Computer Communications 159 (2020) 198–205

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A comprehensive approach for optimizing controller placement in Software-Defined Networks

G. Schütz [a,*], J.A. Martins [b]

[a] Center for Electronic, Optoelectronic and Telecommunications (CEOT) and Institute of Engineering (ISE) of the University of Algarve, Faro, Portugal
[b] Center for Electronic, Optoelectronic and Telecommunications (CEOT), Campus de Gambelas, University of Algarve, Faro, Portugal

ARTICLE INFO

Keywords:
Software-Defined Networks
Controller placement
Mathematical formalization
Heuristic

ABSTRACT

Software-Defined Networks (SDNs) are characterized by dividing a network architecture in a data plane (i.e., any packet-relaying nodes like switches or routers) and a control plane, where specialized controllers assign forwarding decisions to the underlying data plane, and must do so in a very short timeframe. Thus, controllers play a key role in SDNs and the Controller Placement Problem (CPP) becomes a critical issue, affecting network delays and synchronization. If there are significant propagation delays between controllers and nodes, or among controllers, their ability to quickly react to network events is affected, degrading reliability. In this work, we propose a comprehensive mathematical formalization of the CPP, which constrains propagation latency and controller capacity, and determines simultaneously the minimum number of controllers, their location and the assignment of nodes to each, while keeping a balanced load distribution among controllers. As CPP is NP-hard, a heuristic approach is also presented. Simulations for 60 network scenarios show that this approach obtains balanced and resilient solutions, in negligible time, which are proven to be optimal or near optimal for 90% of the evaluated cases.

1. Introduction

1.1. Motivation

The key architectural characteristics of Software-Defined Networks (SDNs) are the separation of the control and data planes, as well as defining a logically centralized control plane. This new network topology offers significant advantages for cloud datacenters and for service providers in Wide Area Networks (WANs), as it simplifies data forwarding and allows managing a network in a flexible way. However, it also introduces additional challenges, affecting SDN performance, which have to be addressed as a whole. To obtain a balanced and failure-resilient topology, it is necessary to determine a minimum number of controllers and where should they be placed [1], while ensuring low latencies between nodes and associated controllers, as well as between controllers. Simultaneously, it is necessary to keep balanced the processing loads of controllers, and within their operating capacity [2–4]. Thus, any proposed solution for the Controller Placement Problem (CPP) of SDNs needs to address how many controllers are required and where to place them [1], taking into consideration the following issues: (a) capacities of controllers, (b) propagation latencies between controllers and nodes, and inter-controllers, and (c) load distribution among controllers. Good solutions will increase the ability of the network to react to events quickly, leading to a more robust topology, even when failures occur. The above reasoning grounds our approach. In this work, we present a mathematical formalization of the CPP, which minimizes the number of needed controllers, while simultaneously addressing the constraints related to issues (a), (b) and (c).

1.2. Related work

Previous research has addressed many issues related to the CPP (see [5] and [6] for a comprehensive survey). This section briefly overviews some of the works related to the main issues addressed by our approach: the optimum number of controllers, latency, resilience and load balancing.

The CPP was first introduced by [1]. The authors studied the placement of controllers in SDNs, with the objective of minimizing the propagation latency from nodes to their assigned controller. Their approach includes the CPP in a large class of localization problems, corresponding to a K-median problem or a K-centre problem, when minimizing the average propagation latency or the worst-case propagation latency, respectively. Their proposal is exhaustive with all potential locations being evaluated. Han et al. [7] proposed a minimum-control-latency optimized algorithm aiming to maximize the number of controlled switches within a latency bound, while Wang et al. [8] proposed an

* Corresponding author.
E-mail addresses: gschutz@ualg.pt (G. Schütz), jamartins@ualg.pt (J.A. Martins).

https://doi.org/10.1016/j.comcom.2020.05.008
Received 12 December 2019; Received in revised form 6 February 2020; Accepted 4 May 2020
Available online 6 May 2020

optimized K-means algorithm to reduce the maximum latency. These works only address the propagation-latency issue (b) above.

Concerning controller load balancing, G. Yao et al. [2] apply a K-centre strategy while taking into consideration controller capacity; Advait Dixit et al. [9] propose a dynamic assignment of nodes to controllers, balancing the load of controllers in real-time, while Jiménez et al. [3] introduce a K-critical algorithm to construct balanced trees, distributing the load among controllers while satisfying delay limits. Hu et al. [10] study the trade-off between delay and load balance. Killi et al. [11] proposed a controller placement methodology based on cooperative game theory to improve a K-means algorithm. Likewise, our work considers load balancing and delay but it also determines the minimum number of controllers and their placement, assuring the requested latency constraints (between controllers and nodes, and also inter-controllers) and complying with controller capacities.

Other approaches, like Hock et al. [12] solve the problem by proposing specific metrics and analyse different resilience issues, while Hu et al. [13] and Muller et al. [14] maximize the number of paths that can be used for routing nodes to controllers, and inter-controllers, aiming to increase reliability. Schütz [15] proposes a K-cover model which ensures disjoint primary and backup paths between switches and controllers, providing reliable networks against link, switch and controller failures. In this paper, we propose a more generalized approach that is able to build a robust topology, while satisfying several constraints.

The CPP has been formalized as a multi-objective optimization problem [12,16,17], where genetic algorithms and adaptive heuristics are applied to estimate the Pareto frontier. Also [18] extended POCO framework [12] adding a heuristic approach, which provides faster computation times but is less accurate. Usually, multi-objective problems are solved by applying genetic algorithms and meta-heuristics. These methods require excessive execution times and a lot of hyper-parameter tuning. Instead, we opt to consider the latency and balancing requirements as constraints and only have one objective, formalizing the CPP as a binary linear programming problem. We also propose an effective and very fast heuristic, necessary for large networks, due to the CPP hardness.

