Controller Placement and Availability Link Upgrade Problem in SDN Networks*

Dorabella Santos† Teresa Gomes‡†

dorabella.santos@gmail.com teresa@deec.uc.pt

September 2019

Abstract

In SDN networks, the problem of how many controllers and where to place them, has been extensively studied. This is known as the controller placement problem, and has been addressed mainly considering the delays between the SDN switches and controllers. Although the delays between switches and controllers and the intercontroller delays are key aspects, a less addressed issue is the availability of the control paths (the routing paths between the switches and their controllers). In this paper,

*This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under project grant UID/Multi/00308/2019 and was financially supported by ERDF Funds and National Funds through FCT under project CENTRO-01-0145-FEDER-029312.

†INESC Coimbra, Coimbra, Portugal

‡University of Coimbra, Department of Electrical and Computer Engineering, Coimbra, Portugal

the problem regarding controller placements with QoS requirements, both in terms of delays and availability of the control paths, is addressed. To guarantee the availability requirements, a set of links is selected to have upgraded availability. An exact method and a heuristic method are proposed for solving the problem. Computational results show that the heuristic method provides near-optimal solutions within reasonable runtimes, when the exact method becomes computationally expensive.
Keywords: SDN, availability, controller placement problem, optimization.

1 Introduction

Software-Defined Networking (SDN) has gained significant importance for telecommunication networks. The networking paradigm decouples the data plane from the control plane, providing more flexibility for control management. The data plane consists of SDN enabled switches that are basically forwarding devices, controlled by a SDN controller. To provide robustness in the control plane, multiple controllers can be distributed in the network, while working as a centralized unit, in the logically centralized control plane architecture. In this case, each switch is managed by one of the controllers.

The problem of how many controllers and where to place them in the network, known as the controller placement problem (CPP) [5], has been extensively studied. The CPP has been considered mostly conditioned by the switch-controller (SC) delays, i.e., the delays between the switches and their controllers [8]. Since having multiple controllers implies synchronization delays between controllers, the controller-controller (CC) delays are also important, among other aspects such as controller load balancing (number of switches managed by each controller) [6]. Other works have considered the CPP in resilient contexts against failures [18], [12] and also against malicious attacks [14].

A less studied aspect in the context of SDN networks, is the issue of availability. Most works assess the availability of SDN networks using models to evaluate the availability in terms of reachability between switches and controllers [11] or in terms of network connectivity [9], [10]. Although the CPP is not addressed in [15], the work addresses control path availability and path redundancy is considered as a means to increase availability.

Fewer works consider availability in the context of the CPP. In [13], the CPP is addressed to guarantee a lower bound on the availability of the control paths (routing paths between the switches and their controllers), allowing the switches to connect to multiple controllers (one primary and one or more backup controllers). In [7], several heuristics for controller placement are proposed, aiming to maximize availability, which is given in terms of the expected percentage of control path loss. In these works, delay guarantees are not imposed in the CPP.

In this paper, the CPP is addressed considering maximum SC and CC delay values, as well as, minimum control path availability values. To achieve the required control path availability, a set of links is selected to have upgraded availability, by adding a parallel fibre to each of these links [3]. The set of such links constitutes a substructure of higher availability in the network, called the spine [1], [4]. The existence of this structure in the network provides for high resilience routing and also resiliency differentiation, in a more effective manner than just increasing availability through path redundancy [2]. Actually, path redundancy alone may not be sufficient to ensure high end-to-end availabilities for mission critical services, while providing higher than needed availability for most consumers than they are willing to pay for [17].

Therefore, the CPP variant is addressed herein which considers the control path availability requirements and is referred to, in this paper, as availability link upgrade CPP (ALU-CPP). An ILP model of the ALU-CPP is formulated, using an exact linearised version for the constraints guaranteeing the control path availability requirements. The contributions of this paper are summarized as follows:

- Formulation of the ALU-CPP: the problem jointly optimizing the controller placement and the selection of a set of links to have upgraded availability in the context of SDN networks.
- A novel way (to the best of our knowledge) of representing the exact linearized constraints for the control path availability requirements, considering link availability upgrade.
- An exact method and a heuristic method to solve the ALU-CPP.

The paper is organized as follows: the ALU-CPP is formulated as an integer linear programming (ILP) model and the linearized version of the availability requirement constraints is discussed in Section 2; an exact method and a heuristic method for solving the ALU-CPP are proposed in Section 3; computational results for four networks are presented in Section 4; and finally conclusions are drawn and some future work considerations are presented in Section 6.

2 The Availability Link Upgrade CPP

In this section, the availability link upgrade CPP (ALU-CPP) is formulated. When multiple controllers exist in a SDN network, their placement contributes directly to the control plane performance. Each switch is connected to a controller, and each controller manages a set of switches. In this work, the controllers are assumed to be collocated with the switches and the control plane is assumed to be implemented in-band (a more realistic approach for large SDN networks).

The control plane performance strongly depends on the SC and CC delays. To guarantee acceptable delay requirements, it is assumed that the SC delays cannot exceed a given maximum value D_{sc}, and similarly, the CC delays cannot exceed a given maximum value D_{cc}. Since the SC communications are more critical and much more frequent than the CC communications (also due to the larger number of switches compared to the number of controllers), it makes sense that $D_{sc} < D_{cc}$.

