Improving East/Westbound SDN Connectivity via Enhanced Availability

Dorabella Santos
INESC Coimbra
DEEC, Rua Sílvio Lima, Polo 2,
3030-290 Coimbra, Portugal
dorabella.santos@gmail.com

João P. Vidal
INESC Coimbra
DEEC, Rua Sílvio Lima, Polo 2,
3030-290 Coimbra, Portugal
joaosantosvidal_1994@hotmail.com

Teresa Gomes, Lúcia Martins
University of Coimbra, INESC Coimbra,
Department of Electrical and Computer Engineering,
Rua Sílvio Lima, Polo 2,
3030-290 Coimbra, Portugal
{teresa,lucia}@deec.uc.pt

***Abstract*—The control plane in software-defined networking (SDN) is composed by a logically centralized set of controllers which acts as the brain of the SDN network. These controllers need to be connected through east/westbound interfaces in order to have a synchronized view of the network and therefore it is a key issue to guarantee the intercontroller connectivity for a proper network behaviour.**

In the present work, the problem of controller placement under QoS constraints with the additional requirement of having the intercontroller connectivity with a given availability at minimum cost is addressed as a bi-objective problem. This problem is NP-complete and to tackle it a heuristic approach is proposed based on the problem decomposition into sub-problems. In a previous heuristic to solve a less general problem, a similar strategy was followed based on the assumption that a Steiner tree connecting the controllers is the set of links that should be upgraded to achieve the required availability. In this paper a more general sub-graph is considered for intercontroller availability improvement, and an additional optimization level is also considered to address this more general problem.

To assess the performance of the proposed heuristic a baseline heuristic to serve as a benchmark is also proposed.

***Index Terms*—SDN, controller placement, availability, heuristics, integer linear programming, bi-objective**

I. Introduction

Since SDN has an important role in several network technologies including 5G, the control plane availability is of key importance for the proper functioning of the network [1]. An inherent problem in SDN is known as the controller placement problem (CPP), which consists in determining how many controllers are needed and where to place them in the network. This problem was formally introduced in [2] and has been extensively studied in the literature.

Having only one controller can lead to resilience issues, since the single point of failure in the network is the controller. Moreover, in large networks, the controller can quickly be overloaded and/or the latencies between the data plane switches and the controller can be quite large, leading to scalability issues. Deploying multiple controllers as a logically centralized set, improves resilience and scalability but also raises some challenges. While increasing the number of controllers helps decrease switch-controller latencies, one of the challenges is that deploying a large number of controllers can lead to important intercontroller communication overhead [1], [3]. Therefore, in several works, maximum delay values have been imposed between the switches and controllers and between also the controllers themselves, to ensure acceptable control plane performance [4]–[6].

Intercontroller communication is ensured by an east/westbound interface, allowing the controllers to have a full and detailed view of the topology. Since the controllers are responsible for the management and decision making in the SDN network, it is essential to guarantee intercontroller availability requirements. In [7], a "five-nines" availability, which is typically required in recent communication networks [8], is guaranteed for the southbound, i.e., between the switches and the controllers that manage them.

In our previous work [6], we proposed a heuristic to solve the NP-complete optimization problem of finding the controller placement under delay constraints, and finding a Steiner tree to be the sub-graph, whose links can be upgraded to have improved availability at a given cost, in order to achieve availability targets. We considered intercontroller "five-nines" availability requirements with path redundancy: for each pair of controllers, the pair of primary and backup paths must ensure that the end-to-end availability is at least 0.99999. In [9], we also studied a similar problem, but the availability targets were between the controllers and the switches they manage. In that work, the sub-graph was a spanning tree which made the problem easier to formulate and solve.

Although path redundancy does increase end-to-end availability [10], it may not be sufficient to ensure the desired availability alone [11]. In [7], the authors show that each switch has to connect to 2 or 3 controllers to achieve the desired availability. This is the motivation to consider a sub-graph which is based on the spine concept, proposed in [12]. The links of this sub-graph can undergo an upgrade in which its availability is increased, and that can be achieved either by reducing the mean time to repair (e.g. allocating closer maintenance centers) or by making the link more robust so

This work is funded by ERDF Funds through the Centre's Regional Operational Program and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. under the project CENTRO-01-0145-FEDER-029312. This work was also partially supported by FCT under project grant UIDB/00308/2020.

that the mean time between failures is increased (e.g. by burying an aerial cable).

In [13], the authors address a CPP variant for resilience against single link failures between the controllers. They also consider a Steiner tree for the intercontroller communication paths, but in their case they do not consider intercontroller delay constraints, nor do they deal with availability guarantees. Finally, in [14] a multi-objective optimization problem concerning both controller placement and path availability is presented. They consider the availability for the paths between switches and controllers, whereas we consider between the controllers themselves. Moreover, the problem of selecting links to have enhanced availability is not considered.

In this paper, the problem of controller placement under QoS constraints with the additional requirement of having the intercontroller connectivity with a given availability at minimum cost is addressed as bi-objective optimization problem. This optimization problem is NP-complete, meaning that exact methods become computationally prohibitive. Therefore, a heuristic approach is proposed based on the one presented in [6], breaking the problem down into its constituent sub-problems. The problem addressed in [6] is a particular case of the present problem addressed herein, and was based on the assumption that a Steiner tree connecting the controllers is the set of links that should be upgraded to achieve the required availability. In this paper, a more general sub-graph is considered for intercontroller availability improvement, and a second level optimization is proposed for link availability update, to further improve the intercontroller availability update cost.

Furthermore, to assess the performance of the proposed heuristic and because a benchmark heuristic does not exist in the literature, a baseline heuristic is also put forward.

This paper is organized as follows. In Section II, the addressed global problem is formulated as a bi-objective optimization problem. In Section III, the heuristic strategy is presented in its different aspects: in Subsection III-A the CPP under delay constraints is formulated; in Subsection III-B the intercontroller availability improvement is presented; in Subsection III-C the two-level optimization model for availability update is formulated; and in Subsection III-D the bi-objective heuristic is proposed. In Section IV, a baseline heuristic is presented to serve as a benchmark for performance evaluation. In Section V, the computational results are discussed showing the gains obtained with the bi-objective heuristic, and in Section VI the conclusions are presented.

