This is the accepted version of the following article: **Dorabella Santos, Teresa Gomes, and David Tipper. Software-Defined Network Design driven by Availability Requirements.** ***2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020*****, Milano, Italy, 25-27 Mar. 2020, pp. 1-7. DOI: 10.1109/DRCN48652.2020.1570604282**, which has been published in final form at `https://ieeexplore.ieee.org/document/9089377`.

Software-Defined Network Design driven by Availability Requirements*

Dorabella Santos† Teresa Gomes‡† David Tipper§

dorabella.santos@gmail.com teresa@deec.uc.pt dtipper@pitt.edu

January 2020

Abstract

In Software Defined Networking (SDN), the controller locations are mainly constrained by delays between switches and controllers, and between the controllers themselves. In addition to the delay requirements, the availability of the connections between switches and controllers is also a key issue for control plane performance. Here, we explore the idea of having a spanning tree substructure called the *spine*,

*This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under project grant UIDB/00308/2020 and was financially supported by FEDER 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

§Department of Informatics and Networked Systems, University of Pittsburgh, Pittsburgh, PA 15260

whose links can be upgraded to have high availability in order to support availability requirements for the control paths (routing paths between switches and controllers). We formulate an optimization model of the joint controller placement and spine design problem for SDN networks, under delay, availability and path redundancy requirements.Numerical results are presented showing the trade-offs between the number of controllers, delay requirements and upgrade cost.
Keywords: SDN, controller placement, availability, spine design, optimization

1 Introduction

In Software Defined Networking (SDN), the controller locations are mainly constrained by two types of delays: (i) the delays between each controller and the set of switches they manage; (ii) the delays between the controllers themselves. The problem of how many controllers to deploy in a SDN network and where to deploy them, is known as the Controller Placement Problem (CPP).

In a logically centralized control plane, the set of controllers function as a unit. While delays between controllers can influence control plane communication overhead, the communication between each controller and their switches is much more critical and frequent. Moreover, the question of the availability of the connections between each controller and their switches is also important. The availability of the control paths (routing paths connecting the switches to their controllers) is usually assessed in terms of control plane connectivity [7,8] or reachability [9]. Traditionally, end-to-end availability is increased by path redundancy, i.e., by guaranteeing a link/node-disjoint backup path between the respective end nodes. In [16], although controller placement is not addressed, path redundancy is considered as a means to increase control path availability.

There are also works addressing the CPP with availability requirements. In [12], the CPP is studied, where each switch connects to a primary and to one or more backup controllers, in order to guarantee availability requirements for the control paths. In [6], the control path availability is assessed in terms of the expected percentage of control path loss given the failure probability of each network component. Several heuristics are proposed for the controller placement problem, aiming to maximize the availability of the control paths. The previous works do not jointly consider delay constraints for the controller placements and availability requirements for the control paths. Moreover, the desired availability for the control paths cannot always be achieved by path redundancy alone. In [1], the concept of a spine is proposed, where a higher availability subgraph exists in the network.

In this work, we assume that the spine is a spanning tree, and that its links can be upgraded to have increased availability at a given cost [2], by reducing the average time to repair of the link – and/or reducing the time between failures (by installing more robust equipment on the link, for example). Our previous work [15] also addresses the problem of controller placement and link upgrade. However, path redundancy was not considered and the upgraded links did not have to be on a spanning tree subgraph.

The contribution of this paper is as follows: (i) we address the problem of determining the controller placements and selecting a spanning tree to be the spine, i.e., an higher available subgraph in the network, such that the cost of upgrading the spine is minimized, while satisfying delay, path redundancy and availability requirements; (ii) we present an exact way of linearizing the inherent availability constraints; (iii) we formulate the problem as a compact integer linear programming (ILP) model.

The paper is organized as follows: in Section 2 we describe the optimization problem of joint controller placement and spanning tree selection, such that the cost of upgrading the links on the spanning tree to achieve the desired levels of control path availability is minimized; in Section 3 we present the optimization problem as an integer linear programming (ILP) model and we detail an exact way of linearizing the availability constraints therein; in Section 4 we discuss the computational results for our ILP model; and in Section 5 we present our conclusions.

2 Joint CPP and Spine Design Problem

The problem addressed in this paper, is the problem of selecting the controller locations and a spanning tree to be the spine, such that the upgrade cost of the spine is minimized, while satisfying delay and availability requirements.

