of the controllers, which should take into account both the two kinds of control traffic. We evaluate, for some real ISP network topologies, the delay tradeoffs for the controller placement problem and we propose a novel evolutionary algorithm to find the corresponding Pareto frontier. Furthermore, we develop a simple model to estimate the reaction time perceived by the switches, which is accurately validated in an operational Software Defined WAN (SDWAN). We also formalize the optimization problem to minimize the reaction time and devise a novel approximation algorithm, whose performance is assessed against the optimal solver in real ISP network topologies.

Our work provides novel quantitative tools to optimize the planning and the design of the network supporting the control plane of SDN networks, especially when the network is very large and in-band control plane is adopted. We also show that for operationally distributed controllers (e.g., OpenDaylight and ONOS), the location of the controller that acts as leader in the consensus algorithm has a strong impact on the reactivity perceived by the switches.

Keywords: Software Defined Networking; distributed controllers; controller placement problem; consistency data models.

1. Introduction

The centralized network control of the Software Defined Networking (SDN) paradigm, which enables the development of advanced network applications, poses two main issues. First, limited reliability due to the single point-of-failure. Second, the control traffic between the switches and the controller concentrates on a single server whose processing capability is limited, raising scalability issues. Distributed SDN controllers are designed to address the above issues, while preserving a logically centralized view of the network state necessary to ease the development of network applications. In a distributed architecture, multiple controllers are responsible for the interaction with the switches. Thus, the processing load at each controller decreases, because the control traffic between the switches and the controllers is distributed, with a beneficial load balancing effect. Furthermore, resilience mechanisms can be implemented to improve network reliability in case of controller failures.

Distributed controllers adopt coordination protocols and algorithms to synchronize their shared data structures and to enable a centralized view of the network state for the applications. These schemes follow a consensus-based approach in which coordination information is exchanged among controllers to reach a common network state. This induces some delay that, as discussed in Sec. 2.2 and shown experimentally in Sec. 4.3, can heavily affect the controller reactivity perceived at the switches. Indeed, any read/write of a shared data structure at a controller is directed to a possibly different controller acting as "owner" of the data. Thus, the controller-to-controller delays must be added to the switch-to-controller delays when evaluating the controller's reactivity perceived at the switches. As a consequence, the optimal placement of the controllers on the network topology must consider not only the delays between the switches and the controllers, but also the delays between controllers. Most of the past literature concentrated on the Openflow-based interaction, thus con-

sidering the switch-to-controller delays only, and neglecting the controller-to-controller delays, which are instead the focus of our work.

The adoption of distributed SDN controllers in Wide Area Networks (SDWANs) is more challenging than in data centers. In the latter scenario, the limited physical distances between network devices permits the installation of a separate network among the controllers (e.g., using dedicated Ethernet or InfiniBand connections), providing an out-of-band control plane. Conversely, SDWANs adopt an in-band control plane: the control packets and data packets share the same network infrastructure. Furthermore, the problem of supporting a responsive controller-to-controller interaction is exacerbated by the SDWAN's geographical extent.

Our contributions

In this paper, we address two complementary views of the controller placement problem. First, we study optimal placements in terms of both controller-to-controller delays and switch-to-controller delays, referring to the concept of Pareto optimality. This permits to understand the possible tradeoffs independently of the applications running on the network. Second, we consider specifically a reactive forwarding application and optimize the controller placement to minimize the reaction time perceived by the switches.

In more detail, we provide the following novel contributions:

1. we discuss Pareto-optimal controller placements considering controller-to-switch and controller-to-controller delays for WAN topologies adopted in some real ISP networks, based on the adopted data-ownership models (as defined later);
2. we propose a low-complexity algorithm to find the approximated Pareto frontier in large networks;
3. we propose some formulas to compute the reaction time perceived by the switches when interacting with the controllers, due to the inter-controller control traffic, for a reactive network application;
4. we validate the previous formulas with traffic measurements on an operational SDWAN;
5. we define an Integer Linear Programming (ILP) problem to find the optimal placement that minimizes the reaction time perceived at the switches;
6. we propose an approximation algorithm to solve the above ILP problem and assess its performance in real ISP networks.

The experimental validation of point 4 highlights the practical validity of the adopted methodology, to find either all the Pareto optimal placements or the single optimal placement for a network application.

The paper is organized as follows. In Sec. 2 we provide an overview of distributed SDN architectures, providing the main background to understand the role of the controller-to-controller communications and the different data-ownership models. In Sec. 3 we discuss the related work. Sec. 4 is devoted to presenting all the main contributions of our work. Finally, we draw our conclusions in Sec. 5 .

2. Distributed SDN controllers

Many architectures have been proposed to support distributed SDN controllers, with the goal of improving scalability and/or ensuring reliability. In this paper, we only concentrate on some specific architectural aspects. For more details, the reader can refer to the extensive survey in [1]. In distributed controllers, two control planes can be identified. First, the switch-to-controller plane, denoted as *Sw-Ctr plane*, supports the interaction between any switch and its controller (denoted as *master controller* for a given switch) through the controller's "south-bound" interface. This interaction is usually devoted to data plane commands (e.g., through the OpenFlow (OF) [2] protocol) as well as to configuration and management of network switches (e.g., through OF-CONFIG or OVSDB protocols). Second, the controller-to-controller plane, denoted as *Ctr-Ctr plane*, permits the direct interaction among the controllers through the controller's "east-west" interface. Indeed, the controllers exchange heart-beat messages to ensure liveness and to support resilience mechanisms. In addition, controllers also need to *synchronize the shared data structures* to guarantee a consistent global network view.

2.1. Data consistency models

The traffic on the Ctr-Ctr plane is crucial to achieve a consistent shared view of the network state, which is the required condition to correctly run network applications. The network state (e.g., topology graph, the mapping between any switch to its master controller, the list of installed flow rules) is stored in shared data structures, whose consistency across the SDN controllers can be either *strong* or *eventual*. Strong consistency implies that simultaneous reads of some data occurring in different controllers always lead to the same result. Eventual consistency implies that simultaneous reads may

eventually lead to different results, for a transient period. Different levels of data consistency heavily affect the availability and resilience of the controller, as the well-known CAP theorem highlights [3, 4]. In a nutshell, anytime a data structure is shared across the controllers, they must synchronize through a consensus algorithm that guarantees a consistent view of the data in case of updates. In general, a consensus algorithm is very complex, to deal with all possible failures and network partitions, and it is tailored to a specific level of data consistency. Each controller is required to interact with the other controllers through the Ctr-Ctr plane, introducing some latency to synchronize their internal data structures.

In both OpenDaylight (ODL) [5] and Open Network Operating System (ONOS) [6], two of the most relevant SDN controllers, strong consistency for the shared data structures is achieved by the recently proposed Raft consensus algorithm [7]. Indeed, the most recent versions of ODL (e.g., Beryllium) provide a clustering service to support multiple instances of the controller, and the clustering module can be built with a Raft implementation [8], whose code is available in [9]. ODL clustering service organizes the data of different modules into shards. Each shard is replicated to a configurable odd number of ODL controllers. Similarly, the most recent versions of ONOS (2015-17) adopt the Raft algorithm for distributed data store creation and mastership maintenance [10, 11], according to which data is shared across different shards. Each shard is managed by an independent instance of the Raft algorithm: this ensures that operations on different shards can proceed independently.

The Raft consensus algorithm is based on a logically centralized approach, since any data update is always forwarded to the controller defined as data structure *leader*, which is unique across the cluster of controllers. Then, the leader propagates the update to all the other controllers, defined as *followers*. The update is considered committed whenever the majority of the follower controllers acknowledge the update. Sec. 4.2.2 will describe in more detail the adopted protocol, based on the description provided in [7]. Note that the role of leader/follower controller for a data structure is independent of the role of master/slave controller for a switch.

In ONOS data can also be synchronized according to an eventual consistent model, in parallel to strong-consistent data structures. Eventual consistency is achieved through the so called "anti-entropy" algorithm [11, 12], according to which updates are local in the master controller and propagated periodically in the background with a simple gossip approach: each controller picks at random another controller, compares the replica and eventual differences are reconciled based on timestamps.

2.2. Data-ownership models and delays tradeoff

The controller reactivity as perceived by a switch depends on the availability of the data necessary for the controller. We can identity two distinct operative models.

In a *single data-ownership* (SDO) model, a single controller (denoted as "data owner") is responsible for the actual update of the data structure. Any read/write operation on the data structures performed by any controller must be forwarded to the data owner. In this case, the Ctr-Ctr plane plays a crucial role for the interactions occurring in the Sw-Ctr plane, because some Sw-Ctr request messages (e.g., `packet-in`) trigger transactions with the data owner on the Ctr-Ctr plane. Thus, the perceived controller reactivity is also affected by the delay in the Ctr-Ctr plane. As discussed in Sec. 2.1, the SDO model is currently adopted in ODL and ONOS, for all the strong-consistent data structures managed by the Raft algorithm: a local copy of the main data structures is stored at each controller, but any read/write operation is always forwarded to the leader. With this centralized approach, data consistency is easily managed and the distributed nature of the data structures is exploited only during failures.

In a *multiple data-ownership* (MDO) model, each controller has a local copy of the data and can run locally read/write operations. A consensus algorithm distributes local updates to all other controllers. This model has the advantage of decoupling the interaction in the Sw-Ctr plane from the one occurring in the Ctr-Ctr plane, thus improving the reactivity perceived by the switch. The main disadvantage is the introduction of possible update conflicts that must be solved with ad-hoc solutions, and of possible temporary data state inconsistencies leading to network anomalies (e.g., forwarding loops) [4]. This model applies to generic eventual-consistent data structures, such as the ones adopted in ONOS.