II. A Bi-Objective Optimization Problem

The controller placement and intercontroller availability optimization problem is the CPP together with the problem of link selection for availability update, among the links involved in the connections between controllers, in order to guarantee the desired availability at minimum cost. Additionally, the connections between the controllers must also satisfy delay constraints, as well as between the switches and the controllers managing them. This problem can be formulated as a bi-objective optimization problem, because the minimum number of controllers may not lead to the minimum cost solution for the intercontroller availability update. Therefore, instead of a global optimum, a set of non-dominated solutions will be obtained, that represents the trade-off between the number of controllers and the update cost, so that the decision maker can choose which solution is most appropriate for each case.

Consider that the SDN data plane is represented by a graph $G = (N, E)$, where N is the set of nodes and E is the set of links. Each link is represented by its end nodes $\{i, j\}$.

Assume that there are K different discrete levels for availability update. Each link $\{i, j\}$ can be updated to level k with cost $\breve{c}_{ij}^{k}$. Then, the problem of jointly minimizing the number of controllers and the cost of intercontroller availability update can be defined as:

Problem $\mathcal{P}$:

$$\min \sum_{i \in N} y_i \tag{1}$$

$$\min \sum_{k=1}^{K} \sum_{\{i,j\} \in E} \breve{c}_{ij}^{k} \breve{z}_{ij}^{k} \tag{2}$$

s.t.

Delay constraints

Flow continuity constraints

Intercontroller availability constraints

where

- y_i binary variable that is 1 if a controller is placed in node $i \in N$, and 0 otherwise;
- $\breve{z}_{ij}^{k}$ binary variable that is 1 if link $\{i, j\} \in E$ is updated to level k, and 0 otherwise.

The CPP is known to be NP-hard [2] and the link availability update problem at a minimum cost is NP-complete [15], which results in the addressed optimization problem being NP-complete. Therefore, it cannot be solved in practice by exact methods. Next, a heuristic approach is proposed.

III. Heuristic Strategy

To tackle the bi-objective optimization problem presented in the previous section, a heuristic strategy is herein presented based on breaking down the problem into its constituent sub-problems: (i) the CPP under switch-controller and intercontroller delay constraints; (ii) the assumption of a sub-graph for intercontroller availability improvement, where a primary and a backup path can be defined; (iii) the link availability optimization at minimum cost, under intercontroller availability guarantees.

A. Controller Placement Problem

The CPP is one of the main sub-problems in the optimization problem. It addresses how many controllers should be placed in the network and where they are placed. An in-band control plane is assumed, where each switch is connected to a controller.

Each link $\{i,j\} \in E$ has an associated length ℓ_{ij} and an associated delay. The delay between two nodes i and j is proportional to the length of the path connecting them and is denoted by d_{ij} [2]. A maximum delay D_{sc} is imposed between each switch and the controller that manages it. Moreover, a maximum delay D_{cc} is also imposed between any two controllers. Since switch-controller communication is more frequent than intercontroller communication, it is assumed that $D_{sc} < D_{cc}$ [4].

The ILP model to solve the CPP is given by:

$$\min \sum_{i \in N} y_i \tag{3}$$

s.t.

$$\sum_{j \in N : d_{ij} \leq D_{sc}} y_j \geq 1 \quad i \in N \tag{4}$$

$$y_i + y_j \leq 1 \qquad i,j \in N : d_{ij} > D_{cc} \tag{5}$$

$$y_i \in \{0,1\} \qquad i \in N \tag{6}$$

The objective function (3) aims to minimize the number of controllers (one of the objectives of the main optimization problem). Constraints (4) guarantee that for any node $i \in N$, there is a controller placed with a delay of at most D_{sc} from it. Constraints (5) guarantee that the delay between any two controllers is at most D_{cc}. Finally, constraints (6) are the variable domain constraints.

Note that the minimum number of controllers is conditioned by D_{sc}, while D_{cc} conditions the maximum number of controllers that can be placed in the network.

B. Intercontroller Availability Improvement

Once the controller placements are obtained, it is possible to consider a sub-graph $\mathcal{S}$ connecting the controllers which can be obtained in different ways as explained ahead. The availability of the links belonging $\mathcal{S}$ can be updated to guarantee the availability requirements at minimum cost.

To increase availability between controllers, a primary path and a node-disjoint backup path are guaranteed between each pair of controllers. Consider $\mathcal{S}_P \subset \mathcal{S}$ and $\mathcal{S}_B \subset \mathcal{S}$ as the sub-graphs that support the primary and backup paths, respectively, where $\mathcal{S} = \mathcal{S}_P \cup \mathcal{S}_B$. It is noted that $\mathcal{S}_\mathcal{P} \cap \mathcal{S}_B$ may not be empty. The primary paths must have a delay of at most D_{cc} routed over $\mathcal{S}_P$ (the delay constraint is relaxed for the backup paths).

An availability minimum of Λ is required for the intercontroller connections, i.e., the intercontroller availabilities achieved by the pair of primary and backup paths must be at least Λ. This is not always possible by using path protection alone. Therefore, we consider that the availability of the links belonging $\mathcal{S}$ can be updated at a given cost.

Consider that each link $\{i,j\} \in E$ has a default distance-based availability a_{ij}^0, which depends on the link length ℓ_{ij}, the mean time to repair (MTTR) and the mean time between failures (MTBF) (details in [6]). The availability of a path is given by the product of the availabilities of the elements of the path (i.e., of the links and nodes belonging to the path). In the present work, the nodes are assumed to not fail, i.e., the node availabilities are equal to one.

Consider that each link of $\mathcal{S}_P$ (for the primary paths between the controllers) can be upgraded to K different levels of increased availability, where for each level the link unavailability is decreased by a factor $\epsilon \in (0,1)$. This can be achieved by decreasing the MTTR (e.g. allocating closer maintenance centers), or increasing the MTBF (e.g. making the link more robust). The availability of link $\{i,j\} \in \mathcal{S}_P$ in level k can be given as an expression dependent of level $k-1$:

$$a_{ij}^k = a_{ij}^{k-1} + \varepsilon \left(1 - a_{ij}^{k-1}\right) \quad k = 1, ..., K. \tag{7}$$

The cost for upgrading the link availability to level k (in relation to the default link availability) is given by [15], [16]:

$$c_{ij}^k = -\ell_{ij} \cdot \ln \left(\frac{1 - a_{ij}^k}{1 - a_{ij}^0} \right) \quad k = 1, ..., K \tag{8}$$

The cost function translates an exponential increase as the link availability is upgraded to the next level.

Analogously, consider that each link of $\mathcal{S}_B$ (for the backup paths between the controllers) can be downgraded to have decreased availability, where the unavailability is increased by ϵ (in this case, only one level of downgrade is assumed). This can happen by increasing the MTTR (e.g. when moving maintenance centers closer to the links of $\mathcal{S}_P$, these centers can be further away from the links of $\mathcal{S}_B$).