More precisely, a given number C of controllers must be deployed in the network, such that: (i) the delay between a switch and its controller – switch-controller (SC) delay – cannot exceed a given maximum value D_{sc}; (ii) the delay between any two controllers – controller-controller (CC) delay – cannot exceed a given maximum value D_{cc}. These maximum values guarantee acceptable control plane performance, in a logically centralized control plane, where besides the switch-controller communication, intercontroller communication is also present for synchronization. Since the SC communications are more frequent than the CC ones, we assume that $D_{sc} < D_{cc}$ [11]. The minimum value C can be obtained by solving the CPP problem with respect to the delay requirements [14].

Moreover, we consider path redundancy to protect the control paths (routing paths between the switches and their controllers) against single link and node failures. Therefore, a pair of node-disjoint paths is guaranteed between each switch and its controller [18]. A minimum end-to-end availability λ between each switch and its controller is also required (given by the pair of node-disjoint routing paths).

The end-to-end availability $\mathcal{A}$ of the node-disjoint pair of paths is given by $\mathcal{A} = 1 - (1 - \mathcal{A}_p)(1 - \mathcal{A}_b)$, where $\mathcal{A}_p$ and $\mathcal{A}_b$ denote the availabilities of the primary path and of the backup path, respectively. To ensure the minimum availability requirement, we further consider the primary path must satisfy an availability minimum of λ_p, whereas the backup path must satisfy an availability minimum of λ_b, such that $1 - (1 - \lambda_p)(1 - \lambda_b) \geq \lambda$ [2].

To achieve the required availabilities, we assume that the links of the network can be upgraded to have enhanced availability, and we impose that the upgraded links sit on a spanning tree subgraph. However, we do not force any of the paths to be routed on the spanning tree – the paths will use the spanning tree links if this is necessary to achieve the required availability.

Consider that the SDN data plane is represented by an undirected 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 $\{i, j\}$, with $i \neq j$. We consider that each link of the network has a default distance-based availability given as in [17] (page 186)

$$\alpha_{ij}^0 = 1 - \frac{MTTR}{MTBF_{ij}} \tag{1}$$

where $MTTR$ denotes the mean time to repair (in hours) and is considered to be 24 h. Parameter $MTBF_{ij}$ denotes the mean time between failures (in hours) of link $\{i, j\} \in E$ and is given by $MTBF_{ij} = CC \times 365 \times 24/\ell_{ij}$, where CC denotes the cable cut rate and is considered to be 450 km and ℓ_{ij} denotes the link length.

Given a spanning tree to be the spine, each link of the spine can be upgraded, so that corresponding unavailability is decreased by a given value $\varepsilon \in (0, 1)$. Considering κ levels of link availability upgrade, we have that the unavailability of link $\{i, j\}$ upgraded to level k is denoted by μ_{ij}^k and given by [2] (we do not consider any downgrade level here) $\mu_{ij}^k = (1 - \varepsilon)\mu_{ij}^{k-1}$, $k = 1, ..., \kappa$ where $\mu_{ij}^0 = 1 - \alpha_{ij}^0$ denotes the default unavailability. In terms of availability, we have that $\mu_{ij}^k = 1 - \alpha_{ij}^k$ and so $\alpha_{ij}^k = \alpha_{ij}^{k-1} + \varepsilon - \varepsilon\alpha_{ij}^{k-1}$, $k = 1, ..., \kappa$.

The cost for upgrading the link availability to level k is given in [2, 5] as,

$$c_{ij}^k = -\ell_{ij} \cdot \ln\left(\frac{1-\alpha_{ij}^k}{1-\alpha_{ij}^0}\right) \quad k = 1, ..., \kappa \tag{2}$$

Consider Fig. 1 to better illustrate these ideas. The spanning tree selected for nobel_germany network (from SNDlib) is shown as solid lines and the controller nodes are indicated by red circles. In this example, we considered $C = 6$ controllers, $D_{sc} = 35\%$, $D_{cc} = 70\%$, $\lambda_p = 0.999$ and $\lambda_b = 0.90$.

The thickness of the lines are proportional to the level of availability upgrade of the link of the tree (we have used $\kappa = 3$ levels). The thin solid lines means that those links did not need to be upgraded, the thicker solid lines means the links were ugraded to level $k = 1$ and the thickest solid lines means that they were upgraded to level $k = 2$. There was no need to upgrade to level $k = 3$.