Besides the delay requirements, the availability of the routing path between a switch and its controller, known as the control path, is also a key aspect. One way to mitigate this issue is to consider controller redundancy [12–14,18] or path redundancy [15,18]. This work, however, focuses on improving control path availability by upgrading the availability of the links constituting the spine. The default availability of each link is considered to be distance-based [16] and given by

$$a_{ij} = 1 - \frac{MTTR}{MTBF_{ij}} \tag{1}$$

where $MTTR$ denotes the mean time to repair (in hours) and $MTBF_{ij}$ denotes the mean time (in hours) between failures of link $\{i, j\}$ which is dependent on its length ℓ_{ij} (in km) and given by

$$MTBF_{ij} = \frac{CC \times 365 \times 24}{\ell_{ij}} \tag{2}$$

The parameter CC denotes the cable cut rate and is considered to be 450 km, while $MTTR$ is considered to be 24 hours [1].

Traditionally, the improvement in availability is to add parallel redundancy. Therefore, the upgraded link availability is obtained by adding a parallel connection which is considered to have the same availability as the original link (e.g. adding a fiber of the same length) and so is given by [3]:

$$\hat{a}_{ij} = a_{ij}(2 - a_{ij}) \tag{3}$$

The set of upgraded links in this manner form the higher availability spine. The selection of such links depends on the requirement that each control path cannot have an availability less than a given minimum value λ.

2.1 Problem Formulation

The ALU-CPP can be formulated as an ILP model. Therefore, consider that the SDN data plane can be represented by a graph $G = (N, E)$, where N is the set of nodes (switches) and E is the set of links. Each link is represented by its end nodes $\{i, j\}$. Consider A to be the set of arcs or directed links. The arc directed from node i to node j is represented by the ordered pair (i, j). For each node i, the set of adjacent nodes to $i \in N$ is denoted by $V(i)$, i.e., $V(i) = \{j \in N : \{i, j\} \in E\}$. Moreover, the delay between two nodes i and j is denoted by d_{ij}.

Consider the following parameters:

- $C \in \mathbb{N}$ integer parameter that indicates the number of controllers to be placed in the network
- $t_i^s \in \{0, 1\}$ binary parameter that indicates if $i = s$

and the following decision variables:

- $y_i \in \{0, 1\}$ binary variable that is 1 if a controller is placed in node i, and 0 otherwise
- $a_i^s \in \{0, 1\}$ binary variable that is 1 if the controller of switch s is placed in node i, and 0 otherwise
- $x_{ij}^s \in \{0, 1\}$ binary variable that is 1 if arc $(i, j) \in A$ belongs to the path of switch s to its controller, and 0 otherwise
- $z_{ij} \in \{0, 1\}$ binary variable that is 1 if edge $\{i, j\}$ belongs to the spine, i.e., if the link has upgraded availability, and 0 otherwise
- $w_{ij}^s \in \{0, 1\}$ binary variable equal to $x_{ij}^s \cdot z_{ij}$

The ALU-CPP is formulated as an ILP model given by

$$\min \sum_{\{i,j\}\in E} z_{ij} \tag{4}$$

s.t.

$$\sum_{i\in N} y_i = C \tag{5}$$

$$\sum_{\substack{j\in N:\\ d_{ij}\le D_{sc}}} y_j \ge 1 \qquad i \in N \tag{6}$$

$$y_i + y_j \le 1 \qquad i \in N, j \in N : d_{ij} > D_{cc} \tag{7}$$

$$\sum_{j\in V(i)} \left(x^s_{ij} - x^s_{ji}\right) = t^s_i - a^s_i \qquad s \in N,\, i \in N \tag{8}$$

$$a^s_s = y_s \qquad s \in N \tag{9}$$

$$a^s_i = 0 \qquad s \in N,\, i \in N : d_{is} > D_{sc} \tag{10}$$

$$a^s_i \le y_i \qquad s \in N,\, i \in N : d_{is} \le D_{sc} \tag{11}$$

$$z_{ij} \le \sum_{s\in N} \left(x^s_{ij} + x^s_{ji}\right) \qquad \{i,j\} \in E \tag{12}$$

$$z_{ji} = z_{ij} \qquad \{i,j\} \in E \tag{13}$$

$$w^s_{ij} \le x^s_{ij} \qquad s \in N,\, (i,j) \in A \tag{14}$$

$$w^s_{ij} \le z_{ij} \qquad s \in N,\, (i,j) \in A \tag{15}$$

$$w^s_{ij} \ge x^s_{ij} + z_{ij} - 1 \qquad s \in N,\, (i,j) \in A \tag{16}$$

$$\sum_{\{i,j\}\in A} \left[x^s_{ij} \log(a_{ij}) + w^s_{ij} \left(\log(\hat{a}_{ij}) - \log(a_{ij})\right)\right] \ge \log(\lambda) \qquad s \in N \tag{17}$$

$$y_i \in \{0,1\} \qquad i \in N \tag{18}$$

$$x^s_{ij},\, w^s_{ij} \in \{0,1\} \qquad s \in N,\, (i,j) \in A \tag{19}$$

$$a^s_i \in \{0,1\} \qquad s \in N,\, i \in N \tag{20}$$

$$z_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{21}$$

The objective function (4) aims to minimize the number of upgraded links, which translates to minimizing the cost of upgrading the links, assuming the costs are the same for all links (the ILP can easily be extended to the general case, where the costs depend on the length of the added fibers).