1.3. Contributions

Although the above literature addresses one or more issues, none of them intends to globally minimize the number of controllers necessary to provide a load-balanced topology, while constraining and minimizing latencies between nodes and controllers and inter-controllers. Thus, we believe our approach is significantly different from the previous ones, as it seeks to find a comprehensive solution that simultaneously deals with all of these issues. Thus, the main contributions of this paper are:

1. A mathematical formalization of the CPP, which minimizes the number of required controllers, chooses their location, and assigns nodes to controllers—while satisfying constraints related to capacity, propagation latency and load balancing. Most of other works, as those cited in Section 1.2 do not minimize the number of controllers and are focused on a single issue while evaluating metrics and ratios for different numbers of controllers.
2. The definition of a lower bound on the number of controllers, easily computed based on our formalization, which allows for an evaluation on solution quality.
3. The proposal of a heuristic algorithm to solve our formalization, and the evaluation of its performance on real networks. As the CPP is NP-hard, our formalization will be time consuming when applied to large topologies, so a heuristic approach is required. Our simulations show that it is capable of achieving balanced topologies, complying with requested latency constraints (both from nodes to controllers and inter-controllers), while requiring the minimum feasible number of controllers, within a negligible execution time.

1.4. Organization

The remainder of this article is organized as follows: the problem overview and the proposed model are described in Section 2, which includes the mathematical formalization and a lower-bound definition. Section 3 defines the heuristic approach. The experimental simulation is described in Section 4, while Section 5 presents and evaluates the results. Section 6 discusses the achieved results.

2. Problem formalization

2.1. Problem overview

Many previous approaches to the CPP minimize one metric or choose the solution with the best trade-off between two metrics [1,3]. In this work, we consider that the CPP consists of determining the minimum number of controllers, where they should be placed in the network and which nodes should be assigned to each controller, while satisfying the constraints related to issues (a), (b) and (c), as referred in Section 1.1. Clearly, these issues are interdependent and a solution that simultaneously fulfils them increases the ability of the network to quickly react to events and the reliability of communications, leading to a resilient topology even in the presence of failures. Accordingly, our formalization integrates constraints to keep latencies and load distribution within imposed limits.

Regarding latency, expressed in issue (b), the most commonly used metrics in the literature are summarized next, where N is the number of nodes in the network and $d_{v,c}$ is the shortest path from node $v \in V$ to controller $c \in C$.

Average propagation latency
For a set of controllers C [1]:

$$L_{\text{avg}}(C) = \frac{1}{N} \sum_{v \in V} \min_{c \in C} d_{v,c}. \tag{1}$$

Worst-case propagation latency
For a set of controllers C [1]:

$$L_{\text{wc}}(C) = \max_{v \in V} \min_{c \in C} d_{v,c}. \tag{2}$$

Inter-controller propagation latency
For a set of controllers C [19]:

$$L_{\text{ic}}(C) = \max\,(d_{c,c'}), \forall c, c' \in C. \tag{3}$$

We assume that is enough to consider propagation latency, as it is generally the most significant part of WAN latency. Therefore, metrics (1) to (3) compute latency as a function of the shortest path distance.

Controller imbalance
The degree of imbalance of a given placement set C is measured as the difference between the maximum and minimum number of nodes assigned to a controller (n_c) [18]:

$$\text{Imb}(C) = \max_{c \in C} n_c - \min_{c \in C} n_c. \tag{4}$$

Some of these metrics are conflictual. Minimizing (3) will lead to a placement of controllers close to each other, but this will generally increase the values given by (1) and (2), so it is not trivial to find a controller placement that minimizes simultaneously all of the above metrics.

2.2. Mathematical formalization

In the following mathematical formalization, SDNs are represented as an undirected graph $G(V, E)$, where $V = \{1, 2, \ldots, N\}$ is a set of physical nodes and E is a set of edges (bidirectional links) connecting nodes. $V_c \subseteq V$ denotes the set of nodes ($v \in V$) hosting a controller.

We formalize the CPP using Binary Linear Programming, as follows:

Binary decision variables

These are defined as:

$$x_i = \begin{cases} 1, \text{if the location of node } i \in V \text{ is} \\ \text{chosen to place a controller} \\ 0, \text{otherwise} \end{cases} \tag{5}$$

$$y_{ij} = \begin{cases} 1, \text{if node } j \in V \text{ is controlled by the} \\ \text{controller located at node } i \in V \\ 0, \text{otherwise} \end{cases} \tag{6}$$

Objective function

The objective is to minimize the overall number of controllers, thus the objective function is:

$$f_o = \sum_{i \in V} x_i. \tag{7}$$

The objective is to find $\min f_o$, conditioned by the following constraints:

(a) There is, at least, one network controller:

$$\sum_{i \in V} x_i \geq 1. \tag{8}$$

(b) All nodes are assigned to controllers, while considering:

$$\sum_{i \in V} y_{ij} = 1, \forall j \in V \tag{9}$$

$$y_{ij} \leq x_i, \forall i, j \in V \tag{10}$$

$$x_i \leq y_{ii}, \forall i \in V. \tag{11}$$

These constraints ensure that: each node is controlled by one, and only one controller (9); a node only can be controlled by an existing controller (10); and that nodes are always assigned to controllers placed on them (11).

(c) The load is distributed among all controllers:

$$\sum_{j=1}^{N} R_j y_{ij} \leq Q x_i, \forall i \in V \tag{12}$$

$$\sum_{j=1}^{N} R_j y_{ij} \geq \theta x_i, \forall i \in V, \tag{13}$$

with R_j as the request demand for node $j \in V$ and, without loss of generality, we assume that all controllers have the same available capacity, which is denoted by Q; θ is the minimum load of each controller. These two constraints lead to a balanced load distribution. The capacity of a controller cannot be exceeded (12) and each controller should have at least part of its capacity occupied (13). We use (13) instead of the imbalance metric (4), as the latter does not take into consideration neither node requests, nor controller capacity.