The upgrading of these links depends on the end-to-end availability requirements between each switch and its controller. The primary and backup paths for switch nodes 2 and 3 are shown. The primary paths are shown as red dashed lines, while the backup paths are shown as blue dashed lines with '+' markers. Note that switch node 2 is managed by controller node 10, while switch node 3 is managed by controller node 1.

It should be pointed out that for switch node 3, the direct link to its controller in node 1 is not on the spanning tree. The primary path is routed on a two-link path, one of which is on the spanning tree. The link on the spanning tree has to be upgraded, to ensure $\lambda_p = 0.999$ is achieved. Note that the backup path, which corresponds to the direct link, not selected to be on the spine, does not ensure λ_p but satisfies $\lambda_b = 0.90$.

For switch node 2, the primary path traverses the shortest path composed of three links, to its controller on node 10. The two links on the spanning tree are upgraded to level $k = 1$, guaranteeing the required λ_p availability.

3 Mathematical Formulation

The optimization problem we aim to solve, is the problem of selecting the controller locations and a set of upgraded links that sit on a spanning tree subgraph of G, such that the cost of upgrading the links in the spanning tree – the spine – is minimized, while satisfying delay and availability requirements. This problem is formulated as an ILP model in the following.

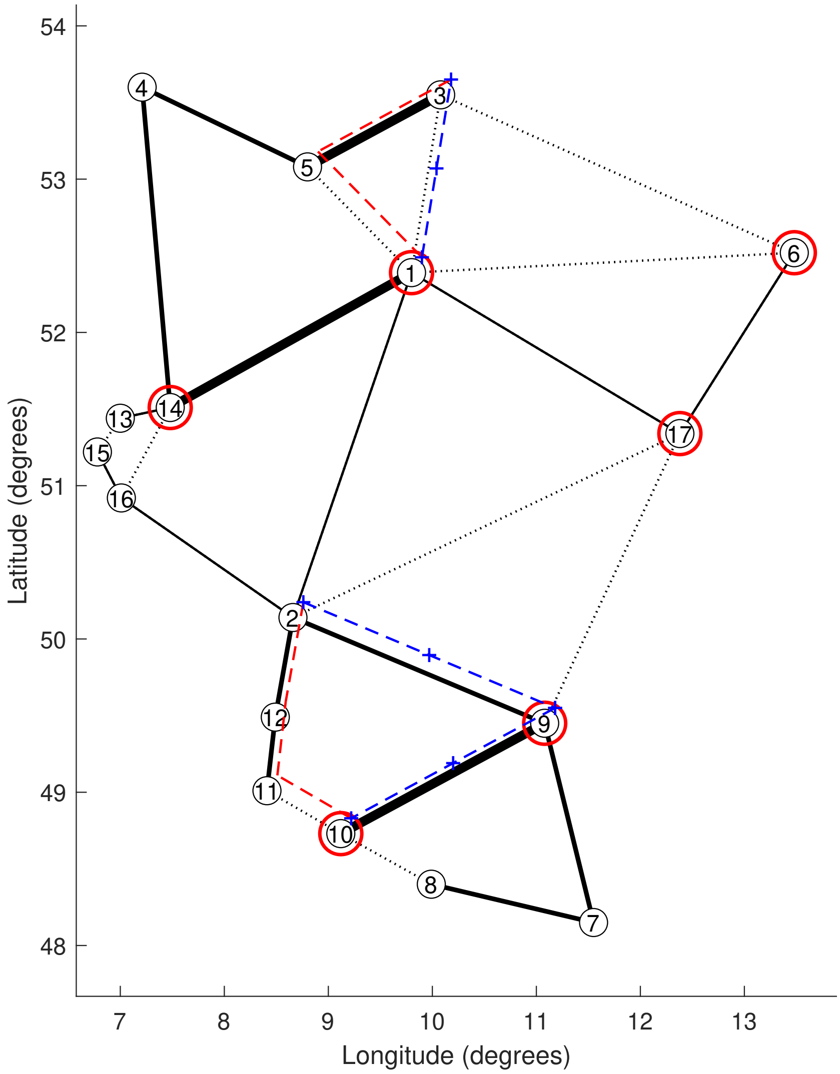

Figure 1: Nobel_germany network with spanning tree shown in solid lines for $C = 6$, $D_{sc} = 35\%$, $D_{cc} = 70\%$ and $\kappa = 3$ levels of availability upgrade; the controller nodes are indicated by the red circles. Primary paths for switch nodes 2 and 3 are shown as red dashed lines; respective backup paths shown as blue dashed lines with '+' markers.