The decreased availability of link $\{i,j\} \in \mathcal{S}_B$ is given by:

$$\hat{a}_{ij} = a_{ij}^0 - \varepsilon \left(1 - a_{ij}^0\right) \tag{9}$$

with a (negative) cost given by:

$$\hat{c}_{ij} = -\ell_{ij} \cdot \ln \left(\frac{1 - \hat{a}_{ij}}{1 - a_{ij}^0} \right) \tag{10}$$

The reason why we consider the downgrade of links of $\mathcal{S}_\mathcal{B}$ as part of the intercontroller availability improvement is explained in the next Subsection.

C. Link Availability Optimization

The problem of updating the link availabilities of sub-graph $\mathcal{S}$ is solved with a two-level optimization model, minimizing the cost of the intercontroller availability update, while guaranteeing the required availability of Λ. Consider $\mathcal{C}$ as the set of controllers. For each pair $c_1, c_2 \in \mathcal{C}$, denote the primary path availability as $A_{c_1c_2}^p$ and the backup path availability as $A_{c_1c_2}^b$. Then, the availability of the pair of paths should be at least Λ, i.e., $1 - (1 - A_{c_1c_2}^p)(1 - A_{c_1c_2}^b) \geq \Lambda$. This is achieved by upgrading some of the links of $\mathcal{S}_P$.

Consider $A_{c_1c_2}^{b0}$ the default availability of the backup path between controllers c_1 and c_2, given by the default availabilities of the links belonging to the path. Then, the necessary primary path availability $A_{c_1c_2}^p$ to achieve Λ is given by $A_{c_1c_2}^p = (\Lambda - A_{c_1c_2}^{b0})/(1 - A_{c_1c_2}^{b0})$.

Denote the primary path between c_1 and c_2 as $p_{c_1c_2}$. Consider the following decision variables:

z_{ij}^k binary variable that is 1 if link $\{i,j\} \in \mathcal{S}_P$ is upgraded to level $k = 0, ..., K$, and 0 otherwise. Note that $k = 0$ means that the link is not upgraded.

The first level ILP model is given by:

$$\min \sum_{k=1}^{K} \sum_{\{i,j\} \in \mathcal{S}_P} c_{ij}^k z_{ij}^k \quad (11)$$

s.t.

$$\sum_{k=0}^{K} z_{ij}^k = 1 \qquad \{i,j\} \in \mathcal{S}_P \quad (12)$$

$$\sum_{k=0}^{K} \sum_{\{i,j\} \in p_{c_1c_2}} z_{ij}^k \ln(a_{ij}^k) \geq \ln(A_{c_1c_2}^p) \quad c_1, c_2 \in \mathcal{C} \quad (13)$$

$$z_{ij}^k \in \{0,1\} \qquad \{i,j\} \in \mathcal{S}_P,\ k = 0, ..., K \quad (14)$$

The objective function (11) aims to minimize the upgrade cost of $\mathcal{S}_P$. Constraints (12) guarantee that each link of $\mathcal{S}_P$ is in only one level $k = 0, ..., K$.

Constraints (13) guarantee that the links of each primary path $p_{c_1c_2}$ are upgraded to achieve the necessary availability $A_{c_1c_2}^p$. The constraints are not linear, but can be linearized with logarithms. Finally, constraints (14) are the variable domain constraints.

Consider the solution obtained by the first level ILP model. Denote $\nu_{\mathcal{S}_P}$ as the value of the objective function and a'_{ij} as the availability of link $\{i,j\} \in \mathcal{S}_P$ in the solution. Denote $A_{c_1c_2}^{p'}$ as the primary path availabilities recomputed considering the updated link availabilities a'_{ij}.

Any link availability over-provisioning obtained in the first level ILP model, can be adjusted in the second level ILP model. Consider $A_{c_1c_2}^b$ as the necessary backup path availabilities given by:

$$A_{c_1c_2}^b = \frac{\Lambda - A_{c_1c_2}^{p'}}{1 - A_{c_1c_2}^{p'}} \quad (15)$$

Denote the backup path between c_1 and c_2 as $b_{c_1c_2}$. Consider the following decision variables:

$\hat{z}_{ij}$ binary variable that is 1 if link $\{i,j\} \in \mathcal{S}_B \backslash \mathcal{S}_P$ is downgraded, and 0 otherwise.

The second level ILP model is given by

$$\min \sum_{\{i,j\} \in \mathcal{S}_B \backslash \mathcal{S}_p} \hat{c}_{ij} \hat{z}_{ij} \quad (16)$$

s.t.

$$\hat{z}_{ij} = 0 \qquad \{i,j\} \in \mathcal{S}_B \cap \mathcal{S}_P \quad (17)$$

$$\sum_{\{i,j\} \in b_{c_1c_2}} \left[(1 - \hat{z}_{ij}) \ln(a'_{ij}) + \hat{z}_{ij} \ln(\hat{a}_{ij})\right] \geq \ln(A_{c_1c_2}^b) \quad c_1, c_2 \in \mathcal{C} \quad (18)$$

$$\hat{z}_{ij} \in \{0,1\} \qquad \{i,j\} \in \mathcal{S}_B \quad (19)$$

The objective function (16) is the minimization of the downgrade of the links of $\mathcal{S}_B \backslash \mathcal{S}_P$. The final link availability update cost is thus the sum of the obtained cost in (16) and $\nu_{\mathcal{S}_P}$.

Constraints (17) guarantee that only the links from $\mathcal{S}_B \backslash \mathcal{S}_P$ can be downgraded (not those of $\mathcal{S}_B \cap \mathcal{S}_P$).

Constraints (18) guarantee that the links of each backup path are downgraded while still satisfying the necessary availability $A_{c_1c_2}^b$. These constraints have been linearized using logarithms, similarly to constraints (13). Constraints (19) are the variable domain constraints.

D. Bi-Objective Heuristic

Next, the global heuristic for the bi-objective problem is presented. Its pseudo-code is shown in Algorithm 1.

The "best" solutions for the bi-objective problem are the non-dominated solutions w.r.t. to the number of controllers and the link availability update cost. A solution with C controllers and link update cost μ, is said to be non-dominated, if any solution with a lower cost than μ has more than C controllers, and any solution with less than C controllers has a higher cost than μ.