(d) The propagation latency is limited by:

$$\frac{1}{N} \sum_{j=1}^{N} d_{i,j}\, x_i \leq \gamma, \forall i \in V \tag{14}$$

$$d_{i_1,i_2}\,(x_{i_1} + x_{i_2} - 1) \leq \delta, \forall i_1, i_2 \in V, i_1 \neq i_2 \tag{15}$$

where D is the matrix containing the shortest path distances between each pair of nodes, γ is the maximum allowed average propagation latency between controllers and nodes, and δ is the maximum allowed inter-controller latency. Constraint (14) forces the average propagation latency to be under a predefined limit—denoting this average as L_{maxavg} we have, for a set of controller placements C, $L_{\text{avg}}(C) \leq L_{\text{maxavg}}(C)$. These constraints cover the metric defined in (1). For the propagation latency between controllers, the limit is imposed by (15), computed as in (3).

(e) Variables x and y can only have binary values:

$$x_i, y_{ij} \in \{0, 1\}, \forall i, j \in V. \tag{16}$$

We emphasize that this CPP formalization includes the *Bin Packing Problem* (BPP) in the objective function (7), and in constraints (8), (9) and (12). The BPP is NP-hard, as stated in [20], meaning that there is no known algorithm to solve it in polynomial time. Thus, the CPP is also NP-hard. We prove this last claim by reducing the BPP to an instance of CPP. Let us recall that for the BPP, we are given a list of items $V = \{1, 2, \ldots, N\}$, with sizes $R_j, \forall j \in V$ and want to pack them into the minimum number of bins, each one with a capacity Q. We can create an instance of the above formalized CPP as follows: the set of nodes, node request demands and controller capacity are, respectively, the set of items, item sizes and bin capacity; the shortest path distance between two nodes, $d_{i,j}$ is equal to a constant w, while $\gamma > w$, $\delta > w$ and $\theta = 1$. Therefore, for this instance, the constraints (13), (14) and (15) are satisfied, and the optimal solution of the CPP (minimum number of controllers) is equal to the optimal solution of the BPP (minimum number of items).

2.3. Establishing a lower bound for the CPP

Martello and Toth [21] introduce two lower bounds for the BPP. The first lower bound (L1) sums the element demands, divides it by the bin capacity and rounds up. The second lower bound (L2), also sums the wasted space before dividing by the bin capacity; it starts by calculating the amount of bin capacity that is wasted (unused) in any packing, and then adds this amount to the sum of all element demands, divides the total by the bin capacity and rounds up. The lower bound L2 is tighter than L1. These are also lower bounds for the CPP, returning the minimum number of controllers needed, when considering only capacity restrictions (12). We use the L2 approach to find a lower bound LB on the number of controllers, computed as described in [22]:

$$LB = \left\lceil \frac{1}{Q}\left(w + \sum_{j \in V} R_j \right) \right\rceil, \tag{17}$$

where w is the wasted space and is computed as follows:

1. Initialize $w = 0$;
2. While V is not empty, remove the node $j \in V$ with the largest R_j and calculate the controller's residual capacity, $res = Q - R_j$, after j is assigned to a controller;
3. Remove from V all nodes j with $R_j \leq res$ and store their sum in the variable *sum*;
4. If $sum \leq res$ then $w = w + res - sum$. Else $(sum - res)$ is added to the *sum* computed for the next controller.

3. Heuristic approach

We have implemented a heuristic approach based on the above CPP formalization. The basic idea of this heuristic is to obtain the minimum number of controllers which: satisfies all controller capacity constraints; balances the load distribution among controllers and minimizes inter-nodes distances to obtain acceptable latencies from nodes to their assigned controller and also between controllers. Therefore, the minimum number of node clusters that satisfies all these conditions is constructed, and afterwards, for each cluster, the best node location regarding latencies and load is chosen to host the controller. Our heuristic is detailed in Algorithm 1, which computes the lower bound using (17) and includes the following steps:

Step A: cluster construction

ρ clusters are created in parallel (initially $\rho = LB$, and in each subsequent iteration, incremented by one), as follows:

1. For each node j, determine the savings for inserting nodes j and k_1, instead of nodes j and k_2, in the same cluster. This is computed by $\text{Savings}(j) = d(j, k_1) - d(j, k_2)$, where k_1 and k_2 are the two nearest nodes of j;
2. Sort the savings list by non-increasing order (biggest savings first). Start at the front of the list, until the end, and insert either:

Algorithm 1 CPP Heuristic

```
1: Input: $G(V, E)$, $D$, $Q$, $R_j, \forall j \in V$, $\gamma$, $\delta$ and $\theta$
2: Output: $\rho = |V_c|$, $x_i$ and $y_{ij}$, $\forall i, j \in V$

3: Use expression (17) to compute a lower bound, $LB$, as described in
   Section 2.3
4: $\rho \leftarrow LB$
5: $feasible \leftarrow 0$
6: while $feasible = 0 \wedge \frac{N}{\rho} \geq 3$ do
7:    Make clusters as described in Step A
8:    $assigned \leftarrow$ number of nodes assigned to clusters
9:    if $assigned < N$ then
10:      $\rho \leftarrow \rho + 1$
11:   else
12:      Assign controllers as described in Step B
13:      if a feasible controller assignment is found then
14:         $feasible \leftarrow 1$
15:         return $\rho = |V_c|$, $x_i$ and $y_{ij}$, $\forall i, j \in V$
16:      else
17:         $\rho \leftarrow \rho + 1$
18:      end if
19:   end if
20: end while
```

(a) Two nodes (j and k_1) in an empty cluster, if none of them are already in one and if constraints (12) are not violated (cluster capacity); or

(b) One of the nodes (j or k_1) in an existing cluster, if the other node is already in that same cluster and (12) is not violated.

3. If there are unassigned remaining nodes, they are inserted in the clusters with the minimum distance increment, assuring that constraints (12) are not violated.
4. If there are clusters that do not fulfil constraints (13), a procedure of inter-cluster reassignment of nodes is applied, which consists of moving a node (among those with the most requests) from the most loaded clusters to the infeasible clusters.

Step B: controller assignment

For each cluster, find the best node to install a controller, which is the one with the best trade-off between the minimum average distance and the minimum of the maximum distance to all other nodes.