We now focus on the delay tradeoff achievable in the Sw-Ctr and in the Ctr-Ctr control planes. For the MDO model, the two planes are decoupled. Thus, small Sw-Ctr delays imply high reactivity of the controllers, whereas small Ctr-Ctr delays imply lower probability of network state inconsistency. For the SDO model, also the Ctr-Ctr delays affect the perceived reactivity of the controllers, as shown in Sec. 4.2. Thus, reducing

Ctr-Ctr delays is as important as reducing Sw-Ctr delays. Due to topological constraints, reducing one kind of delays implies maximizing the other one, and vice versa. The effects of such delays are particularly exacerbated in large networks, where propagation delays are not negligible. These observations motivate the exploration of possible tradeoff, as investigated in Sec. 4.

3. Related work

The work in [13] emphasizes the importance of the network state consistency, and indicates that inconsistent network states degrade the performance of network applications. Thus, [13] motivates our work, since we devise the controller placement problem to target small Ctr-Ctr delays in the MDO model, thus improving the resilience of the network to possible state inconsistencies between controllers.

Many works address the controller placement problem in SDN, but with slightly different objectives. The works [14, 15, 16, 17, 18, 19, 20, 21, 22] target fault tolerance, whereas [23, 24, 25] aim at balancing the loads on the controllers. [26] strives for network resource minimization in mobile cellular networks. [27] addresses the energy-efficient controller placement problem. [28] investigates the optimal placement in the case of switches and controllers being interconnected through Wi-Fi links. [29] performs a methodological analysis among four mainstream methods that are used to solve the controller placement problem. [30] proposes a low complexity algorithm to dynamically minimize the number of provisioned controllers while satisfying the required Sw-Ctr delay in large scale networks.

The works [31, 32, 33, 34, 35] focus on the optimal controller placement by considering only the minimization of Sw-Ctr delays (average or maximum). Differently from us, they neglect completely the interaction among controllers and thus the Ctr-Ctr delays. [36] formulates a multi-objective optimization controller placement problem with the goal of minimizing Sw-Ctr delays, maximizing controller load balancing and network reliability. In the case of SDO model, [31, 32, 33, 34, 35, 36] neglect the relevant role of the data owner. In the case of MDO, we will show in Sec. 4.2.3 that by relaxing the minimum switch-to-controller delay target, it is possible to significantly reduce the Ctr-Ctr delays and improve the convergence to a consistent network state.

Interestingly, [37] aims at achieving minimal Sw-Ctr delays, minimal Ctr-Ctr delays and controller load balancing simultaneously, using an approach based on Nash Bargaining game. [38] addresses the controller placement problem considering a wide combination of metrics: average/maximal Sw-Ctr and Ctr-Ctr delays, the level of load balancing, and the number of isolated switches in case of network partitioning. The latter metric is tailored to a resilient controller placement. Notably, neither [38] or [39] consider the combined effect of Sw-Ctr and Ctr-Ctr delays in the reactivity perceived at the switches as in our paper. From the algorithmic point of view, [38] adopts exhaustive search to find the optimal Pareto controller placements for small size networks, exactly as the Exa-Place algorithm presented in Sec. 4.1.1, and proposes a simulated annealing approach for large networks. Differently from our proposed Evo-Place (presented in Sec. 4.5.1), the algorithm in [38] requires careful tuning of many parameters and obtains the solution with a number of iterations around 1-10% of the sample space, similar to the results obtained by our Evo-Place. Finally, [40] provides a general mathematical framework to compute the optimal controller placement, under generic cost functions, but it neglects the role of Ctr-Ctr delays.

A preliminary version of this work appeared in [41], which provides some preliminary results on the tradeoff between Sw-Ctr and Ctr-Ctr delays (Sec. 4.1.2) and proposes the reactivity models for the two data-ownership models (Sec. 4.2).

4. The placement of distributed controllers

This section provides the main contributions of our work. In Sec. 4.1 we describe the controller placement problem and discuss the Pareto optimal solutions in real ISP topologies. In Sec. 4.2 we propose two simple analytical models to evaluate the reaction time based on the adopted data-ownership model. Sec. 4.3 applies the SDO model to a real reactive forwarding application in OpenDaylight and provides an accurate experimental validation of the formula for the SDO model in an operational SDWAN. In Sec. 4.4 we formalize the ILP problem to minimize the reaction time and numerically investigate its effects. To address the limited scalability of the optimal ILP solvers considered in the previous sections, we devote Sec. 4.5 to present evolutionary algorithms to optimize the controller placement, and numerically evaluate their performance in real ISP topologies.

4.1. The controller placement problem

Let N be the total number of switches in the network and C be the total number of controllers to place in the topology. The output of any placement algorithm can

be represented by the vector denoted as *placement configuration*:

$$\pi = [\pi_c]_{c=1}^{C} \tag{1}$$

where $\pi_c \in \{1, \ldots, N\}$ identifies the switch to which controller c is physically connected. We assume in-band control traffic and that all the controllers are connected to distinct switches (equivalently, two controllers cannot be connected to the same switch), i.e., $\pi_c \neq \pi_{c'}$ for any $c \neq c'$. This is because controllers are connected to distinct switches to maximize scalability and reliability. Furthermore, for the scope of our paper many controllers connected to the same switch are perfectly equivalent in terms of delay tradeoff as placing just one single controller to the same switch.

Let Ω be the set of all placement configurations; thus, the total number of possible placements is

$$|\Omega| = \binom{N}{C} \tag{2}$$

The optimal controller placement problem consists of finding $\pi \in \Omega$ such that some cost function (e.g., the maximum or average Sw-Ctr delay) is minimized. It is an NP-hard problem for a generic graph, as discussed in [31].

The network topology is described by a weighted graph with N nodes where each node represents a switch; each edge represents the physical connection between the corresponding switches and is associated with a delay value. Each controller is directly connected to a switch. We assume that the master controller of a switch is the one with the minimum Sw-Ctr delay. We also assume that all the communications are routed along the shortest path.

4.1.1. *Results on the placement of controllers in ISP networks*

To explore all the possible tradeoffs on the Sw-Ctr and Ctr-Ctr planes, we adopt an optimal algorithm (denoted Exa-Place) to exhaustively enumerate all possible controller placements and get all Pareto-optimal placements[1] and thus the corresponding Pareto-optimal frontier. For small/moderate values of network nodes N and number of controllers C, as considered in this section, the number of possible placements, evaluated in (2), is not so large and thus Exa-Place is computationally feasible. In Sec. 4.5.1 we will instead devise an approximated algorithm to find the Pareto frontier for large networks and/or large number of controllers.

[1] When considering two performance metrics x and y to minimize, a solution (x_p, y_p) is *Pareto optimal* if does not exist any other configuration (x', y') dominating it, i.e., better in terms of both metrics; thus, it cannot be that $x' \leq x_p$ and $y' \leq y_p$. The set of all Pareto-optimal solutions denotes the *Pareto-optimal frontier*.

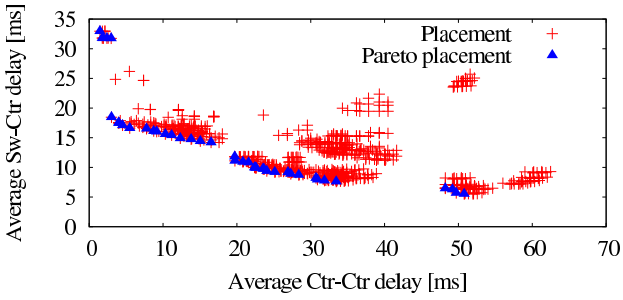


Figure 1: Delay tradeoffs in HighWinds network with 3 controllers

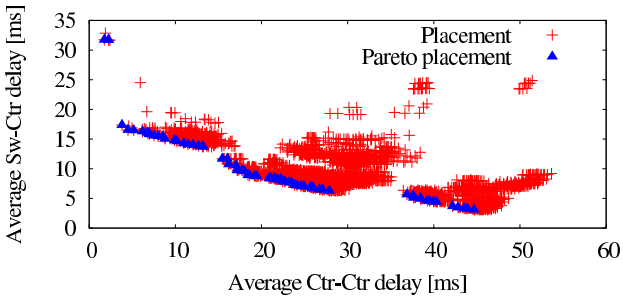


Figure 2: Delay tradeoffs in HighWinds network with 4 controllers

Coherently with previous work [31], we considered the topologies available in the *Internet topology zoo* website [42]. This repository collects more than 200 network topologies of ISPs, at POP level. For each ISP, the repository provides the network graph, with each node (i.e., switch) labeled with its geographical coordinates. From these, we compute the propagation delay between the nodes and associate it as latency of the corresponding edge. For any given controller placement, we evaluate both the Sw-Ctr delay (as the average delay between a switch and its master controller, i.e., the closest controller) and the Ctr-Ctr delay (as the average delay among controllers).

4.1.2. *Tradeoff between Sw-Ctr and Ctr-Ctr delay*

We report here the results only for HighWinds ISP, a world-wide network with 18 nodes. The preliminary version of our work [41] shows the detailed results for some other ISPs, qualitatively coherent with HighWinds.

Figs. 1-2 show the scatter plot with the Sw-Ctr and Ctr-Ctr delays achievable by all possible placements of 3 and 4 controllers, respectively. In total, all the possible $\binom{18}{3} = 816$ and $\binom{18}{4} = 3060$ different placements are

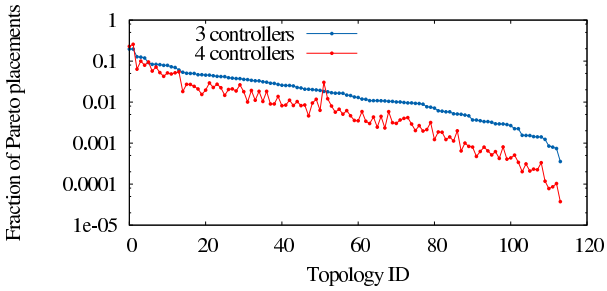


Figure 3: Fraction of Pareto placements for deploying 3 and 4 controllers in 114 ISP networks.