To formulate the ILP model, consider the following notation. Besides the set of links E, consider A to the set of arcs or directed links, i.e., each link $\{i, j\} \in E$ can be represented by the pair of symmetrical arcs (i, j), $(j, i) \in A$. For each node $i \in N$, consider $V(i)$ to be the set of adjacent nodes to node i, i.e., $V(i) = \{j \in N : \{i, j\} \in E\}$. The delay between two nodes i and j is denoted by d_{ij} and given by the shortest distance between them [11].

Consider the following parameters:

ρ arbitrary node referred as the root node to model the spanning tree for the spine;

t_i^s binary parameter that is 1 if $i = s$, and 0 otherwise;

and the following decision variables:

y_i binary variable that is 1 if a controller is placed in node i, and 0 otherwise;

a_i^s binary variable that is 1 if the controller of switch s is placed in node i, and 0 otherwise;

z_{ij}^k binary variable that is 1 if link $\{i, j\}$ is upgraded to level k, and 0 otherwise;

x_{ij}^{s0} binary variable that is 1 if arc $(i, j) \in A$ belongs to the path of switch s to its controller when link $\{i, j\}$ is not upgraded, and 0 otherwise;

x_{ij}^{sk} binary variable that is 1 if arc $(i, j) \in A$ belongs to the path of switch s to its controller when link $\{i, j\}$ is upgraded to level k, and 0 otherwise;

u_{ij}^{s0} binary variable that is 1 if arc $(i, j) \in A$ belongs to the backup path of switch s to its controller when link $\{i, j\}$ is not upgraded, and 0 otherwise;

u_{ij}^{sk} binary variable that is 1 if arc $(i, j) \in A$ belongs to the backup path of switch s to its controller when link $\{i, j\}$ is upgraded to level k, and 0 otherwise;

β_{ij}^{σ} binary variable that is 1 if arc $(i, j \in A)$ is in the path from node σ to the root node ρ, and 0 otherwise;

θ_{ij} binary variable that is 1 if arc (i, j) belongs the spanning tree, and 0 otherwise.

When defining the decision variables, we separate x_{ij}^{s0} from x_{ij}^{sk}, with $k \geq 1$ for better clarification (as with variables u_{ij}^{s0} and u_{ij}^{sk}, with $k \geq 1$). This becomes more relevant in the ILP formulation in Section 3.1, when referring to the link upgrade constraints.

Proposition 1. *The linearized expression related to the availability of the control path of switch s, assuming links can be upgraded up to κ levels, can be expressed as*

$$\mathcal{L}_s = \sum_{(i,j)\in A} \sum_{k=0}^{\kappa} x_{ij}^{sk} \log(\alpha_{ij}^{k}) \tag{3}$$

Proof. The availability of the control path of switch s, i.e., of the routing path from s to its controller, assuming links can be upgraded up to κ levels, is given by

$$\mathcal{A}_s = \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s0}=1}} \alpha_{ij}^{0} \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s1}=1}} \alpha_{ij}^{1} \cdots \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s\kappa}=1}} \alpha_{ij}^{\kappa} \tag{4}$$

By the binary nature of variables x_{ij}^{sk}, it is possible to show that

$$\begin{aligned} \mathcal{A}_s &= \prod_{(i,j)\in A} \prod_{k=0}^{\kappa} \left[x_{ij}^{sk}\alpha_{ij}^{k} + \left(1 - x_{ij}^{sk}\right)\right] \\ &= \prod_{(i,j)\in A} \prod_{k=0}^{\kappa} \left[1 - x_{ij}^{sk}\left(1 - \alpha_{ij}^{k}\right)\right] \end{aligned} \tag{5}$$

Applying logarithms to linearize the expressions, results in

$$\log(\mathcal{A}_s) = \sum_{(i,j)\in A} \sum_{k=0}^{\kappa} \log\left[1 - x_{ij}^{sk}\left(1 - \alpha_{ij}^{k}\right)\right] \tag{6}$$