In our approach, we have considered that the number of controllers should be less or equal to $\frac{N}{3}$ (defined at the *while* on line 6), therefore the heuristic complexity is of order $O(N \times \frac{N}{3})$.

4. Experimental setup

4.1. Network topologies

We have tested this approach on several WAN topologies from *The Internet Topology Zoo*, available at [23]. We selected 15 topologies, among those with "Primary Provenance", with different sizes and configurations. Table 1 summarizes the parameters of these networks. As we have to compute distances and shortest paths, only topologies that had geographical coordinates (or city names) for nodes could be used, as several had unnamed nodes or no coordinates. Also, only connected graphs were chosen and duplicated edges were removed. In order to test large networks, we used topologies 13 and 14, which had one node location unknown, and topology 15, with two unknown locations. These were given a latitude and longitude between their neighbouring nodes.

Table 1
Network parameters.

Ref.	Topology	N	$\lvert E\rvert$	Diameter (km)
1	Abilene	11	14	4 823.10
2	Fccn	23	25	2 420.21
3	BtEurope	24	37	11 468.77
4	AttMpls	25	56	4 814.11
5	Janet Backbone	29	45	868.85
6	Arnes	34	46	254.79
7	NetworkUsa	35	39	1 175.37
8	Geant 2010	37	56	5 784.10
9	PalmettoNet	45	64	617.88
10	Surfnet	50	68	395.17
11	Iris Networks	51	64	859.80
12	Uninett2010	74	101	2 489.74
13	Bestel	84	93	4 312.6
14	Viatel 2011_02	92	96	1 579.66
15	Tata	145	186	3 866.81

4.2. Node requests and controller capacity

Node requests and available controller capacity are hard to quantify, as they depend on available hardware resources and the context of an SDN application. As in [14], we chose to measure requests and capacity in kilo-requests/s and consider that, due to failures in the network, some nodes may need to change their controller on-the-fly, which requires each controller to have, as contingency, a minimum of reserved free capacity. Thus, assuming a total capacity of 1800 kreq/s, as mentioned in [14], we established two different typical values for controller capacity, $Q_1 = 1250$ and $Q_2 = 1500$ kreq/s. Nodes were simulated to support different request rates, so instead of a static value of 200 kreq/s, as used in [14], requests were randomly generated with a rate between 180 and 220 kreq/s.

4.3. Parameters

For simplicity, it is assumed that $\gamma = \delta$. This value represents a limit related to propagation latency, so it depends on the shortest paths and network topology. Defining the *Diameter* of a network as the maximum shortest path between any two nodes in the network, we have considered two values for those limits $\gamma_1 = \frac{3}{4} \cdot Diameter$ and $\gamma_2 = \frac{2}{3} \cdot Diameter$. We set $\theta = \frac{Q}{2}$.

In total, we tested 60 scenarios, across 15 networks and with four parameter variations: two different controller capacities (Q_1 and Q_2), and two different propagation limits (γ_1 and γ_2).

4.4. Internet2 OS3E topology

The Internet2 OS3E topology [24] is widely adopted in the literature to evaluate the CPP. Thus, we have also used it to test our heuristic approach and compare the results with other solutions in the literature. This topology has 34 nodes, 42 edges and the *Diameter* is equal to 5071.6 km. The node coordinates and the distance matrix were obtained from [25]. We have defined the node requests and controller capacity as mentioned above (in Section 4.2). We have considered two scenarios: one with thresholds Q_1 and γ_1 and the other with Q_2 and γ_2.

5. Results

The results obtained by the heuristic approach are presented and analysed below.

Table 2
Number of controllers.

Ref.	Topology	Q_1			Q_2		
		LB	$\rho - LB$		LB	$\rho - LB$	
			γ_1	γ_2		γ_1	γ_2
1	Abilene	2	0	0	2	0	0
2	Fccn	4	0	0	4	0	0
3	BtEurope	4	0	0	4	0	0
4	AttMpls	5	0	0	4	0	0
5	Janet Backbone	5	0	0	4	1	1
6	Arnes	6	0	0	5	0	0
7	NetworkUsa	6	0	0	5	0	0
8	Geant 2010	6	1	1	5	1	1
9	PalmettoNet	8	0	1	7	0	0
10	Surfnet	9	0	0	7	1	1
11	Iris Networks	9	0	0	7	1	1
12	Uninett2010	12	1	1	10	1	1
13	Bestel	14	0	0	12	0	0
14	Viatel 2011_02	15	2	[a]	13	4	[a]
15	Tata	24	1	[a]	20	1	2

[a]A feasible solution was not found, due to the stricter inter-controller propagation latency limit of γ_2.

5.1. Minimum number of controllers

Table 2 shows the number of controllers obtained by the Lower Bound (LB) and by the heuristic solution (ρ) for the 60 scenarios. The columns related to the heuristic ($\rho - LB$) present the difference between the number of controllers obtained by the heuristic and LB. Therefore, when the difference is equal to zero the heuristic solution is proven to be optimal, and when equal to one the solution is, at least, near optimal.

Analysing Table 2, we can elaborate the following conclusions:

1. The heuristic algorithm obtained a feasible solution for 95% of the scenarios (57 out of 60), with only 3 exceptions. These involved networks 14 and 15, under a tighter latency limit (γ_2), where it was impossible to satisfy the inter-controller latency limit;
    - In 61.67% of the scenarios (37 out of 60), involving 11 different networks, the heuristic solution is proven to be an optimal one, since it is equal to the LB. Only in 4 networks (8, 12, 14 and 15), for all scenarios, a proven optimal solution was not found;
    - In 17 scenarios (28.33%) only one additional controller is used by the heuristic solution. These are also probably (not provably) optimal solutions, or at least near optimal;
    - In total, for 90% of the scenarios (54 out of 60), the heuristic was able to find either an optimal solution or one with only 1 additional controller. Only for 6 scenarios (10%) involving two networks (14 and 15), more than one extra controller was needed, or a feasible solution was not found by the heuristic;
2. More optimal solutions were obtained for the tighter capacity limit (Q_1) than for Q_2;
3. Concerning latency limits, only in 6 scenarios (10%), γ_2 led to a higher $\rho - LB$ than γ_1, obtaining a lower number of controllers in two scenarios and the same result for all the other scenarios. This means that the tighter limit can be used without significantly worsening the solution quality.