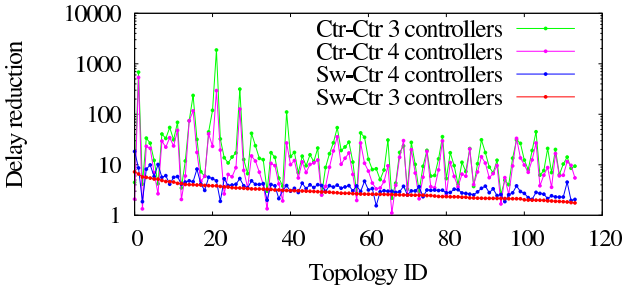


Figure 4: Sw-Ctr and Ctr-Ctr delay reductions for 3 and 4 controllers in 114 ISP networks

shown; the corresponding Pareto-optimal placements are also highlighted. When comparing the two figures, the delays for Pareto-optimal points are smaller for 4 controllers, thanks to the higher number of controllers.

The fraction of placements corresponding to Pareto points is small, equal to $38/816 = 0.46\%$ and $64/3060 = 0.21\%$ for the two scenarios. This suggests that identifying Pareto points is difficult if using random sampling. To generalize our findings, Fig. 3 shows the fraction of Pareto placements for 114 different ISP topologies. When the number of controllers increases, this fraction decreases because of the larger solution space. For some networks, corresponding to small ISPs, the fraction is quite large (around 10%). But for most ISPs the fraction of Pareto placements becomes very small (less than 1%, and, in some large networks, less than 0.1-0.01%).

When considering the specific shape of the placements in Figs. 1-2, high (or small) Sw-Ctr delays imply small (or high) Ctr-Ctr delays, respectively. The graphs show the large variety of Pareto-optimal placements. We denote by $P1$ the Pareto point with the minimum Sw-Ctr delay (i.e., the most right-low point), and by $P2$ the one with the minimum Ctr-Ctr delay (i.e., the most left-high point). To understand the relative tradeoffs along the Pareto frontier from $P1$ to $P2$, we compute the *Sw-Ctr delay reduction*, defined as the ratio between the Sw-Ctr delay in $P2$ and the one in $P1$. Similarly, we define the *Ctr-Ctr delay reduction* as the ratio between the Ctr-Ctr delay in $P1$ and the one in $P2$. Both reductions are ≥ 1 by construction. According to Fig. 1, the Sw-Ctr delay reduction is 6.0 whereas the Ctr-Ctr delay reduction is 34.8. In other words, if we allow the Sw-Ctr delay to increase by 6.0 times, then the Ctr-Ctr delay will decrease by 34.8 times, with strong beneficial effects on the time to reach consistency among the controllers.

To generalize our findings, Fig. 4 shows the Sw-Ctr and Ctr-Ctr delay reductions for 114 ISP topologies. Given the same topology, the Sw-Ctr delay reductions for 3 and 4 controllers are quite similar; interestingly, the Ctr-Ctr delay reductions are usually much higher (also 3 order of magnitude) than Sw-Ctr delay reductions. Thus, we can claim that Ctr-Ctr delays corresponding to Pareto points vary much more than Sw-Ctr delays in a generic network. Indeed, Ctr-Ctr delays are by construction between a minimum of 1-2 hops (when all the controllers are at the closest distance) and the maximum equal to the diameter of the network. The gains for the Sw-Ctr delays are lower, since the availability of multiple controllers decreases the maximum distance to reach the master controller from a switch. We can conclude that larger Sw-Ctr delays with respect to the minimum ones are much more compensated by much smaller Ctr-Ctr delays. This highlights the relevant role of the proper design of the Ctr-Ctr plane in SDN networks.

4.2. Reaction time for the data-ownership models

After a global characterization of the Pareto-optimal frontier in terms of Sw-Ctr and Ctr-Ctr delays, we now specifically evaluate the *reaction time of the controller*, defined as the latency perceived by the switch when a new network event is generated. This time depends on the specific data-ownership model adopted by the controllers and on a specific combination of Sw-Ctr and Ctr-Ctr delays.

4.2.1. Reactivity model for MDO model

In an MDO scenario, a generic event occurring at the switch (e.g., a miss in the flow table) generates a message (e.g., a `packet-in`) to its master controller, which processes the message locally and eventually sends back a control message to the switch (e.g., `flow-mod` or `packet-out` message). In the meanwhile, in an asynchronous way, the master controller advertises the update to all the other controllers. Thus, we can claim:

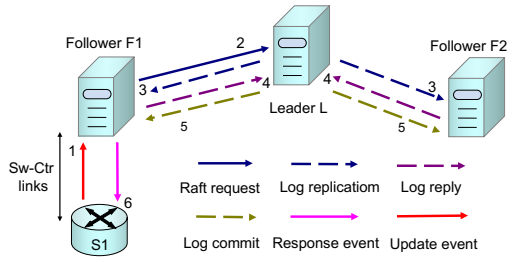


Figure 5: Control traffic due to SDO model for an update event at the switch, coherent with Raft consensus algorithm.

Property 1. *In an MDO scenario for distributed SDN controllers, the reaction time T_R^{MDO} of the controller perceived at the switch is:*

$$T_R^{MDO} = 2d_{sw\text{-}ctr} \tag{3}$$

being $d_{\text{sw-ctr}}$ the delay from the switch to its master controller, i.e., to the switch where its master controller is attached.

4.2.2. Reactivity model for SDO model

In an SDO scenario, we assume the exchange of messages coherent with the detailed description of the Raft algorithm available in [7]. According to the Raft implementation in ODL, the controller can operate as either leader or as one of the followers, for a specific data store (denoted "shard" in the following). As an example, Fig. 5 shows a general message exchange sequence in a cluster with 3 controllers (one leader and two followers), when an update event (e.g., `packet-in` message) for the shard is generated at some switch (S1 in the figure), which receives a response message from its controller due to the update (e.g., *packet-out* message). The arrows show the exchange of messages in both Sw-Ctr and Ctr-Ctr planes triggered by the update event; the number associated to each arrow shows the temporal sequence of each packet. We have now two cases.

In the first case, S1's master controller is a follower for the shard (as depicted in Fig. 5). Thus, the switch sends the update event (message 1) to the master controller, which asks the leader to update the shard through a "Raft request" (message 2). Now the leader sends a "log replication" message to all its followers (message 3) and waits for the acknowledge from the majority of them ("log reply" in messages 4). Only at this point, the update is committed through a "log commit" (message 5) sent to all the followers. Thus, after receiving the commit message, S1's master controller can process the update on the shard and generate the response event (message 6) to the switch.

In the second case, S1's master controller is the leader for the shard. This case is identical to the previous one except for the "Raft request" message 2, which is now missing.

For both cases, the controller's reaction time perceived by switch S1 is given by the time between the update event and the response event messages. Let $d_{\text{sw-ctr}}$ be the communication delay between the switch and its master controller and $d_{\text{ctr-leader}}$ the communication delay from the master controller and the leader (being zero whenever the master is also leader). Assume a cluster of C controllers. Because of the majority-based selection, let $d_{\text{ctr*-leader}}$ be the communication delay between the leader and the farthest follower belonging to the majority (i.e., corresponding to the $\lfloor (C/2) \rfloor$-th closest follower). Observing Fig. 5, the reaction time is obtained by summing twice $d_{\text{sw-ctr}}$, twice $d_{\text{ctr-leader}}$ (only in the first of the above cases) and twice $d_{\text{ctr*-leader}}$. Thus, we can claim:

Property 2. *In an SDO scenario (e.g., adopting Raft consensus algorithm) for distributed SDN controllers, the reaction time T_R^{SDO} of the controller perceived at the switch is:*

$$T_R^{SDO} = 2d_{sw\text{-}ctr} + 2d_{ctr\text{-}leader} + 2d_{ctr*\text{-}leader} \tag{4}$$

Thus, the reaction time is identical to the one for MDO model plus either 2 or 4 times the RTT between the controllers, when the master controller is either leader or follower of the shard, respectively. Notably, the delays between controllers may be dominant for large networks such as SDWAN, as also shown experimentally in Sec. 4.3. The model in (4), here obtained in a speculative way, will be tailored to a specific ODL network application and experimentally validated in an operational SDWAN running ODL, as described in detail in Sec. 4.3. Thus, we can claim that the devised model is very accurate.

4.2.3. Experimental results

We investigate the reaction times achievable for different data-ownership models, based on Properties 1 and 2. Given a controller placement, we study the effect of selecting the data owner among the controllers on the perceived controller reactivity. We show the results just for HighWinds ISP, but the results for other ISP networks are available in [41].

In Fig. 6, we report the scatter plots of the average reaction times for the SDO and the MDO models when considering all possible controller placements and all possible selections for the data owner, in the case of 3 controllers. Each controller placement appears with

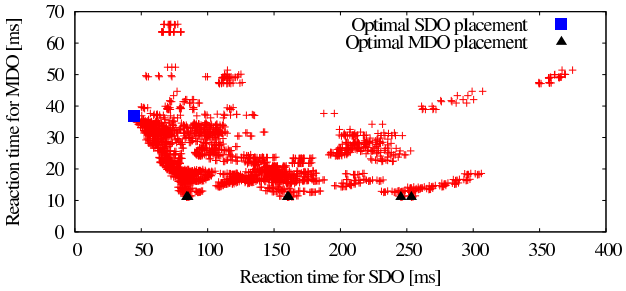


Figure 6: Average reaction times in HighWinds network for all the placements. Optimal placements for the two data-ownership models are highlighted.

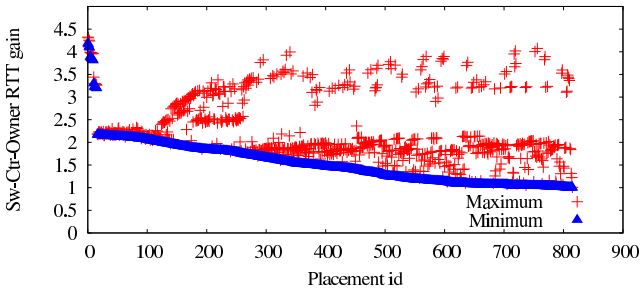


Figure 7: Reaction time reduction in HighWinds network for the optimal selection of the data owner in the SDO model.