Due to the binary nature of variables x_{ij}^{sk}, it is possible to show that $\log(\mathcal{A}_s) = \mathcal{L}_s$. In fact, by definition of variables x_{ij}^{sk}, for each $\{i,j\} \in E$, there is one and only one $k_{ij} \in \{1, ..., \kappa\}$ such that $x_{ij}^{sk_{ij}} = 1$. So,

$$\begin{aligned} \log(\mathcal{A}_s) &= \sum_{(i,j)\in A} \log\left[1 - \left(1 - \alpha_{ij}^{sk_{ij}}\right)\right] \\ &= \sum_{(i,j)\in A} \log\left(\alpha_{ij}^{sk_{ij}}\right) = \mathcal{L}_s \end{aligned} \tag{7}$$

□

Note that this proposition is a generalization of Proposition 2 in [15], where more than one level of availability upgrade is considered. Also the definition of the aggregated decision variables x_{ij}^{sk} with $k = 0, ..., \kappa$, allow for a more compact linearized expression when compared to the decision variables used in [15].

3.1 Optimization Model

Given the notation and proposition above the ILP model for the joint SDN spine design and controller placement problem, with path redundancy, is given as follows.

$$\min \sum_{k=1}^{\kappa} \sum_{\{i,j\} \in E} c_{ij}^k z_{ij}^k \tag{8}$$

s.t.

Controller placement constraints:

$$\sum_{i \in N} y_i = C \tag{9}$$

$$\sum_{\substack{j \in N: \\ d_{ij} \leq D_{sc}}} y_j \geq 1 \qquad i \in N \tag{10}$$

$$y_i + y_j \leq 1 \qquad i \in N, j \in N \backslash \{i\} : d_{ij} > D_{cc} \tag{11}$$

Primary path routing constraints:

$$\sum_{k=0}^{\kappa} \sum_{j \in V(i)} \left(x_{ij}^{sk} - x_{ji}^{sk} \right) = t_i^s - a_i^s \qquad s \in N,\ i \in N \tag{12}$$

$$a_s^s = y_s \qquad s \in N \tag{13}$$

$$a_i^s = 0 \qquad s \in N,\ i \in N \backslash \{s\} : d_{is} > D_{sc} \tag{14}$$

$$a_i^s \leq y_i \qquad s \in N,\ i \in N \backslash \{s\} : d_{is} \leq D_{sc} \tag{15}$$

$$\sum_{k=0}^{\kappa} \sum_{(i,j) \in A} d_{ij} x_{ij}^{sk} \leq D_{sc} \qquad s \in N \tag{16}$$

Link upgrade constraints:

$$x_{ij}^{s0} + x_{ji}^{s0} \leq 1 - z_{ij}^k \qquad s \in N,\ \{i,j\} \in E,\ k = 1, ..., \kappa \tag{17}$$

$$x_{ij}^{sk} + x_{ji}^{sk} \leq z_{ij}^k \qquad s \in N,\ \{i,j\} \in E,\ k = 1, ..., \kappa \tag{18}$$

$$\sum_{k=0}^{\kappa} z_{ij}^k \leq 1 \tag{19}$$

Backup path routing constraints:

$$\sum_{k=0}^{\kappa}\sum_{j\in V(i)}\left(u_{ij}^{sk}-u_{ji}^{sk}\right)=t_i^s-a_i^s \quad s\in N,\ i\in N \tag{20}$$

$$u_{ij}^{s0}+u_{ji}^{s0}\leq 1-z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \tag{21}$$

$$u_{ij}^{sk}+u_{ji}^{sk}\leq z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \tag{22}$$

$$\sum_{k=0}^{\kappa}\left(x_{ij}^{sk}+x_{ji}^{sk}+u_{ij}^{sk}+u_{ji}^{sk}\right)\leq 1 \quad s\in N,\ \{i,j\}\in E \tag{23}$$

$$\sum_{k=0}^{\kappa}\sum_{j\in V(i)}\left(x_{ji}^{sk}+u_{ji}^{sk}\right)\leq 1+a_i^s \quad s\in N,\ i\in N\backslash\{s\} \tag{24}$$

Path availability constraints:

$$\sum_{\{i,j\}\in A}\sum_{k=0}^{\kappa}x_{ij}^{sk}\log(\alpha_{ij}^k)\geq\log(\lambda_p) \quad s\in N \tag{25}$$

$$\sum_{\{i,j\}\in A}\sum_{k=0}^{\kappa}u_{ij}^{sk}\log(\alpha_{ij}^k)\geq\log(\lambda_b) \quad s\in N \tag{26}$$