Constraint (5) guarantees that C controllers are placed in the network. Constraints (6) guarantee that for any switch, there must be at least one controller located not further than D_{sc}, while constraints (7) guarantee that if two nodes are distanced further than D_{cc}, then both nodes cannot host controllers (since controllers cannot be distanced further than D_{cc}).

Constraints (8) are the flow conservation constraints of each switch $s \in N$ routed to its destination given by $a_i^s = 1$. These combine with constraints (9)-(11) to guarantee that each switch's destination in node i has a controller and is distanced at most D_{sc} from s:

- constraints (9) guarantee that if node s has a controller placed there, then the switch is managed by that controller (is routed to itself);
- if node i is distanced further than D_{sc}, constraints (10) guarantee that i cannot be the destination of s;
- otherwise, constraints (11) guarantee that i can only be a destination of s if there is a controller placed there.

Constraints (12) guarantee that a link $\{i, j\}$ can only be upgraded if it belongs to the routing path of a switch to its controller, i.e., if at least one routing path uses one of the arcs of $\{i, j\}$. Constraints (13) account for both arcs in a link, meaning that if one arc is upgraded, then the arc in the opposite direction is upgraded too.

Constraints (14)-(16) are the linearization constraints guaranteeing that $w_{ij}^s = x_{ij}^s \cdot z_{ij}$. These variables w_{ij}^s are auxiliary variables used to define constraints (17), which guarantee that the control paths have availability values of at least λ. Constraints (17) result from the linearization of the availability constraints, as will be thoroughly discussed in subsection B. The constraints correspond to exact linearized versions of the availability constraints and not the usual approximation expressions.

Finally, constraints (18)-(21) are the variable domain constraints.

2.2 The availability constraints

In this subsection, constraints (17) are discussed. To this end, two linearized expressions are presented for the control path availability, one assuming there is no link availability upgrade, and the other assuming there is link availability upgrade. These expressions are exact and novel (to the best of our knowledge) linearized expressions, that do not result from the usual approximation approach usually employed. In fact, they result from the usual application of logarithms to linearize the availability expressions, and then the corresponding expressions resulted by inspection (taking advantage of the binary nature of the decision variables involved).

First, the linearized expression for the control path availability without upgrade, is presented.

Proposition 1. *The linearized expression for the control path availability of any switch s, without link availability upgrade, is given by*

$$\mathcal{L}_s^0 = \sum_{(i,j)\in A} x_{ij}^s \log(a_{ij}) \tag{22}$$

Proof. The availability of the control path of switch s is given by the product of the availability of each link of the path. Assuming there are no upgraded links, the availability of the control path of switch s is given by

$$\mathcal{A}_s^0 = \prod_{\substack{(i,j)\in A:\\ x_{ij}^s=1}} a_{ij} = \prod_{(i,j)\in A} \left[1 - x_{ij}^s(1-a_{ij})\right] \tag{23}$$

The second equality of (23) results from the definition of variables x_{ij}^s, i.e., x_{ij}^s is a binary variable that indicates if arc (i,j) belongs to the control path of s.

The linearization of (23) is obtained by applying the logarithm, which is given by

$$\log(\mathcal{A}_s^0) = \sum_{(i,j)\in A} \log\left[1 - x_{ij}^s(1-a_{ij})\right] \tag{24}$$

To show that expressions (22) and (24) are equivalent, i.e., that $\mathcal{L}_s^0 = \log(\mathcal{A}_s^0)$, recall that variables x_{ij}^s are binary. Hence,

- if $x_{ij}^s = 0$, then $\log\left[1 - x_{ij}^s(1-a_{ij})\right] = \log(1) = 0$ and $x_{ij}^s \log(a_{ij}) = 0$
- if $x_{ij}^s = 1$, then $\log\left[1 - x_{ij}^s(1-a_{ij})\right] = \log(a_{ij})$ and $x_{ij}^s \log(a_{ij}) = \log(a_{ij})$

Therefore, $\mathcal{L}_s^0$ and $\log(\mathcal{A}_s^0)$ are equivalent. □

The linearized expression for the control path availability, assuming link availability upgrade, is now presented.

Proposition 2. *The linearized expression for the control path availability of any switch s,* *with link availability upgrade, is given by*

$$\mathcal{L}_s = \sum_{(i,j)\in A} \left[x_{ij}^s \log(a_{ij}) + w_{ij}^s \left(\log(\hat{a}_{ij}) - \log(a_{ij})\right)\right] \tag{25}$$

Proof. Assuming links can be upgraded, the availability of the control path of switch s, is now given by

$$\mathcal{A}_s = \prod_{\substack{(i,j)\in A:\\ x_{ij}^s=1}} [a_{ij} + w_{ij}^s(\hat{a}_{ij} - a_{ij})] \tag{26}$$

Since w_{ij}^s is also a binary variable indicating that arc (i,j) belongs to the control path of s with upgraded availability, following the same line of thought as for $\mathcal{A}_s^0$,

$$\mathcal{A}_s = \prod_{(i,j)\in A} [1 - x_{ij}^s(1-a_{ij}) - w_{ij}^s(a_{ij} - \hat{a}_{ij})] \tag{27}$$

The linearization of (27) is obtained by applying the logarithm, which is given by

$$\log(\mathcal{A}_s) = \sum_{(i,j)\in A} \log\left[1 - x_{ij}^s(1-a_{ij}) - w_{ij}^s(a_{ij} - \hat{a}_{ij})\right] \tag{28}$$