3 points aligned horizontally, one for each data owner, since the data owner selection does not affect the MDO reaction time. In the plots we have highlighted the placements with the minimum reaction time according to the SDO and MDO models. By construction, the minimum reaction time for the MDO model is always smaller than the one for the SDO model. From these results, the optimal placements appear very different for the two data-ownership models and this fact motivates the need for a careful choice not only of the controller placement, but also of the data owner, in the SDO case.

To highlight the role of the proper selection of the data owner for the SDO model, in Fig. 7 we investigate the benefit achievable when considering the best data owner among the 3 available controllers, for the three ISPs under consideration. Assume that a given controller placement corresponds to three values of reaction times: d_1, d_2 and d_3, sorted in increasing order. The minimum *reduction factor* is defined as d_2/d_1 and the maximum reduction factor as d_3/d_1. We plot the delay reduction factor due to the optimal choice of the data owner, for any possible placement. For the sake of readability, the placements have been sorted in decreasing order of minimum reduction factor. Fig. 7 shows that a careful choice of the data owner in the SDO model decreases the reaction time by a factor around 2 and 4.

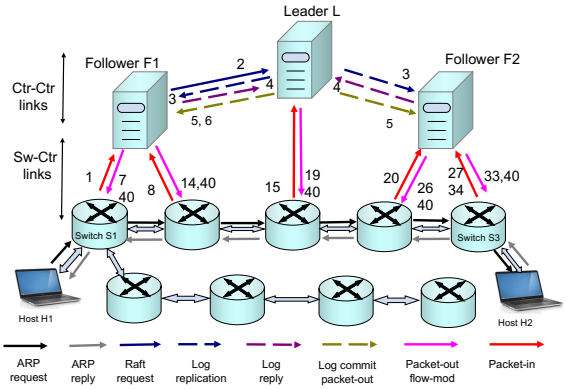


Figure 8: The control traffic for l2-switch application in ODL. For the sake of clarity, we report just some sequence numbers. The packets sent with sequence 2-6 repeat as 9-13, 21-25, 28-32, 35-39. The packets sent with sequence 3-5 repeat as 16-18, since the master controller of the third switch along the path is also the shard leader. Only the messages between the controllers and the switches along the source-destination path are shown.

These results show that the selection of the data owner in the SDO model has the largest impact on the perceived performance of the controller, and can be easily optimally solved by considering all the possible C cases, after having fixed the controller placement.

4.3. Flow setup time for reactive forwarding in ODL

To validate and show the practical relevance of the SDO model devised in Sec. 4.2.2, we apply Property 2 to compute the flow setup time for the specific layer-2 forwarding application called "l2-switch" available in ODL, deployed on a generic topology. Notably, even if formula (4) is derived in a speculative way, in the following section we will show that it is *very accurate* from experimental point of view, and thus its relevance is practical. The same methodology can be applied to analyze other applications, given the knowledge of their detailed behavior.

ODL l2-switch application provides the default reactive forwarding capabilities and mimics the learning/forwarding mechanism at layer 2 of standard Ethernet switches. Anytime a new flow enters the first switch of the network, the corresponding ARP-request is flooded to the destination, and only when the ARP-reply is generated, the controller installs a forwarding rule at the MAC layer in all the switches involved in the path from the source to the destination, and vice versa. The association of a MAC address with the switch port, needed for the learning phase, is distributed to the other controllers using the Raft algorithm.

Assume a generic topology as shown in Fig. 8 connecting source host H1 to destination host H2, with ev-

ery switch s attached to its master controller (denoted as $c(s)$), which can be either a follower or a leader (denoted as L) within the cluster. We assume initially empty flow tables in all the switches. At the beginning, the first ARP-request corresponding to a new flow from H1 is flooded in the whole network (loops are avoided by precomputing a spanning tree on the topology). Anytime the ARP packet is received at a switch, a packet-in is generated and the association [MAC source address, ingress port, switch identifier] is stored in the shared data store, in order to mimic the standard learning process. This means that at each switch, along the path from the source to the destination, a latency is experienced according to formula (4). When the ARP-request reaches the destination, H2 sends back an ARP-reply, which generates a packet-in from the last switch (denoted as s') to the controller. This event generates another update since the controller learns the port of s' at which H2 is connected. *Only at this point*, the controller installs a flow rule across all the switches in the source-destination path and then the ARP-reply is switched back to the source. Thus the flow setup time can be evaluated as *ARP reaction time* $t_{\mathrm{r,\,ARP}}$, defined as the interval between the time when the H1's switch sends the packet-in message (due to the ARP-request) to its controller and the time when H1's switch receives the packet-out/flow-mod messages (due to the ARP-reply). Note that the flow setup time depends on the considered application, which installs the flow rules across all the switches involved in the path just after the ARP reply at the destination host is generated.

Let $d_{i,j}$ be the propagation delay between nodes i and j, computed by summing all the contributions along the shortest path from i to j. Let $\mathcal{P}$ be the list of all the nodes involved in the routing path from $H1$ to $H2$, in which the last switch s' appears twice. Thus, the total number of updates within a flow is $|\mathcal{P}|$. We can claim:

Property 3. *In OpenDaylight (ODL) running the l2-switch application, the flow setup time can be computed as*

$$t_{r,ARP} = 2d_{H1,H2} + \sum_{s\in\mathcal{P}}(2d_{s,c(s)} + 2d_{c(s),L}) + 2|\mathcal{P}|d_{cnt^*\text{-}follower} + |\mathcal{P}|t_c \quad (5)$$

Indeed, the first term in (5) represents the delay to send the ARP request and reply along the routing path, the second term represents the delay occurring for all the switches along the path (the final switch s' is double counted) due to the packet-in and the packet-out/flow-mod ($2d_{s,c(s)}$) and due to the Raft-request and log commit ($2d_{c(s),L}$), the third term represents the delay to get the acknowledgement from the majority for each of the updates, and the fourth term represents the computation time for each update at the controller (assuming to be constant and equal to t_c).

4.3.1. Experimental validation in a SDWAN

We validate Property 3 on a real and operational network. Since (5) depends mainly on formula (4) obtained for the SDO model, our results validate Property 2.

Specifically, we run a cluster of OpenDaylight (Helium SR3 release) controllers running the default "Simple Forwarding" application. We run our experiments in the JOLNet, which is an experimental SDN network deployed by Telecom Italia Mobile (the major telecom operator in Italy). JOLNet is an Openflow-based SD-WAN, with 7 nodes spread across the whole Italy, covering Turin, Milan, Trento, Venice, Pisa and Catania. Each node is equipped with an OF switch and a compute node. The compute node is a server deploying virtual machines (VMs), orchestrated by OpenStack. Network virtualization is achieved though FlowVisor [43] and the logical topology among the OF switches is fully connected.

Due to the limited number of nodes in the JOLNet and the limited flexibility in terms of topology, we augment the topology with an emulated network running on Mininet [44] in one available compute node. We adopt the linear network topology of Fig. 9 with a variable number of nodes (from 3 to 36) and with one host attached at each switch. We generate ICMP traffic using the `ping` command among the different hosts. We run a single controller cluster with multiple ODL instances running in different nodes of the JOLNet, in order to distribute geographically the controllers across Italy. The controllers instances and Mininet run individually on single VMs for a flexible placement across the nodes. Thanks to the large physical extension of the network, we experiment with a large variety of scenarios, e.g., with large Sw-Ctr delays (when the VM of the master controller is located in a compute node far from the switch node) and/or large Ctr-Ctr delays (when the VMs of the controller instances are located in nodes far one from each other). By selecting the master controller of the switches, we change the data owner of the shared data structure within the cluster. We consider a cluster of 3 controllers and we measure the flow setup time $t_{\mathrm{r,ARP}}$ by comparing the timestamps of the packets obtained by using Wireshark as network sniffer at the Mininet interface towards the controllers.

As first step to validate Property 3, we evaluate the RTT among each pair of nodes in JOLNet and then we estimate the delay between any pair of nodes i and j

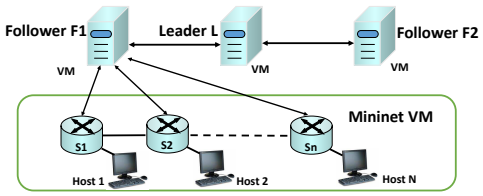


Figure 9: Network configuration for the validation of the SDO model

Table 1: Placement of the VMs for the experimented scenarios. "L": leader controller. "F1", "F2": follower controllers. "Net": Mininet.

Scenario	Turin VMs	Milan VMs	Pisa VMs
TT	Net, L, F1, F2	-	-
TMC	Net, F1	L, F2	-
TMF	Net	L, F1, F2	-
TPC	Net, F1	-	L, F2
TPF	Net	-	L, F1, F2

Table 2: Input parameters for the model, accuracy of measurements and relative error of the model

Scenario	$d_{\text{sw-ctr}}$	$d_{\text{ctr-ctr}}$	t_c	Experimental accuracy (λ)	Model error (δ)
TT	0.25 ms	0.25 ms	20 ms	1.2% - 2.7%	3.2%
TMC	0.25 ms	2.0 ms	20 ms	0.7% - 3.9%	5.2%
TMF	2.0 ms	0.25 ms	20 ms	0.6% - 3.6%	5.1%
TPC	0.25 ms	66 ms	20 ms	0.3% - 1.3%	9.2%
TPF	66 ms	0.25 ms	20 ms	0.6% - 2.3%	0.5%

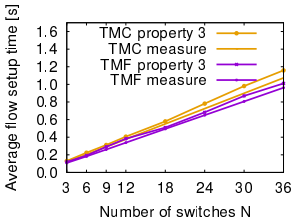


Figure 10: Experimental results with the VMs running either in Turin or Milan

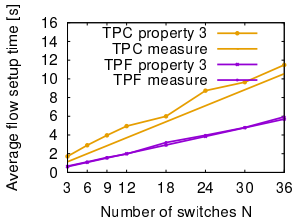


Figure 11: Experimental results with the VMs running either in Turin or Pisa

as $d_{ij} = \text{RTT}_{ij}/2$, required to apply formula (5). The experiments reported in the following refer to the scenarios using 3 JOLNet nodes, namely Turin, Milan and Pisa, to deploy the VMs. The measured RTT between Turin and Milan is 4 ms, whereas the one between Turin and Pisa is 132 ms.