Spanning tree constraints:

$$\sum_{j\in V(i)}(\beta_{ij}^{\sigma}-\beta_{ji}^{\sigma})=\begin{cases}1 & i=\sigma\\ 0 & i\neq\sigma\end{cases} \quad i\in N\backslash\{\rho\},\ \sigma\in N\backslash\{\rho\} \tag{27}$$

$$\theta_{ij}\geq\beta_{ij}^{\sigma} \quad (i,j)\in A,\ \sigma\in N\backslash\{\rho\} \tag{28}$$

$$\sum_{j\in V(i)}\theta_{ij}=\begin{cases}1 & i\neq\rho\\ 0 & i=\rho\end{cases} \quad i\in N \tag{29}$$

$$z_{ij}^k\leq\theta_{ij}+\theta_{ji} \quad \{i,j\}\in E,\ k=1,..,\kappa \tag{30}$$

Variable domain constraints:

$$y_i\in\{0,1\} \quad i\in N \tag{31}$$

$$x_{ij}^{sk},u_{ij}^{sk}\in\{0,1\} \quad s\in N,\ (i,j)\in A,\ k=0,...,\kappa \tag{32}$$

$$a_i^s\in\{0,1\} \quad s\in N,\ i\in N \tag{33}$$

$$z_{ij}^k\in\{0,1\} \quad \{i,j\}\in E,\ k=1,...,\kappa \tag{34}$$

$$\theta_{ij}\in\{0,1\} \quad (i,j)\in A \tag{35}$$

$$\beta_{ij}^{\sigma}\in\{0,1\} \quad (i,j)\in A,\ \sigma\in N\backslash\{\rho\} \tag{36}$$

The objective function given by (8) aims to minimize the cost of upgrading the links in the spanning tree.

Constraints (9)-(11) refer to the controller placement in the network [14]:

- constraint (9) guarantees that C controllers are placed in the network;
- constraints (10) guarantee that for each switch s, there is a controller distanced at most D_{sc} from it;
- constraints (11) guarantee that any two controllers are distanced at most Dcc from each other.

Constraints (12)-(16) guarantee that the primary paths are routed from each switch s to its controller (located at node i such that $a_i^s = 1$), satisfying the SC maximum delay requirement [4]:

- constraints (12) are the flow conservation constraints for the primary path of switch s to its controller;
- constraints (13) guarantee that if node s hosts a controller, then s is managed by that controller;
- constraints (14) guarantee that any node i distanced further than D_{sc} from s, cannot host the controller managing s;
- constraints (15) guarantee that if node i is distanced at most D_{sc} from s and manages s, then a controller must be hosted in i;
- constraints (16) guarantee that each primary path is not longer than the SC maximum delay requirement.

Constraints (17)-(19) guarantee the correct assignment of the flow variables according to the upgrade level of the links [3]:

- constraints (17) guarantee that variables x_{ij}^{s0} are assigned to the non-upgraded arcs of the primary path from switch s to its controller;
- constraints (18) guarantee that variables x_{ij}^{sk} are assigned to the arcs of the primary path for switch s to its controller, that are upgraded to level $k = 1, ..., \kappa$;
- constraints (19) guarantee that each link is upgraded to one and only one specific level $k = 1, .., \kappa$, or is not upgraded at all ($k = 0$).

Constraints (20)-(24) guarantee the backup path between each switch and its controller, node disjoint to the corresponding primary path:

- constraints (20) are the flow conservation constraints for the backup path of switch s to its controller (not necessary if node s hosts a controller);

- constraints (21) guarantee that variables u_{ij}^{s0} are assigned to the non-upgraded arcs of the backup path from switch s to its controller;

- constraints (22) guarantee that variables u_{ij}^{sk} are assigned to the arcs of the backup path for switch s to its controller, that are upgraded to level $k = 1, ..., \kappa$;

- constraints (23) guarantee that the primary and backup paths for switch s are link disjoint;

- constraints (24) guarantee that the primary and backup paths for switch s are node disjoint. These constraints allow for (23) to be simplified to only encompass the direct link between s and its controller; since the controller location is not known beforehand but is given by the variable $a_i^s = 1$, constraints (23) are maintained.