To show that expressions (25) and (28) are equivalent, i.e., that $\mathcal{L}_s = \log(\mathcal{A}_s)$, recall that variables x_{ij}^s and w_{ij}^s are binary. Hence, by taking $\gamma = 1 - x_{ij}^s(1-a_{ij}) - w_{ij}^s(a_{ij} - \hat{a}_{ij})$,

- if $x_{ij}^s = 0$, then $w_{ij}^s = 0$ resulting in $\gamma = \log(1) = 0$ and
  $x_{ij}^s \log(a_{ij}) + w_{ij}^s [\log(\hat{a}_{ij}) - \log(a_{ij})] = 0$
- if $x_{ij}^s = 1$ and $w_{ij}^s = 0$, then $\gamma = \log(a_{ij})$ and
  $x_{ij}^s \log(a_{ij}) + w_{ij}^s [\log(\hat{a}_{ij}) - \log(a_{ij})] = \log(a_{ij})$
- if $x_{ij}^s = 1$ and $w_{ij}^s = 1$, then $\gamma = \log(\hat{a}_{ij})$ and
  $x_{ij}^s \log(a_{ij}) + w_{ij}^s [\log(\hat{a}_{ij}) - \log(a_{ij})] = \log(\hat{a}_{ij})$

Therefore, $\mathcal{L}_s$ and $\log(\mathcal{A}_s)$ are equivalent. □

Imposing the control path availability requirements, results in the constraints given by $\mathcal{A}_s \geq \lambda$. The linearized version of these constraints is given by $\log(\mathcal{A}_s) \geq \log(\lambda)$, which as a consequence of Proposition 2 results in constraints (17).

Since $a_0, a_1, \lambda < 1$, then $\log(a_0), \log(a_1), \log(\lambda) < 0$ hence (17) can be rewritten as

$$\sum_{\{i,j\} \in A} \left[-\log(a_0) x_{ij}^s + \left(\log(a_0) - \log(a_1)\right) w_{ij}^s \right] \leq -\log(\lambda) \tag{29}$$

to yield positive values.

3 Solution Methods

In this section, an exact method and a heuristic method are proposed for solving the ALU-CPP.

The exact method involves solving two ILP problems. The main ILP model is the ALU-CPP, i.e., the ILP model given by (4)-(21). However, the number of controllers C used in the exact method is fixed, and it is the optimal value for the standard CPP which only considers the delay constraints. This CPP is referred to as delay-CPP herein and is given by

$$\begin{aligned} &\min \textstyle\sum_{i \in N} y_i \\ &\text{s.t.} \\ &\quad (6) - (7) \end{aligned} \tag{30}$$

Since the control plane performance strongly depends on the SC and CC delays, minimizing the number of controllers aims to minimize the CC delay communication overhead. Deploying more controllers tends to increase the average CC delay, although it does tend to decrease SC delays. However, due to the maximum CC delay allowed, D_{cc}, it may not be possible to deploy many more controllers.

The exact method consists in two steps as follows:

(i) first, the delay-CPP is solved, which provides the optimal number C of controllers to be placed in the network given the delay constraints;

(ii) then, with the optimal value C, the ALU-CPP is solved, which provides an optimal controller placement and an optimal set of links to have upgraded availability.

For medium-sized networks, the exact method can become computationally expensive. Hence, a heuristic is derived which not only considers the optimal value C of the delay-

CPP, but also considers the controller placement solution of the delay-CPP. In this way, variables y_i become fixed, making the ALU-CPP model much easier to solve.

The heuristic consists in two steps as follows:

(i) first, the delay-CPP is solved, which provides the optimal number C of controllers and an optimal controller placement in the network given the delay constraints;

(ii) then, with the optimal value C and fixing variables y_i to reflect the controller placement solution obtained in (i), the modified ALU-CPP is solved, which provides an optimal set of links to have upgraded availability (for that controller placement).

By fixing the controller placement in step (ii), the heuristic produces a set of links to have upgraded availability, that may be suboptimal when considering the joint optimization of the exact method.

To better illustrate the difference between both methods, consider the polska network from SNDlib (12 nodes and 18 links). As in [14], the delay requirements D_{sc} and D_{cc} are given as percentages of the graph diameter (i.e., longest shortest path between any two nodes of the network).

By solving the delay-CPP for $D_{sc} = 35\%$ and $D_{cc} = 65\%$, the optimal number of controllers provided is $C = 3$. The 3 controller placements obtained are shown as red squares in Fig. 1(a). The heuristic method considers this placement in the ALU-CPP. Then considering the availability requirement given by $\lambda = 0.999$, the heuristic provides a set of 4 links to have upgraded availability, which are shown as red and thick lines in Fig. 1(a).

By allowing the ALU-CPP to also optimize the controller placement, the exact method provides a solution with 3 controllers, shown as red squares in Fig. 1(b). By shifting a controller to a neighbor node (the bottom controller), the exact method now provides a set of 3 links (shown as red lines) to have upgraded availability (instead of 4 links). Note that the controller placement in Fig. 1(b) is also an optimal solution for the delay-CPP, whereas the controller placement in Fig. 1(a) does not yield the optimal solution for the ALU-CPP.