As second step for the validation, we perform 100 measurements, by clearing the whole forwarding tables and restarting the controllers at each run. According to rigorous methodology, we evaluate the width I_{95} of the 95% confidence interval for the measurements and we compute the *measurement accuracy* as $\lambda = I_{95}/(2\bar{\mu})$, where $\bar{\mu}$ is the average measure. For each scenario and topology, the *relative error of the model* is instead computed as: $\delta = |M_i - T_i|/|T_i|$ where M_i is the average flow setup time according to the experiments and T_i is the flow setup time according to formula (5).

We consider different scenarios, depending on the placement of the controllers and of Mininet across the different JOLNet nodes. We refer to the physical distance between the network nodes (emulated with Mininet) and the controllers (followers and leader) as "close" when the corresponding VMs are running in the same node, and "far" when on remote nodes. Table 1 lists all the experimented scenarios, discussed in the following Sec. 4.3.2. In our cluster of 3 ODL controllers, the leader controller is denoted as "L" and the two followers are denoted as "F1" and "F2". Controller F1 is set to be master controller for all the switches in the Mininet network. "Net" represents the Mininet emulated network.

4.3.2. *Experimental results*

Table 2 summarizes the input parameters that have been used for the analytical formula in (5), and shows also the final experimental results in terms of measurement accuracy and of model error. The input parameters are obtained either by the RTT measurements when the VMs are located in different nodes, or by the steps explained below. In more detail:

- **Scenario TT (Turin-Turin):** We run the VMs of all the controllers and of Mininet in the same node, in order to evaluate the baseline latency due to the controller processing time and to the communication overhead (through the local virtual interfaces). First, we measure the communication delay between VMs, due to the local hypervisor running the different VMs, using ping command. We obtain 0.5 ms, thus we set the delay between the network and the controller, as well as between the controllers, equal to 0.25 ms. By running Mininet and measuring the flow setup time, we estimate an average processing latency of 20 ms, used as reference for all the other experiments. We run many experiments varying n_{sw} in the interval $[3, 36]$ and observe a relative error of the model equal to 3.2%, so very small.

- **Scenario TMC (Turin-Milan-Close):** Leader L and Follower F2 of the cluster are located in Milan, whereas follower F1 is co-located with the network in Turin node, thus all OF switches are close to their master controllers. The dominant term in (5) is the delay between controllers, equal to $4/2 = 2$ ms. Fig. 10 shows the average flow setup time computed

according to (5) and the one measured. According to Table 1, the experimental results are quite stable for the different values of nodes. The measured value is quite close to the theoretical one, with a relative error 5.2%. Interestingly, the flow setup time in absolute values is always larger than 100 ms and can reach also 1.2 s when the network is quite large. Notably, this is mainly due to the interaction between the master controller and the leader controller.

- **Scenario TMF (Turin-Milan-Far):** All the controllers are located in Milan, thus all OF switches are far from their master controller. Thus, now the dominant term in (5) is the delay from the switch to the controller (2 ms). Fig. 10 compares the theoretical flow setup time with the measured ones. Now the relative error of the model is 5.1%. As in the previous case, the flow setup time can be very large, due to the latency in the interaction between the network and its master controller.
- **Scenario TPC (Turin-Pisa-Close)** All OF switches and their master controller F1 are located in Turin, whereas the leader controller and the other controller are located in Pisa. The dominant term is the delay between controllers (132/2 = 66 ms). As shown in Fig. 11, the measured value approaches the theoretical value with a relative error of 9.2%. The measured delays can range from 2 to 12 s as we vary the number of switches. These huge delays are due to the interaction between leader L and follower F1, as well explained by our model.
- **Scenario TPF (Turin-Pisa-Far)** All the controllers are located in Pisa with the network still in Turin. Due to the large delay between Turin and Pisa (66 ms), the dominant term in (5) is the delay between follower F1 and the network. In all the results shown in Fig. 11, the relative error (0.5%) of the model with respect to the theoretical value is very small. Also in this case, the flow setup time is huge in absolute terms (up to 6 s), due to the large delay between the network and the controllers.

In summary, our experimental results show clearly that the reactivity of controllers, as perceived by the network nodes, is strongly affected by the inter-controller communications. Furthermore, they validate our analytical models for the SDO model, which appear to be very accurate for the Raft consensus algorithm.

4.4. Optimal placement for minimum reaction time

We present a mathematical ILP formulation of the optimal controller placement problem, in order to minimize the reaction time at the switches (or equivalently, maximizing the network reactivity), for the SDO and the MDO models. Note that in our formulation we aim at finding also the best master controller to allocate to each switch. Indeed, differently from the results shown so far, we do not assume that the master controller of a switch is the closest controller to the switch. We will show that connecting a switch to the closest controller is not always optimal, when considering also the overhead due to coordination traffic among the controllers.

4.4.1. Optimization model

We consider the network graph describing the physical interconnection among the switches. Let $\mathcal{N}$ denote the set of switches; the number of switches is $N = |\mathcal{N}|$. As a reminder, d_{mn} is defined as the delay between switch $m \in \mathcal{N}$ and $n \in \mathcal{N}$ along the shortest path. Let C denote the set of SDN controllers to be deployed; the number of controllers is $C = |C|$.

We define the following binary decision variables, for any controller $i \in C$ and any switch $n \in \mathcal{N}$:

- $X_{in} = 1$ iff controller i is placed at switch n;
- $Y_{ni} = 1$ iff controller i is the master controller of switch n.

According to standard approach, we also define some auxiliary decision variables, introduced to model the product of two binary decision variables while maintaining the problem linear.

- $\epsilon_{ijmn} = X_{im} \times X_{jn}, \forall i, j \in C, \forall m, n \in \mathcal{N}$. Thus $\epsilon_{ijmn} = 1$ only if controller i is placed at switch m and controller j is placed at switch n. To avoid non-linearities, the product can be equivalently defined as a set of linear constraints:

$$\epsilon_{ijmn} \leq X_{im}, \quad \epsilon_{ijmn} \leq X_{jn}, \quad \epsilon_{ijmn} \geq X_{im} + X_{jn} - 1$$

- $\gamma_{nim} = X_{im} \times Y_{ni}, \forall i \in C, \forall n, m \in \mathcal{N}$. Thus $\gamma_{nim} = 1$ iff controller i is placed at switch m and is the master controller of switch n. This is equivalent to:

$$\gamma_{nim} \leq Y_{ni}, \quad \gamma_{nim} \leq X_{im}, \quad \gamma_{nim} \geq Y_{ni} + X_{im} - 1$$

We define the following constraints:

- each controller is placed at only one switch:

$$\sum_{n \in \mathcal{N}} X_{in} = 1, \qquad \forall i \in C \tag{6}$$

- each switch can host at most one controller:

$$\sum_{i \in C} X_{in} \leq 1, \qquad \forall n \in \mathcal{N} \tag{7}$$

- each switch has exactly one master controller:

$$\sum_{i \in C} Y_{ni} = 1, \qquad \forall n \in \mathcal{N} \tag{8}$$

We aim at minimizing the average reaction time perceived across the switches. We have two different linear formulations for the objective function, depending of the considered data-ownership model.

According to the MDO model, (3) defines the reaction time $T_R^{MDO}(n)$ perceived at switch n as $2d_{\text{sw-ctr}}(n)$. If switch n is connected to controller i as master controller, which is placed at switch m, then $d_{\text{sw-ctr}}$ is equal to $X_{im} \times Y_{ni} \times d_{mn}$. Now the average $d_{\text{sw-ctr}}$ across all the switches can be computed as

$$\frac{1}{N} \sum_{n \in \mathcal{N}} d_{\text{sw-ctr}}(n)$$

and thus, after neglecting constant factors, the objective function for the MDO problem is:

$$\min \sum_{m \in \mathcal{N}} \sum_{i \in C} \sum_{n \in \mathcal{N}} \gamma_{nim} \times d_{mn} \tag{9}$$

For the SDO model, we aim at minimizing the average reaction time T_R^{SDO} computed as (4). Note that, as discussed in Sec. 4.2.3, the choice of the leader controller, acting as data owner, affects the reaction time strongly. In order to find the optimal choice of the leader controller we assume, without loss of generality, that controller 1 is also the leader. Observe now that the reaction time $T_R^{SDO}(n)$ perceived at switch n is composed by three terms, according to (4). The first term, $d_{\text{sw-ctr}}(n)$, is equal to the MDO case. The second term, $d_{\text{ctr-leader}}$, can be computed setting that switch n has controller i as master controller, which is placed at switch m, and that the leader controller 1 is placed at switch s, i.e., $Y_{ni} \times X_{im} \times X_{1s} \times d_{ms} = \gamma_{nim} \times \epsilon_{1ism} \times d_{ms}$; then, we average across all the switches. We compute the last term, $d_{\text{ctr*-leader}}$, by approximating the median of the delay between the leader controller and the other controllers with its average value. If the leader controller is placed at switch s and that controller i is placed at switch m, then the delay among them is $X_{im} \times X_{1s} \times d_{ms} = \epsilon_{1ism} \times d_{ms}$ and the overall average is

$$\frac{1}{C} \sum_{i \in C} \sum_{m \in \mathcal{N}} \sum_{s \in \mathcal{N}} \epsilon_{1ism} \times d_{ms}$$

By combining all the above terms, the objective function for the SDO model can be formalized as follows:

$$\begin{aligned} \min \frac{1}{N} \sum_{m \in \mathcal{N}} \sum_{i \in C} \sum_{n \in \mathcal{N}} \gamma_{nim} \times d_{mn} + \\ \frac{1}{N} \sum_{n \in \mathcal{N}} \sum_{i \in C} \sum_{m \in \mathcal{N}} \sum_{s \in \mathcal{N}} \gamma_{nim} \times \epsilon_{1ism} \times d_{ms} + \\ \frac{1}{C} \sum_{i \in C} \sum_{m \in \mathcal{N}} \sum_{s \in \mathcal{N}} \epsilon_{1ism} \times d_{ms} \end{aligned} \tag{10}$$

Note that the second term appears to be non-linear, but actually it can be converted to a linear relation as we showed at the beginning of Sec. 4.4.1 by defining a new set of auxiliary variables. We omit the details for the sake of space.