Constraints (25)-(26) guarantee the availability minimum requirements for the primary and backup paths, applying the linearization used in (3):

- constraints (25) guarantee that the primary path of each switch to its controller has a minimum availability of λ_p;

- constraints (26) guarantee that the backup path of each switch to its controller has a minimum availability of λ_b.

Constraints (27)-(30) guarantee that the upgraded links sit on a spanning tree structure [13]:

- constraints (27) guarantee a routing path from any node σ to the root node ρ;

- constraints (28) account for the spanning tree links given by the routing paths from σ to ρ;

- constraints (29) guarantee a directed spanning tree towards the root node ρ;

- constraints (30) guarantee that the upgraded links belong to the spanning tree.

Finally, constraints (31)-(36) are the variable domain constraints.

Note that we do not force the primary path to be on the spanning tree, since the SC delay cannot exceed D_{sc}, which may happen if the paths are forced to be routed on the

Table 1: Topological characteristics of the networks

Network	#nodes	#links	avg deg	D_g [km]
polska	12	18	3.00	811
nobel_germany	17	26	3.06	790

spanning tree. Nor do we force the backup paths to be on the spanning tree, since this can lead to longer paths. Any of the paths will use the spanning tree if it is necessary to obtain the desired end-to-end availability.

4 Computational Results

For the computational results, we used two networks from SNDlib [10]: polska and nobel_germany. The topological characteristics of these networks are shown in Table 1.

We assume the maximum delay values D_{sc} and D_{cc} are given as percentages of the graph diameter D_g (longest shortest path between two nodes in the network) [11]. For the polska network, we have considered the maximum SC delay value to be $D_{sc} = 35\%, 40\%$, while the maximum CC delay value was considered to be $D_{cc} = 70\%, 75\%$. For the nobel_germany network, we have considered the maximum SC delay value to be $D_{sc} = 35\%, 40\%$, while the maximum CC delay value was considered to be $D_{cc} = 65\%, 70\%$.

For the number of controllers C, we have chosen to start with the minimum value allowed for the delay requirements, and we increment C in order to minimize the upgrade cost as much as possible. However, having a large number of controllers is undesirable for the network operator, since the intercontroller communication overhead increases, jeopardizing the control plane performance. We show the trade-off between the number of controllers and the upgrade cost, for the network operator to weigh the benefits of each solution. Eventually, increasing the number of controllers will result in the problem becoming infeasible, since having too many controllers will not allow for the maximum CC delay value D_{cc} to be satisfied.

Since these networks have a similar graph diameter D_g, we have considered minimum availability values for the primary control paths given by $\lambda_p = 0.999$, and for the backup control paths given by $\lambda_b = 0.99$, to achieve end-to-end availabilities of at least $\lambda = 0.99999$. To obtain these desired availability levels, each link of the selected spanning tree subgraph can be upgraded up to $\kappa = 4$ levels where, for each level, the link unavailability is reduced

Table 2: Computational results for polska; nondominated solutions are marked with *

	$D_{cc} = 70\%$								$D_{cc} = 75\%$							
				no. upg links								no. upg links				
					k								k			
D_{sc}	C	cost	t(s)	total	1	2	3	4	C	cost	t(s)	total	1	2	3	4
35%	3	5054.43*	43.69	11	1	0	3	7	3	4055.60*	280.36	11	1	2	6	2
	4	2995.09*	59.39	9	2	1	4	2	4	2995.09*	117.49	9	2	1	4	2
	5	2469.68*	43.28	9	2	5	2	0	5	2340.76*	15.64	8	1	5	2	0
	6	2299.17*	6.66	8	2	4	2	0	6	2229.16*	10.86	8	2	4	2	0
	7	2266.59*	3.60	9	3	3	3	0	7	2058.65*	3.38	7	2	3	2	0
	8	2266.59	2.08	9	3	3	3	0	8	1947.05*	2.02	6	1	3	2	0
	9	–	0.05	–	–	–	–	–	9	1947.05	0.66	6	1	3	2	0
									10	–	0.05	–	–	–	–	–
40%	3	4445.85*	2525.28	10	2	1	3	4	3	3695.86*	252.18	11	2	2	7	0
	4	2635.35*	121.62	9	3	1	5	0	4	2635.35*	106.75	9	3	1	5	0
	5	2299.17*	26.14	8	2	4	2	0	5	2299.17*	40.43	8	2	4	2	0
	6	2217.38*	8.83	8	2	5	1	0	6	2143.90*	30.69	8	3	3	2	0
	7	2217.38	5.15	8	2	5	1	0	7	1958.83*	5.40	6	2	0	4	0
	8	2184.80*	3.04	9	3	4	2	0	8	1829.91*	3.16	5	1	0	4	0
	9	–	0.06	–	–	–	–	–	9	1795.25*	0.69	5	0	2	3	0
									10	–	0.03	–	–	–	–	–

by $\varepsilon = 0.5$.