4 Computational Results

In this section, computational results are presented for the exact and heuristic methods. Different thresholds for D_{sc} and D_{cc} were chosen to assess the sensitivity of the ALU-CPP

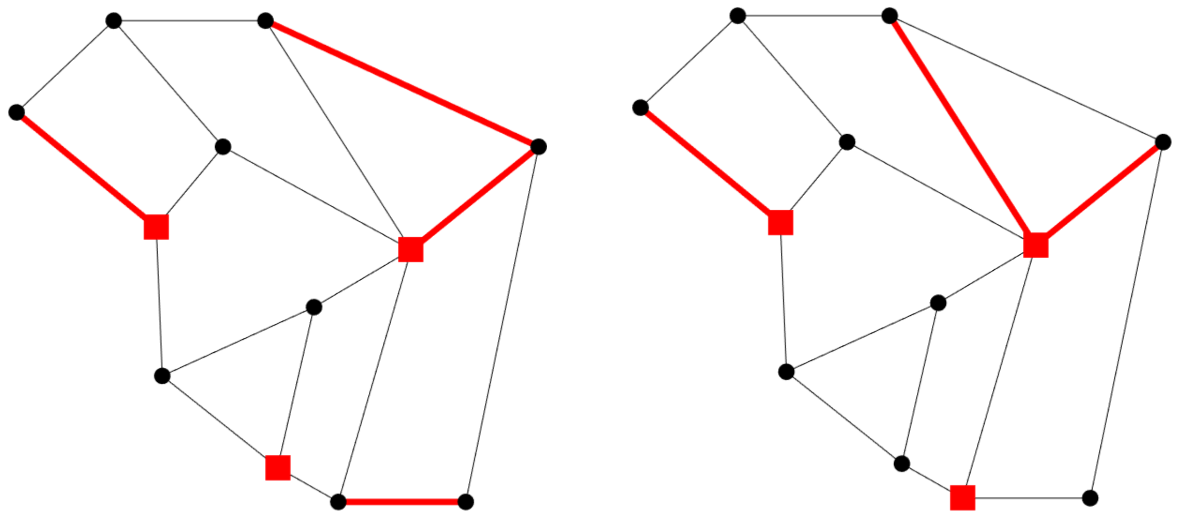
(a) Heuristic solution. (b) Exact solution.

Figure 1: Results for polska with $D_{sc} = 35\%$, $D_{cc} = 65\%$ and $\lambda = 0.999$. The controller placements are shown as red squares, while the upgraded links are shown as red lines.

Table 1: Characteristics of the networks

Network	#nodes	#links	avg deg	diameter [km]
polska	12	18	3.00	811
nobel_germany	17	26	3.06	790
janos_us	26	42	3.23	4690
cost266	37	57	3.08	4032

to these parameters. As mentioned above, these parameters are considered as percentages of the graph diameter.

Four networks were chosen from SNDlib: polska, nobel_germany, janos_us and cost266. The characteristics of these networks are summarized in Table 1, which shows the number of nodes, the number of links, the average node degree and the graph diameter (in km) for each network.

As in [14], the link lengths were calculated by considering the node coordinates and computing the shortest path between the nodes over the Earth's surface. Then, the delay between two nodes d_{ij} is given by the shortest path length between them.

Both methods were implemented in C++, using CPLEX 12.8 callable libraries to solve the ILP models. The instances were run on a 8 core Intel Core i7 PC with 64 GB of RAM, running at 3.6 GHz.

The computational results are shown in Table 2 for polska, Table 3 for nobel_germany,

Table 2: Computational results for polska network

Parameters			C	Exact		Heuristic	
λ	D_{sc}	D_{cc}		#links	t(s)	#links	t(s)
0.999	35%	65%	3	3	0.152	4	0.212
		70%	3	3	0.175	4	0.198
		75%	3	3	0.342	3	0.132
	40%	65%	3	3	0.339	4	0.188
		70%	3	3	0.524	4	0.292
		75%	3	3	0.505	4	0.301
	45%	65%	3	3	0.615	4	0.193
		70%	2	6	0.706	6	0.542
		75%	2	6	0.767	6	0.515

Table 4 for janos_us and Table 5 for cost266. The parameters are shown in the first three columns of each table. Note that the SC delay parameter D_{sc} was chosen to range between 35% and 45% (in steps of 5%) for the smaller network polska, while for the larger networks D_{sc} was chosen to range from 30% to 40% (also in steps of 5%; $D_{sc} = 30\%$ is infeasible for polska). Following the same reasoning, the CC delay parameter D_{cc} was chosen to range from 65% to 75% (in steps of 5%) for polska, while for the larger networks D_{cc} was chosen to range from 60% to 70% (also in steps of 5%; $D_{cc} = 60\%$ for $D_{sc} = 35\%$ is infeasible for polska).

Moreover, the availability parameter was chosen to be $\lambda = 0.999$ for the smaller networks polska and nobel_germany (Tables 2 and 3). For the larger networks janos_us and cost266 (Tables 4 and 5), two values for λ were chosen, 0.995 and 0.9975, to show that a small increase of the availability requirement, profoundly affects the performance of the optimization methods.

The delay-CPP is the first ILP model to be solved in both methods. It provides the optimal value C for the ALU-CPP, which is shown in the fourth column of each table. The ALU-CPP model is then solved for the exact and heuristic methods. Recall that while the heuristic method considers the number and controller placements given by the delay-CPP, the exact method only considers the number of controllers and their placement is part of the problem to be solved.