4.4.2. Numerical results

We implemented an optimal solver for the optimal placement problem in (9) and in (10) through the Gurobi solver [45].

Figs. 12-13 show the placement to minimize the reaction times, by solving (9) and (10) for both data-ownership models, when 3 controllers are deployed in the Highwinds network. As expected, the results obtained for the two models are completely different. For the MDO case, one controller is placed in each continent, in order to minimize the average delay between the switches and their master controllers. For the SDO case, instead, all the controllers are placed close in the continent with the higher number of switches, and specifically the leader is chosen close to the continent (Europe) with the second largest number of switches. This is intuitively the best tradeoff between Sw-Ctr delays and Ctr-Ctr delays.

Fig. 14 shows the minimum average reaction time for both SDO and MDO considering 3 and 4 controllers, for the smallest 89 ISP networks available in [46]. We could not run our solver for the largest topologies, because of the limited scalability of the ILP approach. In all the topologies, the optimal reactivity is mainly affected by the adopted data-ownership model, and not by the number of controllers. The MDO model achieves reaction times that are 1-2 orders of magnitude smaller than the SDO model, showing the non-negligible impact of the Ctr-Ctr traffic on the perceived performance at the switches for the SDO model.

Considering the actual master controller chosen for each switch, based on our experiments, in the MDO model all the switches are connected to the closest controller, as expected. But for SDO model, each switch is connected to the first controller along the its shortest paths to the leader controller. Thus, for the SDO model

Figure 12: Placement for MDO in HighWinds.

Figure 13: Placement for SDO in HighWinds.

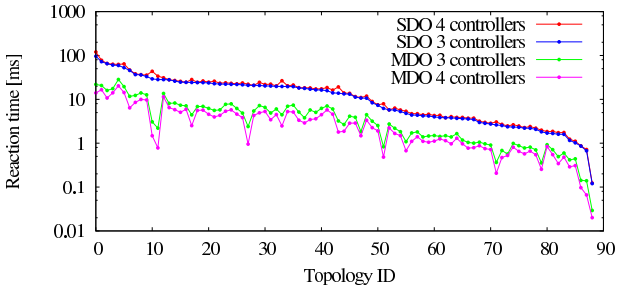


Figure 14: Optimal control plane reactivity obtained with the optimization framework when deploying 3 and 4 controllers, in 89 ISP networks.

the closest controller may not be the best solution. Indeed, it may be convenient for a switch to connect directly to the leader controller, even if farther than the closest controller, in order to reduce the impact of the Ctr-Ctr traffic among the controllers. In the considered 89 topologies, on average 25% and 35% of the switches were not connected to their closest controller for the 3 controllers and 4 controllers scenarios, respectively.

4.5. Evolutionary placement algorithms for large networks

When commenting on the numerical results in Secs. 4.1 and 4.4, we noticed that for the largest ISP topologies (in terms of number of switches and links) we could not run the exhaustive search (i.e., Exa-Place defined in Sec. 4.1.1) and the ILP solver (Sec. 4.4.2) to get the optimal placements, due to the limited scalability of the considered optimal approaches.

In the following, we present two evolutionary algorithms suitable for large networks. The first one, denoted as Evo-Place and described in Sec. 4.5.1, finds a set of Pareto controller placements. The second one, denoted as Best-Reactivity and described in Sec. 4.5.2, finds the final placement that minimizes the average reaction time perceived at the switches.

4.5.1. An evolutionary algorithm for Pareto-optimal placements

The basic idea of our algorithm is to discover new non-dominated solutions by perturbing the current set of Pareto solutions for the controller placement. Specifically, starting from a given controller placement in the network, we may get a new controller placement with better Ctr-Ctr delay by putting the controllers closer, and a new controller placement with better Sw-Ctr delay by distributing the controllers more evenly in the network. By continuously performing such perturbation, we achieve a good approximation of the Pareto frontier for the placement problem.

As term of comparison for our algorithm, we define a basic randomized algorithm, denoted as Rnd-Place and reported in Algorithm 1, to find a set of Pareto solutions just using a random sampling. The input parameters are the number of controllers C, the number of nodes N and the number of iterations $i_{\max}$. We assume that function Random-Permutation(N, C) (called in step 4) provides the first C elements of a random permutation of size N, with $C \in [1, N]$; its complexity is $O(C)$ thanks to the classical Knuth shuffle algorithm. Let P be the current set of all Pareto (i.e., non-dominated) solutions. At each iteration, a new placement is generated (step 4). Now function Add-Prune eventually adds π to P. More precisely, if π is dominated by any Pareto solution in P, then it not added to P since it is not Pareto (step 10). Instead, if any current solution $p \in P$ is dominated by π, then it is removed (step 12) and then π is added as a new Pareto solution (step 13). Add-Prune returns true if π was added successfully, otherwise it returns false.

The set P returned by Rnd-Place at the end of $i_{\max}$ iterations collects all the Pareto placements found by the procedure, and corresponds to an approximation of the optimal Pareto frontier for the controller placement problem. The randomized approach is simple but quite inefficient in terms of complexity, since it takes around $|\Omega| \log |\Omega| + 0.58|\Omega|$ iterations (thanks to the well known results about the coupon collector problem), where $|\Omega|$ is the total number of placements (see (2)), to approach the exhaustive search and find the optimal Pareto placements.

We modify Rnd-Place to exploit an evolutionary approach to boost the efficiency of the random sampling. Algorithm 2 reports the pseudocode of our proposed Evo-Place. At each iteration, the algorithm selects a random placement π (step 4) and tries to add to P, as in Rnd-Place. If the addition is successful (i.e., π is Pareto), then π is perturbed (step 7) and the new placement π' is eventually added to P (step 8). The loop for

Algorithm 1 Random algorithm for finding Pareto controller placements

```
1:  procedure RND-PLACE(C, N, i_max)
2:      P = ∅                          ▷ Init the set of Pareto solutions
3:      for i = 1 → i_max do           ▷ For i_max iterations
4:          π =RANDOM-PERMUTATION(N, C)
5:          ADD-PRUNE(P, π)
6:      return P
7:  procedure ADD-PRUNE(P, π)
8:      for all p ∈ P do
9:          if π is dominated by p then
10:             return false           ▷ No addition of π
11:         if p is dominated by π then
12:             P = P \ {p}            ▷ Remove p
13:     P = P ∪ {π}                    ▷ Add π since not dominated
14:     return true                    ▷ Successful addition of π
```

Algorithm 2 Evolutionary algorithm for finding Pareto controller placements

```
1:  procedure EVO-PLACE(C, N, i_max)
2:      P = ∅                          ▷ Init the set of Pareto solutions
3:      for i = 1 → i_max do           ▷ For i_max iterations
4:          π =RANDOM-PERMUTATION(N, C)
5:          success_flag=ADD-PRUNE(P, π)
6:          while (success_flag=true) do
7:              π' =DECREASE-CTR-CTR-DELAY(π)
8:              success_flag=ADD-PRUNE(P, π')
9:              π = π'
10:     return P
```

Algorithm 3 Perturb a given controller placement π to decrease Ctr-Ctr delay

```
1:  procedure DECREASE-CTR-CTR-DELAY(π)
2:      for c = 1 → C do
3:          h_c = Σ_{k≠c} d_{π_c π_k}      ▷ Total delay from c
4:      c' = arg max_c{h_c}                ▷ Farthest controller
5:      c'' = arg min_{c≠c'}{d_{π_c π_c'}} ▷ c''s closest cnt.
6:      n=find first node in the shortest path from c' to c''
7:      if n ≠ π_c'' then
8:          π_c' = n                       ▷ Move c' into n
9:      return π
```

the perturbation ends when the newly perturbed solution cannot be added to P, since it is dominated by other solutions (steps 6-9). The perturbation phase is described by DECREASE-CTR-CTR-DELAY, whose pseudocode is reported in Algorithm 3.

DECREASE-CTR-CTR-DELAY perturbs the given placement solution π by decreasing the Ctr-Ctr delay. Its main idea is to move the farthest controller closer to the others. Indeed, in steps 2-3 the average distance is computed for each controller to all the others (actually, we omit the division by $C-1$ since it is useless for the following steps). We define d_{ij} as the minimum delay from node i to j, based on the propagation delays in the network topology. Now we choose c' as the controller with the maximum average delay towards the others (step 4) and find c'' as the closest controller to c' (step 5). Now we move c' one hop towards c'' (steps 6) along the shortest path from c' to c''; note that the check that c'' is at least 2 hops away from c' guarantees that the movement is possible. As a result, DECREASE-CTR-CTR-DELAY decreases the average Ctr-Ctr distance most of the time.

4.5.2. An evolutionary algorithm to minimize the reaction time

We adopt the same approach of EVO-PLACE to find the best placement that minimizes the reaction time according to the MDO and SDO models, computed according to Property 1 and 2 respectively. The pseudocode in Algorithm 4 describes the proposed evolutionary approach. At each iteration, a random placement (step 4) is generated in order to possibly escape from local minima. Another candidate placement is obtained (step 6) by perturbing the optimal candidate solution π_{BEST} found adopting DECREASE-CTR-CTR-DELAY. The reaction time for each new candidate solution is evaluated in step 7 by calling UPDATE-BEST. This function (step 9) exploits the analytical formulas to compute the reaction time to check whether the given placement has a smaller reaction time than the candidate optimal placement obtained so far. As an example, the formula adopted in Property 3 of Sec. 4.3 can be adopted to compute the reactivity for the standard reactive layer-2 forwarding application in the ODL controller.