Algorithm 1: Pseudo-code for bi-objective heuristic

```
1  VCP ← ∅;
2  Sols ← ∅;
3  while continue = true do
4      CP ← CPP(D_sc, D_cc)\VCP;
5      if CP = ∅ then
6          continue ← false;
7      else
8          Determine S = S_P ∪ S_B for CP set;
9          for c_1, c_2 ∈ CP do
10             Compute backup path availability A^{b0}_{c_1,c_2};
11             A^p_{c_1c_2} ← (Λ − A^{b0}_{c_1c_2})/(1 − A^{b0}_{c_1c_2});
12         (μ, L(S_P)) ← 1st levelILP(S_P, {A^p_{c_1c_2}});
13         if L(S_P) ≠ ∅ then
14             if μ = 0 then
15                 continue ← false;
16             else
17                 for c_1, c_2 ∈ CP do
18                     Recompute primary path availability A^{p'}_{c_1c_2};
19                     A^b_{c1c_2} ← (Λ − A^{p'}_{c1c_2})/(1 − A^{p'}_{c1c_2});
20                 (μ', L(S_B)) ← 2nd levelILP(S_B, {A^b_{c_1c_2}});
21                 μ ← μ + μ'
22             Sols ← Sols ∪ CP;
23         VCP ← VCP ∪ CP;
```

The heuristic exploits the fact that different controller placements provide different sub-graphs $\mathcal{S}$ (even if the sub-graphs are of the same type, for example, Steiner trees) and, different sub-graphs provide different link availability update solutions. A different set of controller placements can be obtained by removing the previous solutions from the search space of the CPP, and finding a new CPP solution for which a new cost for the link availability problem should be obtained.

The heuristic starts by solving the CPP with the ILP model defined by (3)-(6) (step 4). The visited set of controller placements $\mathcal{VCP}$, i.e., the set of already obtained controller placements is empty, as well as the set of solutions $Sols$ (steps 1 and 2).

After the controller placement $\mathcal{CP}$ has been obtained, the heuristic determines sub-graphs $\mathcal{S}_P$ for the intercontroller

primary paths and $\mathcal{S}_B$ for the respective backup paths (step 8). In the computational results (Section V), two sub-graphs were considered: $\mathcal{S}_1$ where $\mathcal{S}_{P1}$ is the set of intercontroller shortest paths; and $\mathcal{S}_2$ where $\mathcal{S}_{P2}$ is the minimum distance intercontroller Steiner tree. Sub-graph $\mathcal{S}_{P1}$ is obtained using Dijkstra's algorithm, while $\mathcal{S}_{P2}$ is obtained heuristically with Takahashi's algorithm [17] where the terminal nodes are the controllers. For both cases, $\mathcal{S}_B$ is the set of minimum distance shortest node-disjoint paths to the respective primary ones, obtained with a modification of Dijkstra's algorithm. For subgraph $\mathcal{S}_2$, if any primary path has a delay greater than D_{cc} over the Steiner tree, $\mathcal{S}_2$ is discarded (omitted in the pseudo-code for clarity) for that $\mathcal{CP}$ set.

Then for each considered sub-graph $\mathcal{S}$, the availability of the backup path $A^{b0}_{c_1c_2}$ between controllers c_1 and c_2 is computed, assuming the default link availabilities a^0_{ij} (step 10). The necessary primary path availabilities $A^{p}_{c_1c_2}$ are then computed, considering $A^{b0}_{c_1c_2}$, in order to achieve the target path pair availability of at least Λ (step 11).

The 1st level ILP defined by (11)-(14) is then solved considering $A^{p}_{c_1c_2}$, and the set of links $\mathcal{L}(\mathcal{S}_P)$ of $\mathcal{S}_P$ can be upgraded with a given cost μ (step 12). If the problem is infeasible, i.e., $\mathcal{L}(\mathcal{S}_P) = \emptyset$, the $\mathcal{CP}$ set is discarded.

Otherwise, if the upgrade cost μ is zero, the "ideal" solution has been achieved and the heuristic stops (steps 14 and 15). This is a valid stopping criteria, since the CPP starts with the minimum number of controllers and generates increasingly larger sets of controllers. Since the optimal solutions are non-dominated for the number of controllers and the cost, no solution with an equal or greater number of controllers can dominate the one with $\mu = 0$. Also, since no link has been upgraded, the 2nd level ILP is not needed. Before the heuristic stops, the "ideal" solution is added to the set of obtained solutions $Sols$ with cost $\mu = 0$ (step 22).

If $\mu > 0$, then the 2nd level ILP defined by (16)-(19) is solved (step 20). To do so, the primary path availabilities $A^{p'}_{c_1c_2}$ are recomputed considering the upgraded solution $\mathcal{L}(\mathcal{S}_P)$ (step 18). Then, the necessary backup path availabilities $A^{b}_{c_1c_2}$ are computed, assuming $A^{p'}_{c_1c_2}$ to ensure the target path pair availability of at least Λ (step 19). The $\mathcal{CP}$ set is then added to the set of obtained solutions $Sols$ with the cost μ obtained in step 21.

Finally, the $\mathcal{CP}$ set is added to the set of visited CPP solutions $\mathcal{VCP}$, to eliminate it from the search space in step 4, when generating a new controller placement. This is done by adding a set of constraints to the ILP model (3)-(6). Consider that the controller placement set is $\{\rho_1, ..., \rho_C\}$, where C is the minimum number of controllers. Then the additional constraint is given by:

$$y_{\rho_1} + \cdots + y_{\rho_C} \leq C - 1 \tag{20}$$

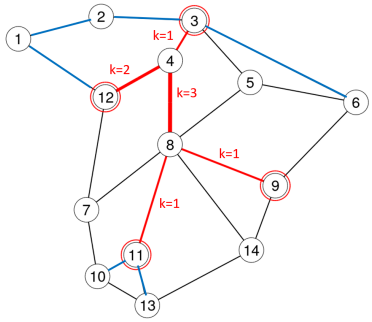
Such a constraint is added for each new controller placement set with C controllers. Eventually, no more C controller placements satisfying D_{sc} and D_{cc} is possible, and so a new controller placement with $C' = C+1$ controllers is generated. In this case, the following constraint is added:

$$\sum_{i \in N} y_i \geq C' \tag{21}$$

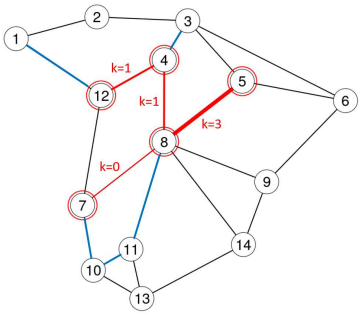
replacing all constraints of type (20). After that, new constraints of that type will be added with C' controllers. When the ILP model returns infeasible, it means that all the controller placements satisfying D_{sc} and D_{cc} were obtained, and so the heuristic stops (steps 5 and 6).