For both methods, the number of links (#links) and the runtimes in seconds (t(s)) are shown, for each instance (last four columns of each table).

Table 3: Computational results for nobel_germany network

Parameters			C	Exact		Heuristic	
λ	D_{sc}	D_{cc}		#links	t(s)	#links	t(s)
0.999	30%	60%	4	2	0.387	3	0.374
		65%	4	2	0.356	3	0.401
		70%	4	1	0.255	3	0.398
	35%	60%	2	5	1.058	5	0.658
		65%	2	5	0.957	5	0.699
		70%	2	5	5.971	5	0.643
	40%	60%	2	5	3.540	5	0.640
		65%	2	5	1.329	5	0.634
		70%	2	5	4.439	5	0.631

Table 4: Computational results for janos_us network

Parameters			C	Exact		Heuristic	
λ	D_{sc}	D_{cc}		#links	t(s)	#links	t(s)
0.995	30%	60%	4	6	352.938	8	214.069
		65%	4	6	291.159	7	4.939
		70%	3	7	67.004	7	22.883
	35%	60%	3	7	104.270	8	90.307
		65%	3	7	299.155	8	90.758
		70%	3	7	191.918	9	181.775
	40%	60%	2	9	191.789	9	134.720
		65%	2	9	194.086	9	134.655
		70%	2	9	222.581	9	134.761
0.9975	30%	60%	4	12	497.095	13	426.130
		65%	4	12	1095.433	12	435.273
		70%	3	13	147.529	13	357.724
	35%	60%	3	13	1891.822	14	588.457
		65%	3	13	1879.174	14	587.115
		70%	3	*	*	14	437.410
	40%	60%	2	15	748.325	15	832.167
		65%	2	*	*	15	827.114
		70%	2	15	3885.988	15	828.747

Table 5: Computational results for cost266 network

Parameters			C	Exact		Heuristic	
λ	D_{sc}	D_{cc}		#links	t(s)	#links	t(s)
0.995	30%	60%	4	5	3777.748	6	289.292
		65%	4	4	820.786	6	288.287
		70%	4	4	365.941	6	289.871
	35%	60%	3	*	*	7	209.102
		65%	3	7	2630.873	7	209.236
		70%	3	*	*	7	209.539
	40%	60%	2	8	308.066	9	180.649
		65%	2	8	274.025	9	179.370
		70%	2	8	245.136	9	179.573
0.9975	30%	60%	4	*	*	15	9783.879
		65%	4	*	*	15	9781.396
		70%	4	*	*	15	9808.842
	35%	60%	3	13	7958.082	15	2636.273
		65%	3	13	7333.177	15	2639.982
		70%	3	*	*	15	2640.573
	40%	60	2	*	*	17	15848.096
		65	2	*	*	17	15845.572
		70	2	*	*	17	15837.884

Table 2 shows that for the smallest network, polska, the exact method is very efficient (with runtimes under 1 second). The heuristic is just as fast and very competitive. In fact, for 3 out of 9 instances, the heuristic method provided an optimal solution (number of links is the same as for the exact method), whereas for the remaining instances it provided a solution with a surplus of one link to have upgraded availability.

Table 3 shows that for nobel_germany, the exact method is still very efficient (with the longest runtime under 6 seconds). Nevertheless, the heuristic method yields runtimes under 1 second, for all instances. The heuristic is very competitive, where for most instances, it provided an optimal solution, whereas for 2 instances it provided a solution with a surplus of 1 link, and for 1 instance it provided a solution with a surplus of 2 links to have upgraded availability.

Table 4 shows results for janos_us, with $\lambda = 0.995$ and $\lambda = 0.9975$. For $\lambda = 0.995$, the exact method is able to provide optimal solutions, for all instances, with the longest runtime under 6 minutes. The heuristic method yields smaller runtimes for all instances, with the longest runtime under 4 minutes. In fact, for $D_{sc} = 30\%$ and $D_{cc} = 65\%$, the runtime of the exact method was almost 300 seconds, while that of the heuristic method was under 5 seconds. Also for $D_{cc} = 70\%$, the runtime of the exact method was about 67 seconds, while that of the heuristic was 22 seconds. Moreover, the heuristic is competitive, where for 4 out of 9 instances, it provided an optimal solution, whereas for 3 instances it provided a solution with a surplus of 1 link, and for 2 instances it provided a solution with a surplus of 2 links to have upgraded availability.

By increasing λ to 0.9975, the exact method becomes computationally challenging, having terminated due to memory issues, in two instances (marked with *), without providing an optimal solution. The heuristic method is able to provide solutions for all instances, with the longer runtimes being around 800 seconds (see Table 4). Although, there are two instances where the exact method requires a smaller runtime than the heuristic method, the latter remains very competitive. For 4 out of 7 instances, the heuristic provided an optimal solution, whereas for the remaining 3 instances it provided a solution with a surplus of 1 link to have upgraded availability.

Table 5 shows the results for the largest network, cost266, with $\lambda = 0.995$ and $\lambda = 0.9975$. For $\lambda = 0.995$, the exact method becomes computationally challenging, having terminated due to memory issues, in two instances (marked with *), without providing an optimal solution. The heuristic method is able to provide solutions, for all instances, in under 5 minutes. The heuristic is competitive, where for 1 out of 7 instances, the heuristic provided an optimal solution, whereas for 4 instances it provided a solution with a surplus of 1 link, and for 2 instances it provided a solution with a surplus of 2 links to have upgraded availability.