4.5.3. Performance of EVO-PLACE

We compare the performance of EXA-PLACE, RND-PLACE and EVO-PLACE on different networks with varying number of controllers. For a fair comparison, we run EVO-PLACE after having fixed $i_{\max}$ and record the actual total number of analyzed placements. Then we use this value to set the number of placements considered by RND-PLACE. Thus, all the results comparing EVO-PLACE with RND-PLACE are obtained with the same number of analyzed placements.

In Fig. 15, we show the results for the Garr network, a nation-wide Italian ISP, taken from [46], with

Algorithm 4 Evolutionary algorithm to find Placement with minimum reaction time

```
1: procedure BEST-REACTIVITY(C, N, i_max)
2:     π_best = {}                          ▷ Init the best solution
3:     for i = 1 → i_max do                 ▷ For i_max iterations
4:         π =RANDOM-PERMUTATION(N, C)
5:         UPDATE-BEST(π, π_best)
6:         π =DECREASE-CTR-CTR-DELAY(π_best)
7:         UPDATE-BEST(π, π_best)
8:     return π_best
9: procedure UPDATE-BEST(π, π_best)
10:    if REACTION-TIME(π)<REACTION-TIME(π_best) then
11:        π_best = π                       ▷ Found better placement
```

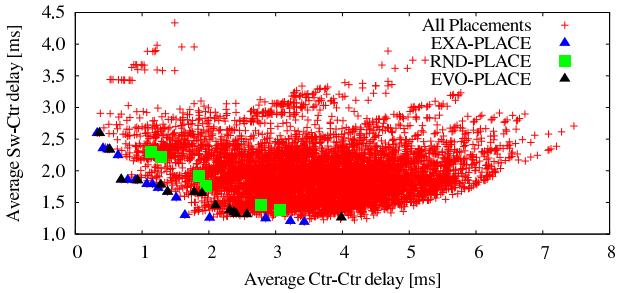


Figure 15: Optimal Pareto frontier (EXA-PLACE) and its approximations (RND-PLACE, EVO-PLACE with $i_{max} = 50$) for the placement of 3 controllers in Garr network

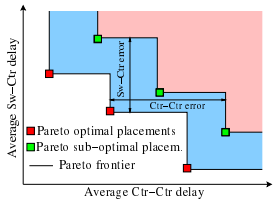


Figure 16: Definition of Ctr-Ctr error and of Sw-Ctr error with respect to the optimal Pareto frontier

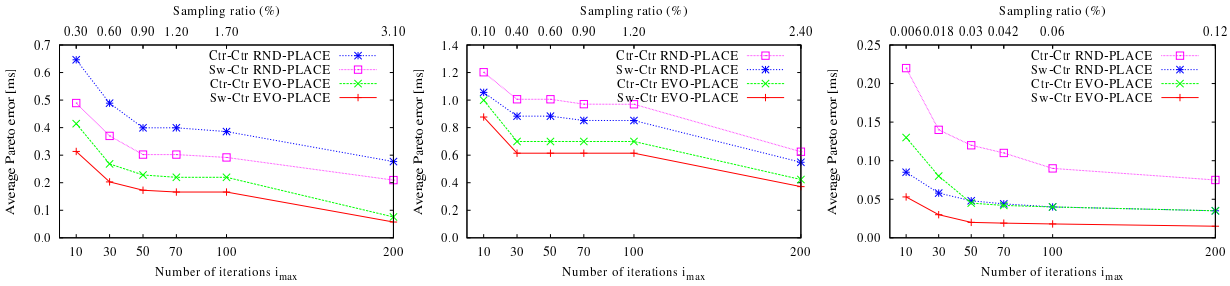
35 nodes, for the case of 3 controllers. Thus, $N = 35$, $C = 3$ and thus $|\Omega| = \binom{35}{3} = 6,545$ are all the possible placements evaluated by EXA-PLACE. The corresponding Pareto points represent the optimal Pareto frontier, used as a reference for the frontiers obtained with the other algorithms. The graphs show the sub-optimal Pareto points obtained by RND-PLACE and EVO-PLACE running for $i_{max} = 50$ iterations, corresponding to a sampling fraction equal to 0.9% of all possible solutions. From the figure, the Pareto placements computed by EVO-PLACE appear to approximate much better the optimal ones than RND-PLACE, given the same number of iterations.

In order to evaluate in a quantitative way the "distance" between the optimal Pareto frontier computed by EXA-PLACE and the approximated ones obtained by RND-PLACE and EVO-PLACE, we define two error indexes, as depicted in Fig. 16, derived from classical volume based performance indexes for Pareto sets [47]: (i) the *average Sw-Ctr error*, computed as the average vertical distance between the optimal Pareto frontier and the approximated Pareto frontier, (ii) the *average Ctr-Ctr error*, computed as the average horizontal distance between the two frontiers.

Fig. 17a shows the behavior of the two errors as a function of the number of iterations, in the same scenario considered in Fig. 15. Each experiment, for a given number of iterations, is repeated multiple times to get an accurate estimation of the error. When increasing the number of iterations, the Pareto errors decrease, thanks to the larger space of considered solutions. As already observed, we expect that the Ctr-Ctr delays are larger than the Sw-Ctr delays in absolute terms, and thus the corresponding errors are following the same behavior. After 200 iterations both the Sw-Ctr error and the Ctr-Ctr error are around 3 times smaller than the errors obtained for 10 iterations, and in absolute terms very small (less than 0.1 ms) for EVO-PLACE. The advantage of EVO-PLACE with respect to RND-PLACE tends to increase with the number of iterations, indeed the errors for EVO-PLACE are between 1.5× (for low number of iterations) and 3× (for high number of iterations) smaller than RND-PLACE. In general, an accurate estimation of the Pareto frontier can be achieved with a sampling ratio equal to 1-3% of the total solution space.

We extend our investigation to other larger topologies, for which EVO-PLACE is much faster than EXA-PLACE. Figs. 17b and 17c show the errors in the Pareto frontiers obtained for China-Telecom and ITC-Deltacom networks, respectively, taken from [46]. In China-Telecom (38 nodes), the Pareto errors decreases by a factor 2.5 from 10 to 200 iterations, and the relative gain of EVO-PLACE with respect to RND-PLACE is around 1.2-1.5, decreasing for larger sampling space. Also in this case, around 1-2% of the sampling space is enough to obtain an accurate estimation of the Pareto region. Fig. 17c shows the errors for ITC-Deltacom network, which is a large USA ISP with 100 nodes. Despite the large size of the network, after 50 iterations (corresponding to 0.03% of sampling ratio) all the errors tend to stabilize for EVO-PLACE, showing a relative gain with respect to RND-PLACE that is always more than 2.

The results obtained for the above 3 ISPs show the effectiveness of the evolutionary approach with respect to the simple random sampling. Notably, also in absolute terms all the errors are small, even if their actual



(a) Garr network (b) Chinanet network (c) Deltacom network

Figure 17: Pareto frontier error with 3 controllers.

values depend on the geographical area covered by each network.

We have also evaluated the scenario with Colt-Telecom from [46], an Europe-wide ISP covering 149 nodes, in the case of 10 controllers. In this scenario Exa-Place cannot run since the total number of possible placements is larger than 10^{15} and thus we cannot evaluate the average errors with respect to the optimal Pareto points. We instead observe that Evo-Place is always outperforming Rnd-Place by reducing the average Sw-Ctr and Ctr-Ctr delays of 0.25 – 1 ms.

In conclusion, for all the scenarios we investigated, we have observed a better Pareto frontier obtained by Evo-Place with respect to Rnd-Place, given the same number of considered placements and thus the same computation complexity. Thus, the evolutionary approach adopted in Evo-Place appears efficient in finding the Pareto placements for a given network topology, especially when the network is large and an exhaustive approach is not anymore feasible.

4.5.4. *Performance of* Best-Reactivity

We evaluate the performance of Best-Reactivity algorithm applied to the MDO and SDO models to find the single optimal placement that minimizes the average reaction time. For the SDO model, we evaluate the performance on 58 middle-size ISP networks, again taken from [46], for 3 controllers. We compare Best-Reactivity with a random sampling of the controller placement. For the sake of space, we do not report the detailed results. The ratio between the reaction time obtained with Best-Reactivity is on average 2.1× (at most 4.4×) smaller than the random sampling, given the same number of considered placements. Our results prove the effectiveness of adopting Decrease-Ctr-Ctr-Delay to decrease the reaction time of the candidate for the optimal placement.

We also compare Best-Reactivity with the solution

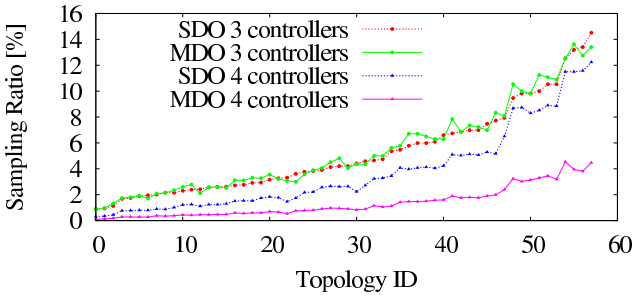


Figure 18: Complexity for Best-Reactivity to achieve a reactivity ≤ 30% larger than the minimum reaction time.

obtained through Gurobi [45] implementing the optimization model described in Sec. 4.4.1, under the same 58 ISP topologies considered above. According to standard methodology, we evaluated the approximation ratio with respect to the optimal algorithm, i.e., the ratio between the average reaction time obtained by Best-Reactivity and the one obtained by Gurobi solver. Fig. 18 shows the number of iterations, measured as sampling ratio with respect to the exhaustive search, to achieve 1.3 approximation ratio, i.e., to obtain a reaction time which is ≤ 30% larger than the minimum one obtained by Gurobi. We investigate both SDO and MDO scenarios, for 3 and 4 controllers. Our results show that the actual solution space to consider depends on the actual topology, and a reasonable good solution (i.e., ≤ 30% error with respect to the optimal one) can be obtained by sampling around 1-10% of the solution space. Interestingly, increasing the number of controllers improves the efficiency of Best-Reactivity thus making us more confident about the robustness of the proposed approach.