In conclusion, this heuristic approach shows to be efficient and effective, obtaining for 90% of the cases an optimal or near-optimal solution, in a very small computation time.

5.2. Propagation latency

Our approach constrains solutions to fulfil certain limits for the average propagation latency, between nodes and controllers (14), and between controllers (15). Thus, any solutions are always below these imposed limits. Nevertheless, Fig. 1 shows the percentage ratios of these constraints and of metrics (1) and (2) to *Diameter*, for the heuristic solutions concerning scenario Q_1 and γ_1. To simplify the plot, only this scenario was considered, as solutions of all 15 networks were obtained and the differences between scenarios are very small, being equal in most of the cases. The figure shows that the average latency ratios, *Average*, are always very low (under 20%), except for the first network (25.79%), which is also the smallest. The lowest percentage is 5.24%, obtained for the largest topology. There is, with a few exceptions, an inverse relation between these ratios and the number of controllers, the latter being directly dependent of the network dimension. Concerning the maximum average latency ratios, *Maxavg*, all values are below 48.71%, and thus far away from the imposed limit of γ_1. The inverse relation of ratios to number of controllers is also present in the worst-case latency ratios, *Worst-case*, which are between 26.03% and 60.1%. Regarding the inter-controller latency, *ICL*, only the solution for Network 15 (70.77%) is near the γ_1 limit (75%). All other networks are bellow, and 8 of them (more than a half) are below 47%. The lowest value is 5.56%, obtained in Network 3. The results show that the compliance with constraints (14) and (15) leads to low values for metrics L_{avg} and L_{wc}. It is also worth noting that there is not an observable relation between these values and network topologies, nor dimension.

To better ascertain the quality of the heuristic solution obtained for scenario Q_1 and γ_1, Fig. 2 shows boxplots depicting normalized propagation latencies: (a) from each node to its assigned controller, and (b) between each pair of controllers. The first considers N values for each network and shows that at least 3/4 of the values are below 50% (except for Network 5, with 55%); the medians are always lower than 30%, and even lower than 10% for 6 networks. In (b), the $(\rho(\rho-1)/2)$ values are considered for each network: we also have at least 3/4 of the values below 50%, while Networks 1 to 5 have all values below 40%, and only Networks 6 and 14 have the median above 30%.

5.3. Controller load balancing

Our approach obliges solutions to assign nodes to controllers ensuring a minimum load (rate of node requests) for each controller (13). Therefore, in each solution, a balanced load distribution among controllers has been obtained. As expected, the results show that the imbalance metric, computed as in (4) is also low. In almost half of the scenarios (29 out of 60), the imbalance is minimal, equal to 0 or 1 (the latter only happens when the number of nodes is not a multiple of the number of controllers). In the presented scenarios, the number of nodes assigned to controllers varies between 4 and 6, or between 4 and 7, respectively, for controller capacities Q_1 or Q_2. Fig. 2(c) shows the load distribution among controllers, obtained from the heuristic solution for scenario Q_1 and γ_1 (ρ values are considered for each network). The load is well balanced among controllers, and only Network 8 shows a larger distribution, between 800 and 1225 kreq/s, which is reasonable.

In summary, this heuristic approach has proved to be very efficient and scalable, concerning the number of controllers required, obtaining optimal or near-optimal solutions, and the quality of those solutions, towards latencies and load balancing. It is also a flexible approach, as other limits can be imposed, according to the requirements of each particular network.

5.4. Results for the Internet2 OS3E topology

The minimum number of controllers required, for the two scenarios under consideration (Q_1, γ_1 and Q_2, γ_2), obtained by the heuristic was 6 and 5, which are equal to their respective LB. Thus, they are proven to be an optimal solution considering the randomly generated node requests and the defined controller capacities. Fig. 3 depicts the

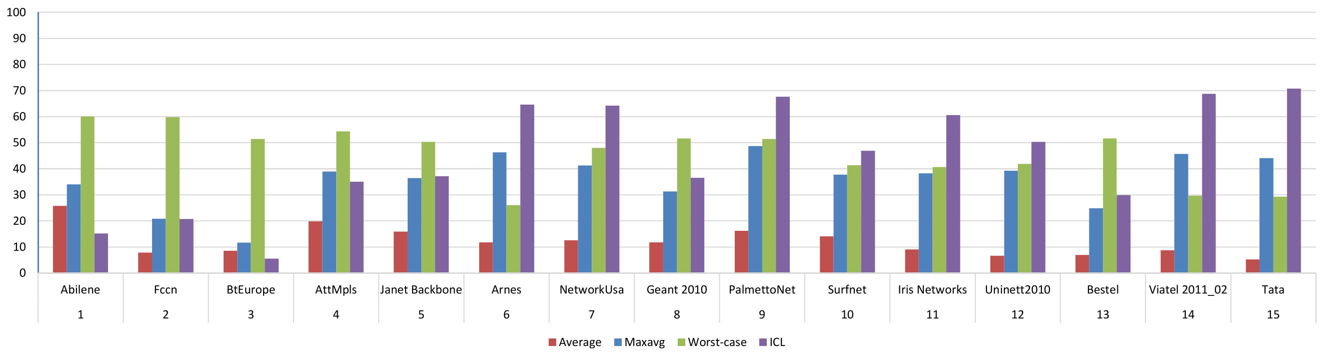

Fig. 1. Normalized Propagation Latency (%), as ratios to *Diameter* of the Average Latency (1), Maximum Average Latency (14), and Worst-Case Latency (2), from nodes to controllers; and of Inter-Controller Latency (15). Obtained from the heuristic solution for scenario Q_1 and γ_1.