By increasing λ to 0.9975, the exact method is only able to provide solutions in two instances, having terminated due to memory issues in the remaining ones. Clearly, the exact method is not feasible for such instances. The heurisitc method is able to find solutions for all instances, although for $D_{sc} = 40\%$, the runtimes soar to over 4 hours.

5 Trade-Off Between Number of Controllers and Availability Link Upgrade

The exact method, presented in Section 2, is equivalent to lexicographically minimizing (C, #links), i.e., first minimizing the number of controllers C, and then minimizing the number of upgraded links. This is motivated by the fact, that the CC communication delay strongly affects the control plane performance. Deploying more controllers tends to increase the average CC delay, although it may decrease the number of links that need upgraded availability. One may question if increasing the number of controllers could lead to avoiding link availability upgrade altogether.

Therefore, the trade-off between increasing the number of controllers versus the number of upgraded links was also studied. To do this, the exact method was solved for incremental values of C, from the optimal value by the delay-CPP, and stopped either when (i) the number of upgraded links is zero; (ii) or when the problem is infeasible. In the latter case, the maximum CC delay requirement D_{cc} imposes a limit on the number of controllers to be deployed. From the set of these solutions, the set of non-dominated solutions were chosen.

Consider two solutions, one with C_1 controllers and L_1 links to have upgraded availability, and another with C_2 controllers and L_2 links to have upgraded availability. The first solution dominates the second if $C_1 < C_2$ and $L_1 \leq L_2$ or if $C_1 = C_2$ and $L_1 < L_2$. To find the set of non-dominated solutions, any solution that is dominated by another in the set is removed.

This study was conducted for the cases in the previous section, where the exact method managed to find the optimal solutions for all instances. Therefore, the study was conducted for the polska and nobel_germany networks (with $\lambda = 0.999$), and for janos_us network with $\lambda = 0.995$. In Tables 6, 7 and 8, the solutions for incremental values of C are shown, for each instance of the polska, nobel_germany and janos_us networks respectively. The non-dominated solutions are marked with an *. The last line is either for the non-dominated solution with zero upgraded links or is indicated with a '–' in case the problem is infeasible for the corresponding value of C.

In Table 6, it is possible to see that for this small network, incrementally increasing the number of controllers does tend to decrease the number of links to have upgraded availability. The price to pay to only upgrade the availability on one link, is to deploy almost half of the nodes with controllers – which can be arguable from the perspective of SDN [5]. Moreover, the D_{cc} requirement makes it impossible to deploy sufficient controllers to avoid availability link upgrade altogether.

In Table 7, the conclusions for nobel_germany are similar for $D_{cc} = 60\%$ and $D_{cc} = 65\%$. However, for the highest value of $D_{cc} = 70\%$, it is possible to discard link upgrade, at the cost of deploying 5 controllers (deploying controllers in almost 1/3 of the nodes).

In Table 8, it is possible to see that for the larger network janos_us, the D_{cc} requirement

Table 6: Set of solutions for polska network with $\lambda = 0.999$; non-dominated solutions marked with *

$D_{sc} = 35\%$						$D_{sc} = 40\%$						$D_{sc} = 45\%$					
$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 75\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 75\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 75\%$	
C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links
3	3*	3	3*	3	3*	3	3*	3	3*	3	3*	3	3*	2	6*	2	6*
4	2*	4	2*	4	2*	4	2*	4	2*	4	2*	4	2*	3	3*	3	3*
5-8	2	5	1*	5	1*	5-8	2	5	1*	5	1*	5	1*	4	2*	4	2*
9	–	6-8	1	6-9	1	9	–	6-8	1	6-9	1	6-8	1	5	1*	5	1*
		9	–	10	–			9	–	10	–	9	–	6-8	1	6-9	1
														9	–	10	–

Table 7: Set of solutions for nobel_germany network with $\lambda = 0.999$; non-dominated solutions marked with *

$D_{sc} = 30\%$						$D_{sc} = 35\%$						$D_{sc} = 40\%$					
$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$	
C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links
4	2*	4	2*	4	1*	2	5*	2	5*	2	5*	2	5*	2	5*	2	5*
5	1*	5	1*	5	0*	3	3*	3	3*	3	3*	3	3*	3	3*	3	3*
6-11	1	6-11	1			4	2*	4	2*	4	1*	4	2*	4	2*	4	1*
12	–	12	–			5	1*	5	1*	5	0*	5	1*	5	1*	5	0*
						6-11	1	6-11	1			6-11	1	6-11	1		
						12	–	12	–			12	–	12	–		

Table 8: Set of solutions for janos_us network with $\lambda = 0.995$; non-dominated solutions marked with *