In the end, the set $Sols$ holds all the solutions obtained and the non-dominated ones are chosen (the term 'non-dominated' is used here loosely in the sense that they are the non-dominated solutions of set $Sols$, but there can exist other solutions for the same problem that dominate the obtained solutions, but are not found by the heuristic.).

Consider Fig. 1, where the two non-dominated solutions for the given topology are shown. The sub-graph considered was $\mathcal{S}_2$, i.e., $\mathcal{S}_P$ is the intercontroller minimum distance Steiner tree. The controller nodes are represented with red outer circles and the Steiner tree links are represented in red, with the thickness of its links being proportional to the level of upgrade. The downgraded links of $\mathcal{S}_B$ are shown in blue.



$C = 4 \quad \mu = 619.93$



$C = 5 \quad \mu = 450.70$

Fig. 1. Non-dominated solutions for the spain network with given D_{sc} and D_{cc} values, using $\mathcal{S}_2$ for link availability update

Note that the top solution has $C = 4$ controllers and a cost of $\mu = 619.93$, resulting from upgrading 3 links of the Steiner tree to level $k = 1$, 1 link to level $k = 2$ and 1 link to level $k = 3$, and downgrading 6 links. The bottom solution has $C = 5$ controllers and a cost of $\mu = 450.70$, resulting from upgrading 2 links of the Steiner tree to level $k = 1$ and 1 link to level $k = 3$ (and one link is not upgraded, $k = 0$), and downgrading 5 links.

IV. Benchmark for Performance Evaluation

A benchmark heuristic does not exist in the literature for the optimization problem addressed in this paper. Therefore, to assess the performance of the proposed heuristic (which is also compared to the one in [6]), a baseline heuristic is put forward to serve as a benchmark. The rationale for the baseline heuristic is to forgo ILP models, except for obtaining the minimum number of controllers $C_{\min}$ (which is an input parameter for this heuristic), defined ahead. The pseudo-code is shown in Algorithm 2.

The CPP is solved without resorting to ILP models. To do so, two approaches based on the k-center algorithm [18] were developed: the greedy approach and the modified k-center approach (modified to account for D_{cc}).

For a fixed root node ρ and a given C, the greedy approach places the first controller c_1 in node ρ. The second controller c_2 is placed at the node furthest away from ρ, but within the maximum delay D_{cc}. The third controller c_3 is placed at the furthest node from c_1 and c_2, within the maximum delay of D_{cc} from both of them. The process is repeated until C controllers have been placed in the network.

The (modified) k-center approach starts by considering ρ as a candidate node for the first controller c_1. The cluster $\mathcal{C}\ell_1$ is the set of nodes within D_{sc} from ρ. If there exists a node $c^* \in \mathcal{C}\ell_1$ such that any node $u \in \mathcal{C}\ell_1$ is within D_{sc} from c^* and such that the average delay from c^* to all nodes in $\mathcal{C}\ell_1$ is less than that of ρ, then c^* is the next candidate node for c_1 and the cluster $\mathcal{C}\ell_1$ is updated. The process is repeated until no such node c^* is found. Then, the first controller node is set.

The candidate node for the second controller c_2 is chosen to be the node furthest away from $\mathcal{C}\ell_1$, but within D_{cc} from c_1. The cluster $\mathcal{C}\ell_2$ is obtained as the set of nodes within D_{sc} from c_2. If c^* exists within $\mathcal{C}\ell_2$, such that any node $u \in \mathcal{C}\ell_2$ is within D_{sc} from c^* and such that the average delay from c^* to all nodes in $\mathcal{C}\ell_2$ is less than that of c_2, then it becomes the next candidate for c_2 and cluster $\mathcal{C}\ell_2$ is updated. The process is repeated until no such node c^* exists within $\mathcal{C}\ell_2$, and so the second controller node is set.

Then, the third controller candidate c_3 is chosen as the node furthest away from clusters $\mathcal{C}\ell_1 \cup \mathcal{C}\ell_2$, but within D_{cc} from c_1 and c_2. The cluster $\mathcal{C}\ell_3$ is obtained and the process is repeated until C controller nodes have been set.

In both approaches, if there exists a node $i \in N$ such that the closest controller has a delay greater than D_{sc} from i, the controller set is considered invalid.

The baseline heuristic starts by considering the first root node $\rho \in N$ (step 2) and solves the CPP using the greedy approach (step 5) and the minimum number of controllers $C_{\min}$ (step 4). If the controller placement set $\mathcal{CP}$ is invalid, then the number of controllers is incremented (step 10), since the set may become valid if more controllers are placed. However, if $C \geq C_{\min} + 2$, then the $\mathcal{CP}$ has the potential of being poorly chosen and it does not pay off to set more controllers, as the cost in the update problem will become very large.

Algorithm 2: Pseudo-code for benchmark heuristic

```
1  Sols ← ∅;
2  for ρ = 1...n do
3      continue ← true;
4      C ← C_min;
5      CPPtype ← greedy;
6      while continue = true do
7          CP ← CPPtype(ρ, C);
8          if CP = ∅ then
9              if C < C_min + 2 then
10                 C ← C + 1;
11             else
12                 if CPPtype = greedy then
13                     C ← C_min;
14                     CPPtype ← k-center;
15                 else
16                     continue ← false;
17         else
18             Determine S_P as a Steiner tree and S_B for CP;
19             for c_1, c_2 ∈ CP do
20                 Compute backup path availability A^{b0}_{c_1,c_2};
21                 A^p_{c_1c_2} ← (Λ − A^{b0}_{c_1c_2})/(1 − A^{b0}_{c_1c_2});
22             (μ, L(S_P)) ← 1^{st}level(S_P, {A^p_{c_1c_2}});
23             if L(S_P) ≠ ∅ then
24                 Sols ← Sols ∪ CP;
25                 C ← C + 1;
26             else
27                 if C < C_min + 2 then
28                     C ← C + 1;
29                 else
30                     if CPPtype = greedy then
31                         C ← C_min;
32                         CPPtype ← k-center;
33                     else
34                         continue ← false;
```

The heuristic goes back to $C_{\min}$ controllers, now using the k-center approach (steps 13 and 14). If $\mathcal{CP}$ is invalid and $C \geq C_{\min} + 2$, then the heuristic chooses another root node ρ (step 16 which goes to step 2). Otherwise, if in any CPP approach the $\mathcal{CP}$ set is valid, the intercontroller Steiner tree is determined (step 18), using Takahashi's algorithm. If any primary path does not satisfy D_{cc} over the Steiner tree, the tree is invalid and the CPP solution is discarded (omitted in the pseudo-code for clarity).