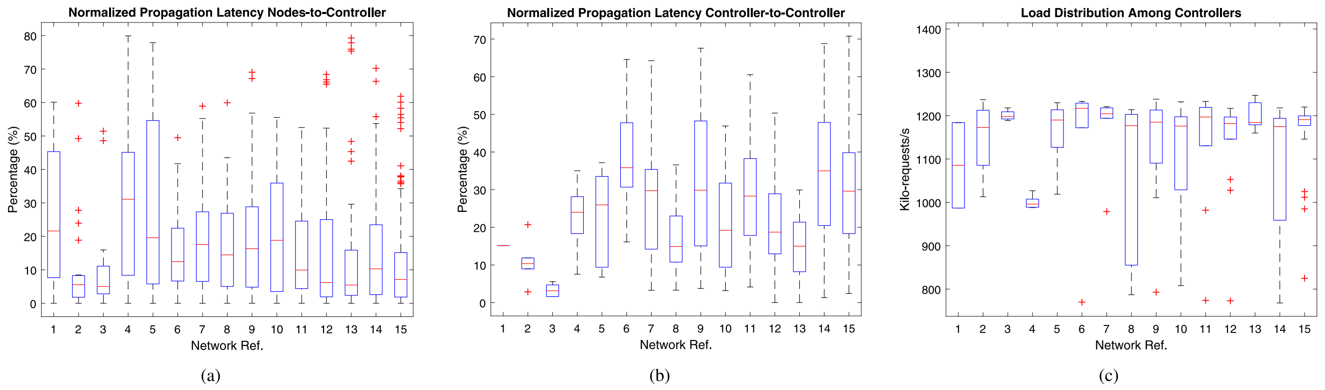

Fig. 2. Propagation latencies as ratios to *Diameter*, obtained from the heuristic solution for scenario Q_1 and γ_1: (a) from each node to its assigned controller (%), and (b) between each pair of controllers. The load distribution among controllers is shown in (c).

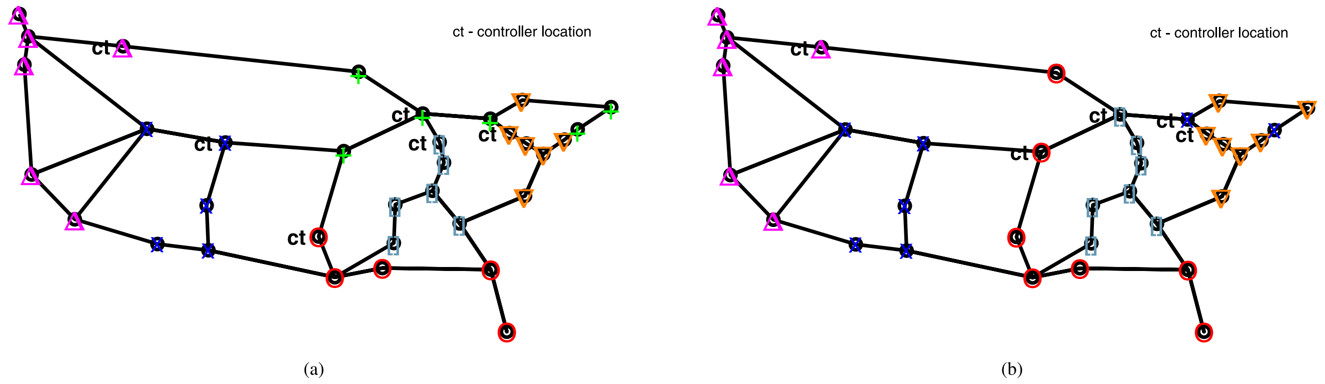

Fig. 3. Controller placement and respective node assignment, obtained by the heuristic, considering: (a) Q_1, γ_1 and (b) Q_2, γ_2.

controller placement and node assignment to controllers, obtained by the heuristic.

Other works have tested their methods by varying the number of controllers to be placed. Therefore, we should only compare our solution with others that also use 5 or 6 controllers, namely [1,8,11,18]. Heller et al. [1] used a K-Center approach and the optimal controller placement for 5 controllers is achieved considering two cases, the average-latency optimization and the worst-case latency optimization. Their solution to the latter case is the same as presented in [8,18]. Lange et al. [18] refer that this is also the solution obtained with the POCO framework for 5 controllers. The solution depicted in [11], also for 5 controllers, is based on a Cooperative K-means algorithm. Wang et al. [8] apply their clustering-based network partition algorithm (CNPA) to 5 and 6 controllers.

Fig. 4 shows normalized propagation latencies (%) as ratios to *Diameter* of metrics (1), (2) and (3) for the other solutions and for our heuristic solution, named here as K-center, CNPA, Cooperative and Heur, respectively. It is worth to point out that these other approaches aim to minimize propagation latencies between nodes and controllers, while our approach aims to simultaneously determine the minimum number of controllers that provide a balanced load distribution among them, while satisfying controller capacity, propagation limits of latencies between nodes and controllers as well as between controllers. The figure shows that our heuristic obtained average and worst-case latencies worse than the others but far from the imposed thresholds, 66,7% and 75%. Regarding the inter-controller latency, the heuristic also obtained values at least 5% below thresholds, while all other methods are between 75% and 76%, thus their solutions are not feasible for the randomly generated node requests and defined controller capacity.

As node assignment to controllers is not depicted in [1], we only compare the network balancing with the other works. The imbalance

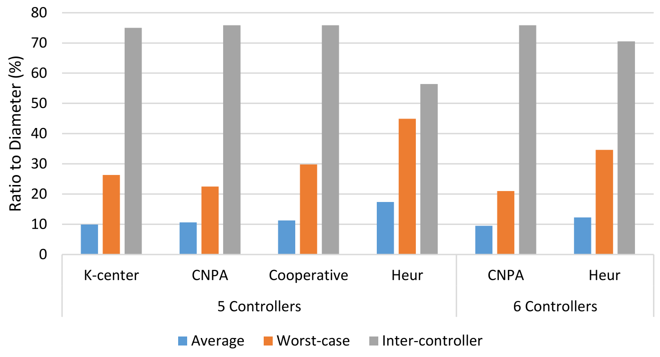

Fig. 4. Normalized propagation latencies (%) as ratios to *Diameter* of the Average Latency (1) and Worst-Case Latency (2) from nodes to controllers; and of Inter-Controller Latency (15).