5. Conclusions

We consider a distributed architecture of SDN controllers, with an in-band control plane. We investigate the placement of the controllers across the network nodes. Differently from previous work, we focus on the coordination traffic exchanged among controllers, needed to synchronize their shared data structures. We distinguish two possible models for the shared data structures: the single (SDO) and the multiple (MDO) data-ownership models, both currently implemented in state-of-the-art controllers for strong consistent and eventual consistent data structures.

First, we study the optimal controller placement problem by considering all the communications occurring in the control plane, with emphasis on the communications between the controllers. We investigate the Pareto frontier that characterizes all the possible tradeoffs between Sw-Ctr delays and Ctr-Ctr delays. Throughout the analysis of realistic ISP topologies, we show that some limited increase in the Sw-Ctr delays with respect to the minimum ones permits to experience much lower Ctr-Ctr delays (up to some order of magnitude). Thus, a placement simply based on minimizing the Sw-Ctr delays may not be optimal. To compute the Pareto frontier for small topologies, we adopt an optimal algorithm (Exa-Place), whose scalability is limited by the adopted exhaustive search. To overcome such limitation, we devise an evolutionary algorithm (Evo-Place) able to approximate the optimal Pareto frontier with a limited error.

Second, we focus on the reaction time as perceived by the switches, which depends on the network application running on the controller. We devise analytical formulas for the two data-ownership models. In particular, the formula for the SDO model is obtained in a speculative way, and we validate its accuracy in a operational SDWAN. Thanks to these formulas, we define an ILP problem to minimize the average reaction time, under the SDO and MDO models, and we devise an approximated algorithm (named Best-Reactivity), able to scale to large networks, whose performance is shown to approximate the optimal ILP solver well.

Specifically regarding the SDO model, we provide two results, which are not expected. First, we show that an optimized choice of the controller acting as data owner can improve the reaction time by 2-4 times. Second, we show that choosing the master controller of a switch as the closest controller does not always minimize the reaction time perceived at the switches. This suggests that also the selection of the master controller of a switch must be coupled with the optimal placement of the controllers.

We believe that our work opens a new perspective for network designers, who can become aware of the importance of knowing the internal mechanisms by which the data structures are shared across the SDN controllers, and of the crucial role of the inter-controller communications. Furthermore, our investigation provides a solid methodology not only to place the controllers but also to design the network supporting the control plane in large networks, as in the scenario of large SDN networks or SDWANs.

References

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103 (1) (2015) 14–76.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.

[3] E. Brewer, Pushing the CAP: Strategies for consistency and availability, Computer 45 (2) (2012) 23–29.

[4] A. Panda, C. Scott, A. Ghodsi, T. Koponen, S. Shenker, CAP for networks, in: HotSDN, New York, NY, USA, 2013.

[5] J. Medved, R. Varga, A. Tkacik, K. Gray, Opendaylight: Towards a model-driven SDN controller architecture, in: WoWMoM, 2014, pp. 1–6.

[6] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, ONOS: Towards an open, distributed SDN OS, in: ACM HotSDN, New York, NY, USA, 2014.

[7] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proc. USENIX Annual Technical Conference, Philadelphia, PA, 2014, pp. 305–320.

[8] OpenDaylight controller clustering.
URL https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Architecture:Clustering

[9] OpenDaylight Raft consensus code review.
URL https://github.com/opendaylight/controller/tree/master/opendaylight/md-sal/sal-akka-raft/src/main/java/org/opendaylight/controller/cluster/raft

[10] J. Prajakta, ONOS Summit: ONOS Roadmap 2015 (Dec 2014).

[11] Distributed primitives.
URL https://wiki.onosproject.org/display/ONOS/Distributed+Primitives

[12] A. Muqaddas, A. Bianco, P. Giaccone, G. Maier, Inter-controller traffic in ONOS clusters for SDN networks, in: IEEE ICC, Kuala Lumpur, Malaysia, 2016.

[13] D. Levin, A. Wundsam, B. Heller, N. Handigol, A. Feldmann, Logically centralized?: State distribution trade-offs in software defined networks, in: ACM HotSDN, New York, NY, USA, 2012.

[14] F. J. Ros, P. M. Ruiz, Five nines of southbound reliability in software defined networks, in: ACM HotSDN, New York, NY, USA, 2014.

[15] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, On reliability-optimized controller placement for software-defined networks, China Communications 11 (2) (2014) 38–54.

[16] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, M. P. Barcellos, Survivor: an enhanced controller placement strategy for improving SDN survivability, in: GLOBECOM, IEEE, 2014, pp. 1909–1915.

[17] H. Li, R. E. De Grande, A. Boukerche, An efficient CPP solution for resilience-oriented SDN controller deployment, in: IPDPSW, IEEE, 2017, pp. 540–549.

[18] B. P. R. Killi, S. V. Rao, Capacitated next controller placement in software defined networks, IEEE Transactions on Network and Service Management (2017) 1–1.

[19] P. Vizarreta, C. M. Machuca, W. Kellerer, Controller placement strategies for a resilient SDN control plane, in: RNDM, IEEE, 2016, pp. 253–259.

[20] Q. Zhong, Y. Wang, W. Li, X. Qiu, A min-cover based controller placement approach to build reliable control network in SDN, in: NOMS, IEEE/IFIP, 2016, pp. 481–487.

[21] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, A. Kamisinski, Impact of SDN controllers deployment on network availability, CoRR.
URL http://arxiv.org/abs/1703.05595

[22] M. Tanha, D. Sajjadi, J. Pan, Enduring node failures through resilient controller placement for software defined networks, in: GLOBECOM, IEEE, 2016, pp. 1–7.

[23] G. Yao, J. Bi, Y. Li, L. Guo, On the capacitated controller placement problem in Software Defined Networks, IEEE Communications Letters 18 (8) (2014) 1339–1342.

[24] H. K. Rath, V. Revoori, S. Nadaf, A. Simha, Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game, in: WoWMoM, IEEE, 2014, pp. 1–6.

[25] Y. Hu, T. Luo, W. Wang, C. Deng, On the load balanced controller placement problem in software defined networks, in: ICCC, IEEE, 2016, pp. 2430–2434.

[26] S. Auroux, H. Karl, Flow processing-aware controller placement in wireless DenseNets, in: PIMRC, IEEE, 2014, pp. 1294–1299.

[27] Y. Hu, T. Luo, N. C. Beaulieu, C. Deng, The energy-aware controller placement problem in software defined networks, IEEE Communications Letters 21 (4) (2017) 741–744.

[28] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, M. El-Nainay, On stochastic controller placement in software-defined wireless networks, in: WCNC, IEEE, 2017, pp. 1–6.

[29] J. Hollinghurst, A. Ganesh, T. Baugé, Controller placement methods analysis, in: ICICM, IEEE, 2016, pp. 239–244.

[30] M. T. I. ul Huque, W. Si, G. Jourjon, V. Gramoli, Large-scale dynamic controller placement, IEEE Transactions on Network and Service Management 14 (1) (2017) 63–76.

[31] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: ACM HotSDN, 2012, pp. 7–12.

[32] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, R. Boutaba, Dynamic controller provisioning in Software Defined Networks, in: CNSM, Zurich, Switzerland, 2013, pp. 18–25.

[33] P. Xiao, W. Qu, H. Qi, Z. Li, Y. Xu, The SDN controller placement problem for WAN, in: ICCC, IEEE, 2014, pp. 220–224.

[34] G. Wang, Y. Zhao, J. Huang, Q. Duan, J. Li, A k-means-based network partition algorithm for controller placement in software defined network, in: ICC, IEEE, 2016, pp. 1–6.

[35] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, T. Li, Density cluster based approach for controller placement problem in large-scale software defined networkings, Computer Networks 112 (2017) 24–35.

[36] B. Zhang, X. Wang, L. Ma, M. Huang, Optimal controller placement problem in Internet-oriented software defined network, in: CyberC, IEEE, 2016, pp. 481–488.

[37] A. Ksentini, M. Bagaa, T. Taleb, I. Balasingham, On using bargaining game for optimal placement of SDN controllers, in: ICC, IEEE, 2016, pp. 1–6.

[38] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, M. Hoffmann, Heuristic approaches to the controller placement problem in large scale SDN networks, IEEE Transactions on Network and Service Management 12 (1) (2015) 4–17.

[39] A. Ksentini, M. Bagaa, T. Taleb, On using SDN in 5G: the controller placement problem, in: GLOBECOM, IEEE, 2016, pp. 1–6.

[40] A. Sallahi, M. St-Hilaire, Optimal model for the controller placement problem in software defined networks, IEEE Communications Letters 19 (1) (2015) 30–33.

[41] A. Bianco, P. Giaccone, T. Zhang, The role of intercontroller traffic in SDN controllers placement, in: IEEE NFV-SDN, Palo Alto, CA, 2016.

[42] S. Knight, H. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet topology zoo, IEEE JSAC 29 (9) (2011) 1765 –1775.

[43] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, Flowvisor: A network virtualization layer, Tech. rep., OpenFlow Switch Consortium (2009).

[44] Mininet: An instant virtual network on your laptop (or other PC).
URL http://mininet.org

[45] Gurobi optimizer reference manual.
URL http://www.gurobi.com

[46] The Internet Topology Zoo.
URL http://www.topology-zoo.org/

[47] T. Okabe, Y. Jin, B. Sendhoff, A critical survey of performance indices for multi-objective optimisation, in: Congress on Evolutionary Computation (CEC), Vol. 2, 2003, pp. 878–885.