Otherwise, the backup path availabilities assuming default link availabilities are determined (step 20), and the necessary primary path availabilities are computed (step 21). The link availability update problem (step 22) is solved in a greedy manner. At first, all Steiner tree links are not upgraded, i.e, the links are in level $k = 0$. Then, each pair of controllers such that the pair of primary and backup paths does not achieve an end-to-end availability of at least Λ, are identified in set $\mathcal{C}_{<\Lambda}$. We then identify the link $\{i, j\}^*$ occurring most frequently in the primary paths of $\mathcal{C}_{<\Lambda}$ (since upgrading this link will enhance the availability of more than one pair of controllers); if there is more than one, we choose the longest link (since we are considering distance-based availabilities). Then $\{i, j\}^*$ is upgraded to $k = 1$. The end-to-end availabilities of $\mathcal{C}_{<\Lambda}$

are recalculated and the set is updated.

While $\mathcal{C}_{<\Lambda} \neq \emptyset$, we identify the link $\{i,j\}^*$ occurring most frequently in the primary paths of $\mathcal{C}_{<\Lambda}$ such that its upgrade level k is less than K; if there is more than one, we choose the link with lowest level of upgrade k; if there is more than one, we choose the longest link. Then, $\{i,j\}^*$ is upgraded to the next level from which it is currently in. The process is repeated until $\mathcal{C}_{<\Lambda} = \emptyset$, meaning that the target availability of at least Λ has been achieved between all pairs of controllers. The solution is added to $Sols$ (step 24) and the number of controllers is incremented (step 25).

If $\{i,j\}^*$ does not exist but $\mathcal{C}_{<\Lambda} \neq \emptyset$, this means that all the primary path links in $\mathcal{C}_{<\Lambda}$ are already upgraded to the maximum level $k = K$. In this case, the problem is infeasible (for the used Steiner tree), meaning that the target end-to-end availability is not achievable, and the current CPP solution is discarded as invalid. The heuristic proceeds as when the $\mathcal{CP}$ set is invalid (steps (27)-(34)).

The heuristic stops in the following conditions: (i) if all nodes have served as root nodes (step 2); (ii) if for a given value C' a solution with zero cost has been found, then the following CPP solutions only need to be obtained for $C < C'$, since no solution $C \geq C'$ can have a cost lower than the 'ideal' zero cost. From the set of solutions in $Sols$, we choose non-dominated ones.

V. Computational Results

To evaluate the performance of the bi-objective heuristic and compare it with the one in [6] and the benchmark heuristic, the following networks were considered: polska and cost266 networks from SNDlib [19], and the spain network from [20]. The topological characteristics of the networks are summarized in Table I, which shows the number of nodes, the number of edges, the average node degree and the graph diameter (longest shortest path between any two nodes) for each network.

TABLE I
Topological characteristics of the networks

Network	#nodes	#links	avg deg	D_g [km]
polska	12	18	3.00	811
spain	14	22	3.14	1034
cost266	37	57	3.08	4032

All heuristics and procedures were implemented in C/C++, using the CPLEX 12.9 Callable libraries for solving the ILP models. All results were obtained on an Intel Core i7 laptop with 8 GB of RAM, running at 2.9 GHz. The maximum delay values D_{sc} and D_{cc} are given as percentages of the graph diameter D_g [2], [4], [5], and were considered to be 35%, 40%, 45% for D_{sc}, and 65%, 70%, 75% for D_{cc}, for all networks (to be equal for all networks and consistent with our previous works [6], [9]). For all heuristics, the sub-graph $\mathcal{S}_2$ such that $\mathcal{S}_{P2}$ is a Steiner tree between the controllers was considered, since it is in principal the minimum cost sub-graph connecting the controllers (as it has no cycles). In the bi-objective heuristic, sub-graph $\mathcal{S}_1$ such that $\mathcal{S}_{P1}$ is the set of intercontroller shortest paths was also considered.

For all the networks, the intercontroller availability guarantee value was considered to be 'five-nines', i.e., $\Lambda = 0.99999$. To obtain the desired availability levels, $K = 4$ levels of link availability upgrade for $\mathcal{S}_P$ were considered with a link unavailability reduction factor of $\varepsilon = 0.5$ per level. In the bi-objective heuristic, the second level ILP for link availability update for $\mathcal{S}_B$ considers one level of downgrade.

The computational results are shown in Tables II, III and IV for polska, spain and cost266, respectively. In all tables, the results are shown for the benchmark heuristic, the heuristic in [6], and the proposed bi-objective heuristic. Each problem instance is defined by the network topology and the given D_{sc} and D_{cc} values.

Since the heuristics cannot guarantee optimality, the term 'non-dominated solutions' is used to refer to the non-dominated solutions in set $Sols$ of the respective heuristic. This means a non-dominated solution of one heuristic can be dominated by a solution of another heuristic. Therefore, each value of C shown in the tables, starting with $C_{\min}$, corresponds to a non-dominated solution of at least one heuristic. For each non-dominated solution, the cost is shown, as well as, the total number of upgraded links (column '#upg'), the number of links upgraded to each level $k = 1, 2, 3, 4$ (columns 'k'). Finally, the last column shows the computational runtime in seconds (column 't(s)'). For the proposed heuristic the number of downgraded links in $\mathcal{S}_B$ (second level of link availability optimization) is also shown (column '#dng').

The runtimes for the heuristic in [6] are smaller than reported before, since there was an error in the implementation of the stopping criteria, and so the runtimes reported in [6] were unnecessarily high for spain and especially for cost266.