$D_{sc} = 30\%$						$D_{sc} = 35\%$						$D_{sc} = 40\%$					
$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$		$D_{cc} = 60\%$		$D_{cc} = 65\%$		$D_{cc} = 70\%$	
C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links	C	#links
4	6*	4	6*	3	7*	3	7*	3	7*	3	7*	2	9*	2	9*	2	9*
5	5*	5	5*	4	5*	4	6*	4	6*	4	5*	3	6*	3	6*	3	6*
6	4*	6	4*	5	4*	5	5*	5	5*	5	4*	4	5*	4	5*	4	5*
7-15	4	7-16	4	6	3*	6	4*	6	4*	6	3*	5	4*	5	4*	5	4*
16	–	17	–	7-18	3	7-15	4	7-16	4	7-18	3	6-17	4	6	3*	6	3*
				19	–	16	–	17	–	19	–	18	–	7	2*	7	2*
														8-20	2	8-20	2
														21	–	21	–

forces the minimum number of upgraded links to be equal or greater to 2. This number is much larger when considering $\lambda = 0.9975$, since the availability constraint is higher (results not shown since not all instances could be determined, as shown in Table 4). Moreover, it is possible to observe that the more controllers are deployed the more difficult it becomes to decrease the number of links. This can be observed, for example, in Table 8 for $D_{sc} = 40\%$ and $Dcc = 60\%$. Increasing the number of controllers from 2 to 3 results in a decrease of 3 upgraded links, whereas increasing the number of controllers from 3 to 4 and then from 4 to 5, results in each case in a decrease of 1 upgraded link. However, from 5 controllers we cannot reduce the number of upgraded links any further (the problem becomes infeasible for 18 controllers).

In summary, the tables show the trade-off between the deployed number of controllers C and the necessary number of links to have upgraded availability. Incrementally increasing the number of controllers eventually leads to a decrease in the number of upgraded links. However for reasonable delay bounds, it is often not possible to avoid availability link upgrade.

6 Conclusions

In SDN networks, the CPP is usually addressed focusing on delay requirements. In this paper, a variant of the CPP is considered, the ALU-CPP, which also considers control path availability requirements. A novel way of expressing the the exact linearized availability constraints is presented and discussed.

An exact method and a derived heuristic method are proposed. The number C of controllers used in both methods is provided by the delay-CPP model (the model only considering delay requirements). The heuristic method also considers the controller placement provided by the delay-CPP. The exact method results in a lexicographical minimization of the number of controllers and then of the number of links to have upgraded availability. The order of the minimization objectives reflects the importance of the CC delays in control plane communication.

Computational results were conducted for two smaller networks and two larger networks. The results for the smaller networks, show that the exact method is computationally efficient for small-sized networks. The results for the larger networks were conducted considering two values of λ. These results show that a small variation in the availability requirement λ, profoundly affects the performance of both methods. The exact method becomes computationally challenging, yielding it impractical for medium-sized networks. The heuristic, however, shows to be very competitive (although runtimes can soar), yielding solutions with a surplus of up to 2 links to have upgraded availability. Hence, the heuristic is a promising compromise to solve the ALU-CPP.

Moreover, a study of the trade-off between the number of controllers and the number of links with upgraded availability to achieve a given λ availability for the control paths, was also conducted. This study shows that incrementally increasing the number of controllers eventually leads to a decrease in the number of upgraded links. However, for the CC delay requirement imposes a limit on the number of controllers to be deployed, making it impossible to avoid link upgrade in most of the cases.

Further research is needed in this problem, and future work entails availability requirements also considering backup paths between the switches and their controllers (or backup paths to backup controllers).

Acknowledgment

This work was partially supported by the COST Action CA15127 ("Resilient communication services protecting end user applications from disaster-based failures – RECODIS").

References

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

[2] A. Alashaikh, D. Tipper, and T. Gomes. Supporting differentiated resilience classes in multilayer networks. In *DRCN*, pages 31–38, Paris, France, 2016.

[3] A. de Sousa, T. Gomes, R. Girão-Silva, and L. Martins. Minimizing the network availability upgrade cost with geodiversity guarantees. In *RNDM*, pages 4–6, Alghero, Italy, 2017.

[4] R. Girão-Silva, L. Martins, T. Gomes, A. Alashaikh, and D. Tipper. Improving network availability - a design perspective. In *ICICT*, pages 799–815, London, United Kingdom, 2018.

[5] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *ACM HotSDN*, pages 7–12, New York, USA, 2012.

[6] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia. Pareto-optimal resilient controller placement in sdn-based core networks. In *ITC*, pages 1–9, Shanghai, China, 2013.

[7] Y. Hu, W. Wendon, X. Gong, X. Que, and C. Shiduan. Reliability-aware controller placement for software-defined networks. In *IM*, pages 672–675, Ghent, Belgium, 2013.

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

[9] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, and A. Kamisinski. Availability modelling of software-defined backbone networks. In *DSN-W*, pages 105–112, Toulouse, France, 2016.

[10] G. Nencioni, B. E. Helvik, and P. E. Heegaard. Including failure correlation in availability modeling of a software-defined backbone network. *IEEE Transactions on Network and Service Management*, 14(4):1032–1045, 2017.

[11] T. A. Nguyen, T. Eom, S. An, J. S. Park, J. B. Hong, and D. S. Kim. Availability modeling and analysis for software defined networks. In *PRDC*, pages 159–168, Zhangjiajie, China, 2015.

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

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

[14] D. Santos, A. de Sousa, and C. Mas Machuca. Combined control and data plane robustness of SDN networks against malicious node attacks. In *CNSM 2018*, pages 54–62, Rome, Italy, 2018.

[15] 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*, 14(2):302–316, 2017.

[16] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery – Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[17] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, and M. Jaeger. General availability model for multilayer transport networks. In *DRCN*, pages 1–8, Ischia, Italy, 2005.

[18] P. Vizarreta, C. M. Machuca, and W. Kellerer. Controller placement strategies for a resilient SDN control plane. In *RNDM*, pages 253–259, Halmstad, Sweden, 2016.