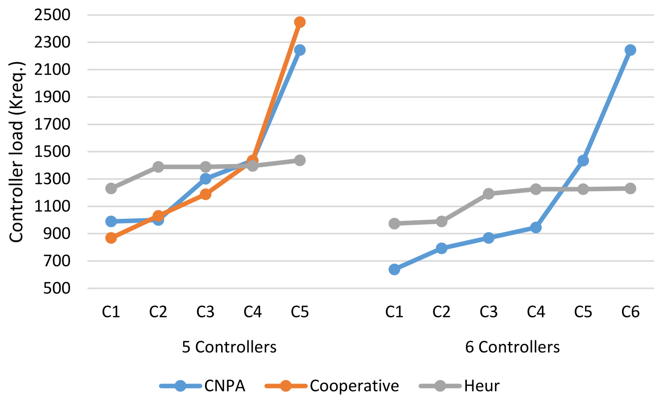

Fig. 5. Load distribution among controllers.

metric computed as (4) is equal to: 1 for Heur, in both scenarios; 8 for Cooperative; 6 and 8 for CNPA—respectively for 5 and 6 controllers. This shows that Heur achieves the smallest possible imbalance at the expense of higher latencies, while the others have too high imbalance values, allowing for lower propagation latencies. To better evaluate the imbalance, we plot in Fig. 5 the load distribution among controllers. The figure shows that our heuristic obtained a balanced load distribution among controllers, with load amplitudes equal to 206 and 256 kreq/s for 5 and 6 controllers, respectively, while the other methods are very imbalanced, with load amplitudes of 1254, 1580 (for 5 controllers) and 1605 kreq/s (for 6 controllers). Again, the other solutions are not feasible as the controller capacities (1250 and 1500 kreq/s) are largely exceeded.

In summary, by focusing only on a partial solution to the CPP, the K-centre, CNPA and Cooperative algorithms obtain better average and worst-case latencies than Heur, but a much higher inter-controller latency, a much larger number of nodes per controller and a very unbalanced load distribution among controllers.

5.5. Execution times

The heuristic algorithm was implemented in Matlab, taking an average execution time of 10.1 ms, and varying between 2.2 ms and 41.8 ms, on an Intel i5-4300U CPU (Mobile Haswell, 15 W TDP, 2C/4T, 1.90 GHz nominal and max turbo of 2.90 GHz). Table 3 shows the execution times, in milliseconds, for the 60 scenarios and for the Internet2 OS3E topology. We do not include execution times for the 3 scenarios where no feasible solution was obtained—the heuristic concluded that it could not find a feasible solution in 191.3 ms and 134.7 ms (for Q_1 and Q_2 of network 14) and in 359.6 ms (for Q_1 of network 15).

Table 3
Heuristic execution times.

Ref.	Topology	Q_1		Q_2		Mean
		γ_1	γ_2	γ_1	γ_2	
1	Abilene	*41.8*	10.0	6.8	6.4	16.3
2	Fccn	18.2	4.9	4.3	4.4	8.0
3	BtEurope	7.2	4.5	4.7	5.9	5.6
4	AttMpls	8.5	4.6	3.6	3.5	5.1
5	Janet Backbone	13.1	*30.3*	*19.6*	*24.7*	21.9
6	Arnes	5.5	5.0	2.6	**2.9**	4.0
7	NetworkUsa	**2.6**	5.6	**2.2**	3.4	**3.4**
8	Geant 2010	18.7	9.5	6.1	8.4	10.7
9	PalmettoNet	8.8	9.5	2.7	3.8	6.2
10	Surfnet	13.0	10.0	6.1	4.9	8.5
11	Iris Networks	4.8	4.5	6.0	4.9	5.0
12	Uninett2010	17.0	11.4	8.6	7.4	11.1
13	Bestel	11.3	**3.7**	3.7	3.1	5.4
14	Viatel 2011_02	36.6	[a]	19.0	[a]	*27.8*
15	Tata	31.6	[a]	7.9	18.5	19.3
Internet2 OS3E		8.7	–	–	11.4	10.1

Column minima are represented in **bold**, and maxima in *italic*.
All values are in milliseconds.
[a]A feasible solution was not found.

Analysing Table 3, we can see that the heuristic is always very fast. It took from 2.2 ms, for Network 7 under the loosest scenario (Q_2 and γ_1) to 41.8 ms for Network 1 under the tightest controller capacity (Q_1). Considering all scenarios, the mean execution time varies between 3.4 ms (Network 7) and 27.8 ms (Network 14), with the most time consuming scenario usually associated with the lowest controller capacity (Q_1, γ_1). The worst execution time occurred on Network 1 (the smallest, with 11 nodes), under Q_1 and γ_1; for the other three scenarios, the worst time was on Network 5, with 29 nodes. Two of the best execution times occurred in Network 7, with 35 nodes, which has also the best overall mean.

It seems that there is no visible relation between heuristic execution times and the number of network nodes, as the topology, latencies and controller capacities seem more important in determining execution times. It is worth to note that, for any dimension and parameters tested, the execution time is always very short and probably negligible. This is of great importance for real-world scenarios, making it possible to adapt the heuristic parameters on-the-fly and immediately find the most appropriate solution.

6. Conclusion

This article presents a mathematical formalization and a heuristic approach to find the minimum number of controllers required in an SDN topology, as well as their optimal placement location, while keeping propagation latencies from nodes to controllers, and inter-controllers, under defined limits, and ensuring a balanced load distribution among controllers. The results show that the proposed approach is able to find, in almost all of the tested topologies (90%), an optimal or near-optimal solution. Their quality regarding the average and worst-case latencies is quite good, and the processing load is well balanced among controllers. The heuristic approach proved to be computationally efficient and scalable, as its global performance is independent of network topology and dimension. Comparison with other works also confirms the viability of this approach. Therefore, we can say that the proposed approach can be used to effectively and efficiently solve the considered CPP of SDNs, under the assumptions discussed in this article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was developed within the Center for Electronic, Optoelectronic and Telecommunications (CEOT), and supported by the Portuguese Science and Technology Foundation (FCT), within the UID/MULTI/00631/2019 project and national funds of DL57/2016/CP1361/CT0040.