In Table II for the polska network, only the proposed heuristic found a solution for $D_{sc} = 35\%$ and $D_{cc} \leq 70\%$, because the Steiner tree sub-graph $\mathcal{S}_{P2}$ cannot guarantee the D_{cc} delay for all the intercontroller primary paths, while $\mathcal{S}_{P1}$ always exists and guarantees that the intercontroller primary paths have delays within D_{cc}. For $D_{sc} = 35\%$ and $D_{cc} = 75\%$, the greedy algorithm for the availability upgrade in the benchmark heuristic, did not succeed in finding a solution that guarantees the minimum intercontroller availability. The other two heuristics found the ideal optimum of zero cost with the minimum number $C_{\min} = 3$ of controllers. The same ideal optimum was found by all heuristics, for all the instances with $D_{sc} = 40\%$ and for the instance with $D_{sc} = 45\%$ and $D_{cc} = 65\%$. For $D_{cc} \geq 70\%$, the value of $C_{\min}$ is 2 and a non-zero cost solution exists. The two heuristics found the same cost solution which is lower than the cost found by the benchmark. The solution with 3 controllers continues to have zero cost in these instances. Note that when $C = 2$, the sub-graphs $\mathcal{S}_1$ and $\mathcal{S}_2$ are the same. Note that the runtimes are very small for the three heuristics, but slightly higher for the proposed heuristic.

In Table III for the spain network, the proposed heuristic

TABLE II
COMPUTATIONAL RESULTS FOR POLSKA

Instance			Benchmark							Heuristic [6]							Proposed Heuristic							
D_{sc}	D_{cc}	C	cost	#upg	k				t(s)	cost	#upg	k				t(s)	cost	#upg	k				#dng	t(s)
					1	2	3	4				1	2	3	4				1	2	3	4		
35%	65%	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0.02
	70%	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0.67
	75%	3	-	-	-	-	-	-	-	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0	1.76
40%	65%	3	0	0	0	0	0	0	0.03	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	0.15
	70%	3	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0.06	0	0	0	0	0	0	0	0.15
	75%	3	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0.08	0	0	0	0	0	0	0	0.28
45%	65%	3	0	0	0	0	0	0	0.03	0	0	0	0	0	0	0.21	0	0	0	0	0	0	0	0.40
	70%	2	300.39	2	2	0	0	0	0.08	274.86	2	2	0	0	0	0.12	274.86	2	2	0	0	0	0	0.51
		3	0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	
	75%	2	-	-	-	-	-	-	0.05	274.86	2	2	0	0	0	0.12	274.86	2	2	0	0	0	0	0.30
		3	0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	

TABLE III
COMPUTATIONAL RESULTS FOR SPAIN

Instance			Benchmark							Heuristic [6]							Proposed Heuristic							
D_{sc}	D_{cc}	C	cost	#upg	k				t(s)	cost	#upg	k				t(s)	cost	#upg	k				#dng	t(s)
					1	2	3	4				1	2	3	4				1	2	3	4		
35%	65%	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	532.65	5	5	0	0	0	3	0.24
	70%	4	1640.37	5	2	2	0	1	0.07	1212.31	5	3	1	1	0	0.52	619.93	5	3	1	1	0	6	1.79
		5	-	-	-	-	-	-		850.49	3	2	0	1	0		450.70	3	2	0	1	0	5	
	75%	4	1640.37	5	2	2	0	1	0.10	1212.31	5	3	1	1	0	0.91	619.93	5	3	1	1	0	6	3.54
		5	-	-	-	-	-	-		850.49	3	2	0	1	0		450.70	3	2	0	1	0	5	
40%	65%	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	532.65	5	5	0	0	0	3	0.23
	70%	4	1640.37	5	2	2	0	1	0.02	999.52	3	1	1	1	0	0.64	388.89	3	1	1	1	0	6	1.89
		5	-	-	-	-	-	-		850.49	3	2	0	1	0		-	-	-	-	-	-	-	
	75%	4	1640.37	5	2	2	0	1	0.03	999.52	3	1	1	1	0	1.09	388.89	3	1	1	1	0	6	3.96
		5	-	-	-	-	-	-		850.49	3	2	0	1	0		-	-	-	-	-	-	-	
45%	65%	3	850.88	4	3	1	0	0	0.06	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	0.53
	70%	3	999.21	4	2	2	0	0	0.03	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	1.35
	75%	3	999.21	4	2	2	0	0	0.06	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	1.98

found solutions for $D_{sc} = 35\%, 40\%$ and $D_{cc} = 65\%$ with 5 controllers, while the other heuristics did not, because the Steiner trees of sub-graph $\mathcal{S}_{P2}$ cannot guarantee the D_{cc} delay between all pairs of controllers for those instances. For the other instances, the heuristic in [6] found solutions with lower cost than the benchmark, and the proposed heuristic found solutions with even lower cost. Note that for $D_{sc} = 40\%$ and $D_{cc} \geq 70\%$, the proposed heuristic found the best solution with the lowest cost of 388.89 and $C_{\min} = 4$ controllers. This solution corresponds to the sub-graph $\mathcal{S}_2$ for the same controller placement found by the heuristic in [6] with cost 999.52, where a total of 3 links where upgraded. In the proposed heuristic, an additional 6 links were downgraded, decreasing the cost to 388.89. In fact, this solution dominates all solutions found being the best solution obtained in these cases, since it is the solution with lowest cost for $C_{\min}$. Both heuristics found the optimal solution of zero cost with $C_{\min} = 3$ for all instances with $D_{sc} = 45\%$, while the benchmark obtained solutions with non-zero cost. The runtimes remain very small for the three heuristics, but continue to be slightly higher for the proposed heuristic.

In Table IV for the cost266 network, the benchmark cannot find a solution for the instances with $D_{sc} = 35\%$ (the greedy algorithm for availability upgrade fails to produce a valid solution). For the other instances, the benchmark finds solutions with cost higher than the other two heuristics. The proposed heuristic finds solutions as good or better than the one in [6]. Note that when $C = 2$, downgrade of link availability is not possible, meaning that the upgraded links do not translate in over-provisioning of intercontroller availability. This is also observed for $D_{sc} = 45\%$ with $C = 3$ controllers.

It was observed that most non-dominated solutions in the proposed heuristic result from using the sub-graph $\mathcal{S}_2$. However, when delays are too restrictive, an intercontroller Steiner tree such that D_{cc} is satisfied for each pair of controllers may not be possible, and so, $\mathcal{S}_2$ may not exist. Nevertheless, $\mathcal{S}_1$ always exists, making this sub-graph preferable in these situations. Although the runtimes for the proposed heuristic are the highest among the three approaches, they are still quite competitive.

VI. CONCLUSIONS

The bi-objective optimization problem for joint controller placement and link availability update, under delay and availability constraints was presented. Since the problem is NP-complete, a bi-objective heuristic approach was proposed based on breaking down the problem into its constituent sub-problems. A previous heuristic to solve a particular case of the optimization problem was used to compare with the proposed approach. Moreover, a benchmark was also proposed to evaluate the performance of both heuristics. We note that

TABLE IV
COMPUTATIONAL RESULTS FOR COST266

Instance			Benchmark							Heuristic [6]							Proposed Heuristic							
D_{sc}	D_{cc}	C	cost	#upg	k				t(s)	cost	#upg	k				t(s)	cost	#upg	k				#dng	t(s)
					1	2	3	4				1	2	3	4				1	2	3	4		
35%	65%	3	-	-	-	-	-	-	-	5658.47	7	0	0	1	6	4.66	5262.50	7	0	0	1	6	3	6.38
		4								5317.30	7	0	0	3	4		4031.01	7	0	0	3	4	9	
	70%	3	-	-	-	-	-	-	-	5658.47	7	0	0	1	6	3.33	5262.50	7	0	0	1	6	3	4.84
	75%	3	-	-	-	-	-	-	-	5658.47	7	0	0	1	6	3.56	5262.50	7	0	0	1	6	3	5.47
40%	65%	2	4195.39	4	0	0	2	2	0.12	4136.10	4	0	0	1	3	2.23	4136.10	4	0	0	1	3	0	3.78
		3	-	-	-	-	-	-		4032.84	6	0	0	4	2		3615.04	6	0	0	4	2	4	
	70%	2	4195.39	4	0	0	2	2	0.12	4136.10	4	0	0	1	3	2.49	4136.10	4	0	0	1	3	0	3.85
		3	-	-	-	-	-	-		4032.84	6	0	0	4	2		3615.04	6	0	0	4	2	4	
	75%	2	4195.39	4	0	0	2	2	0.12	4136.10	4	0	0	1	3	2.46	4136.10	4	0	0	1	3	0	3.67
		3	-	-	-	-	-	-		4032.84	6	0	0	4	2		3615.04	6	0	0	4	2	4	
45%	65%	2	2446.46	4	0	0	4	0	0.14	1402.71	3	0	2	1	0	2.14	1402.71	3	0	2	1	0	0	3.59
	70%	2	2446.46	4	0	0	4	0	0.70	1402.71	3	0	2	1	0	2.61	1402.71	3	0	2	1	0	0	3.96
		3	-	-	-	-	-	-		1352.36	3	0	2	1	0		1352.36	3	0	2	1	0	0	
	75%	2	2446.46	4	0	0	4	0	0.071	1402.71	3	0	2	1	0	3.57	1402.71	3	0	2	1	0	0	5.04
		3	-	-	-	-	-	-		1352.36	3	0	2	1	0		1352.36	3	0	2	1	0	0	

there are no available algorithms in the literature to tackle this problem, except [6] and [9], since it has not been addressed by other authors. However, we have considered existing algorithms for the subproblems, which were integrated into our benchmark heuristic (namely Takahashi's algorithm and the k-center algorithm).

In the proposed bi-objective heuristic, two sub-graphs for link availability improvement were considered and an additional optimization level was formulated to improve the intercontroller availability update cost. The computational results show that the proposed heuristic is able to obtain as good or better solutions than the previous heuristic and benchmark, with negligible runtime increase. It was observed that most non-dominated solutions found by the proposed heuristic results from using the sub-graph $\mathcal{S}_2$ for availability improvement. However, an intercontroller Steiner tree may not always be viable when the required delay is too low. By contrast, $\mathcal{S}_1$ will always be viable, making it the alternative strategy for this situation.

REFERENCES

[1] T. Zhang, A. Bianco, and P. Giaccone, "The role of inter-controller traffic in SDN controllers placement," in *NFV-SDN*, Palo Alto, USA, 2016, pp. 87–92.

[2] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *ACM HotSDN*, New York, USA, 2012, pp. 7–12.

[3] A. S. Muqaddas, P. Giaccone, A. Bianco, and G. Maier, "Inter-controller traffic to support consistency in ONOS clusters," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1018–1031, 2017.

[4] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *DRCN*, Paris, France, 2016, pp. 145–151.

[5] M. Tanha, D. Sajjadi, R. Ruby, , and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[6] D. Santos, J. P. Vidal, T. Gomes, and L. Martins, "A heuristic method for controller placement and enhanced availability between SDN controllers," in *Networks of the Future*, Bordeaux, France, 2020, pp. 1–9.

[7] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *ACM HotSDN*, New York, USA, 2014, pp. 31–36.

[8] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, *Availability evaluation of a virtualized IP multimedia subsystem for 5G network architectures*. CRC Press: Taylor & Francis, 06 2017, pp. 2203–2210.

[9] D. Santos, T. Gomes, and D. Tipper, "Software-defined network design driven by availability requirements," in *DRCN*, Milan, Italy, 2020, pp. 1–7.

[10] S. Song, H. Park, B. Choi, T. Choi, and H. Zhu, "Control path management framework for enhancing software-defined network (SDN) reliability," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 302–316, 2017.

[11] A. Alashaikh, T. Gomes, and D. Tipper, "The spine concept for improving network availability," *Computer Networks*, vol. 82, pp. 4 – 19, 2015.

[12] D. Tipper, "Resilient network design: challenges and future directions," *Telecommunication Systems*, vol. 56, no. 1, pp. 5–16, 2014.

[13] T. Das and M. Gurusamy, "Controller placement for resilient network state synchronization in multi-controller SDN," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1299–1303, 2020.

[14] B. Zhang, X. Wang, and M. Huang, "Multi-objective optimization controller placement problem in internet-oriented software defined network," *Computer Communications*, vol. 123, pp. 24–35, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416307241

[15] A. Alashaikh, D. Tipper, and T. Gomes, "Embedded network design to support availability differentiation," *Annals of Telecommunications*, vol. 74, no. 9-10, pp. 605–623, 2019.

[16] U. Franke, "Optimal it service availability: Shorter outages, or fewer?" *IEEE Transactions on Network and Service Management*, vol. 9, no. 1, pp. 22–33, 2012.

[17] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Mathematica Japonica*, vol. 24, no. 6, p. 573–577, 1980.

[18] Y. Jiménez, C. Cervelló-Pastor, and A. J. García, "On the controller placement for designing a distributed SDN control layer," in *IFIP*, Trondheim, Norway, 2014, pp. 1–9.

[19] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0–Survivable Network Design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010, http://sndlib.zib.de.

[20] R. Martinez, R. Casellas, R. Vilalta, and R. Muñoz, "GMPLS/PCE-controlled multi-flow optical transponders in elastic optical networks [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. 71–80, 2015.