References

[1] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN '12, ACM, New York, NY, USA, 2012, pp. 7–12, http://dx.doi.org/10.1145/2342441.2342444, URL http://doi.acm.org/10.1145/2342441.2342444.

[2] G. Yao, J. Bi, Y. Li, L. Guo, On the capacitated controller placement problem in software defined networks, IEEE Commun. Lett. 18 (8) (2014) 1339–1342, http://dx.doi.org/10.1109/LCOMM.2014.2332341.

[3] Y. Jiménez, C. Cervelló-Pastor, A.J. García, On the controller placement for designing a distributed sdn control layer, in: 2014 IFIP Networking Conference, 2014, pp. 1–9, http://dx.doi.org/10.1109/IFIPNetworking.2014.6857117.

[4] B.P.R. Killi, S.V. Rao, Capacitated next controller placement in software defined networks, IEEE Trans. Netw. Serv. Manag. 14 (3) (2017) 514–527, http://dx.doi.org/10.1109/TNSM.2017.2720699.

[5] Y. Zhang, L. Cui, W. Wang, Y. Zhang, A survey on software defined networking with multiple controllers, J. Netw. Comput. Appl. 103 (2018) 101–118, http://dx.doi.org/10.1016/j.jnca.2017.11.015, URL http://www.sciencedirect.com/science/article/pii/S1084804517303934.

[6] J. Lu, Z. Zhang, T. Hu, P. Yi, J. Lan, A survey of controller placement problem in software-defined networking, IEEE Access 7 (2019) 24290–24307, http://dx.doi.org/10.1109/ACCESS.2019.2893283.

[7] L. Han, Z. Li, W. Liu, K. Dai, W. Qu, Minimum control latency of sdn controller placement, in: 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 2175–2180, http://dx.doi.org/10.1109/TrustCom.2016.0334.

[8] G. Wang, Y. Zhao, J. Huang, Y. Wu, An effective approach to controller placement in software defined wide area networks, IEEE Trans. Netw. Serv. Manag. 15 (1) (2018) 344–355, http://dx.doi.org/10.1109/TNSM.2017.2785660.

[9] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, Towards an elastic distributed sdn controller, SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 7–12, http://dx.doi.org/10.1145/2534169.2491193, URL http://doi.acm.org/10.1145/2534169.2491193.

[10] Y. Hu, T. Luo, W. Wang, C. Deng, On the load balanced controller placement problem in software defined networks, in: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 2430–2434, http://dx.doi.org/10.1109/CompComm.2016.7925135.

[11] B.P.R. Killi, E.A. Reddy, S.V. Rao, Cooperative game theory based network partitioning for controller placement in sdn, in: 2018 10th International Conference on Communication Systems Networks (COMSNETS), 2018, pp. 105–112, http://dx.doi.org/10.1109/COMSNETS.2018.8328186.

[12] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, Pareto-optimal resilient controller placement in sdn-based core networks, in: Proceedings of the 2013 25th International Teletraffic Congress (ITC), 2013, pp. 1–9, http://dx.doi.org/10.1109/ITC.2013.6662939.

[13] Y. Hu, W. Wendong, X. Gong, X. Que, C. Shiduan, Reliability-aware controller placement for software-defined networks, in: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 672–675.

[14] 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: 2014 IEEE Global Communications Conference, 2014, pp. 1909–1915, http://dx.doi.org/10.1109/GLOCOM.2014.7037087.

[15] G. Schütz, Computational Science – ICCS 2019, in: ICCS 2019. Lecture Notes in Computer Science, vol. 11536, Springer, Cham, 2019, Ch. A k-Cover Model for Reliability-Aware Controller Placement in Software- Defined Networks.

[16] B. Zhang, X. Wang, M. Huang, Multi-objective optimization controller placement problem in internet-oriented software defined network, Comput. Commun. 123 (2018) 24–35, http://dx.doi.org/10.1016/j.comcom.2018.04.008, URL http://www.sciencedirect.com/science/article/pii/S0140366416307241.

[17] V. Ahmadi, M. Khorramizadeh, An adaptive heuristic for multi-objective controller placement in software-defined networks, Comput. Electr. Eng. 66 (2018) 204–228, http://dx.doi.org/10.1016/j.compeleceng.2017.12.043, URL http://www.sciencedirect.com/science/article/pii/S0045790617307565.

[18] 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 Trans. Netw. Serv. Manag. 12 (1) (2015) 4–17, http://dx.doi.org/10.1109/TNSM.2015.2402432.

[19] E. Borcoci, R. Badea, S.G. Obreja, M. Vochin, On multi-controller placement optimization in software defined networking - based wans, in: ICN 2015 : The Fourteenth International Conference on Networks, 2015.

[20] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1990.

[21] S. Martello, P. Toth, Lower bounds and reduction procedures for the bin packing problem, Discrete Appl. Math. 28 (1) (1990) 59–70, http://dx.doi.org/10.1016/0166-218X(90)90094-S, URL http://www.sciencedirect.com/science/article/pii/0166218X9090094S.

[22] R.E. Korf, A new algorithm for optimal bin packing, in: Eighteenth National Conference on Artificial Intelligence, American Association for Artificial Intelligence, Menlo Park, CA, USA, 2002, pp. 731–736, URL http://dl.acm.org/citation.cfm?id=777092.777205.

[23] The internet topology zoo, 2018, URL http://www.topology-zoo.org/.

[24] Internet2 open science, scholarship and services exchange, 2019, URL http://www.internet2.edu/network/ose/.

[25] Lehrstuhl für informatik III - Universität Würzburg, 2019, URL https://github.com/lsinfo3/poco/tree/master/topologies.