The ILP model was implemented in C++, using CPLEX 12.8 callable libraries. 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 for polska are shown in Table 2, whereas the computational results for nobel_germany are shown in Table 3. Each table shows the results for the different combinations of D_{sc} and D_{cc} values. Column 'C' shows the number of controllers, starting with the minimum value allowed for the maximum SC and CC delays [14], and ending with the number of controllers for which the problem is infeasible (since the number of controllers is too large to guarantee the maximum CC delay). Column 'cost' shows the optimal cost of upgrading the links of the selected spanning tree in the network to achieve the required availabilities. Column 't(s)' shows the runtime in seconds for each instance. The following columns are grouped under 'no. upg links': column 'total' shows the total number of upgraded links; and the columns grouped under 'k' show the total number of upgraded links for each level from 1 through 4.

For the polska network, it is possible to observe that for $D_{sc} = 35\%$ and $D_{cc} = 70\%$, there is a decrease in the cost of the link upgrade as the number of controllers increases from 3 to 7. Increasing the number of controllers to 8 does not improve the cost and increasing the number of controllers to 9 renders the problem infeasible (too many controllers to guarantee the D_{cc} maximum value for the CC delay). A similar observation can be done for $D_{sc} = 35\%$ and $D_{cc} = 75\%$, where there is a decrease in the upgrade cost up to controllers and then for 10 controllers the problem is infeasible. The solutions marked with an asterisk (*) show the nondominated set of solutions obtained for the minimization of the number of controllers versus the minimization of the upgrade cost. A nondominated solution is a solution such that any other solution to the problem which has a smaller number of controllers must have a greater upgrade cost, or such that any other solution with a smaller upgrade cost must have a greater number of controllers. In other words, a nondominated solution is such that it is not possible to improve both objectives simultaneously even further.

For $D_{sc} = 40\%$ and $D_{cc} = 70\%$, there is a decrease in the upgrade cost up to 6 controllers. Increasing the number of controllers to 7 does not improve the upgrade cost, however, increasing the number of controllers to 8 does improve the upgrade cost even more, resulting in another nondominated solution for the minimization of the number of controllers versus the minimization of the upgrade cost. For $D_{sc} = 40\%$ and $D_{cc} = 75\%$ there is a decrease in upgrade cost as the number of controllers increases to 9, and then for 10 controllers the problem is infeasible.

For the nobel_germany network, it is possible to observe that for $D_{cc} = 65\%$, there is decrease in the upgrade cost as the number of controllers increases from 2 to 6. Then increasing the number of controllers further, does not improve the upgrade cost. In fact surprisingly, increasing the number of controllers from 9 to 10 actually worsens the upgrade cost. This is due to the fact that fixing the 9 controller placements of the solution $C = 9$, implies that the 10^{th} controller must be placed further than D_{cc} from some of the controllers. The controllers actually begin 'overcrowding' each other and to maintain the CC delay inside the maximum value D_{cc} between all controller pairs, the controllers need to be repositioned leading to higher distances for a few switches and, thus, forcing the link upgrade cost to increase (cost values shown in bold).

Table 3: Computational results for nobel_germany; nondominated solutions are marked with *

	$D_{cc} = 65\%$								$D_{cc} = 70\%$							
D_{sc}	C	cost	t(s)	no. upg links					C	cost	t(s)	no. upg links				
				total	k							total	k			
					1	2	3	4					1	2	3	4
35%	2	3076.88*	171.0	15	5	6	3	1	2	3076.88*	365.3	15	5	6	3	1
	3	2682.48*	14995.0	16	8	6	2	0	3	2682.48*	2401.7	16	8	6	2	0
	4	2068.35*	697.0	14	7	6	1	0	4	2068.35*	1243.3	14	7	6	1	0
	5	1477.79*	131.7	10	5	4	1	0	5	1477.79*	242.1	10	5	4	1	0
	6	1337.08*	12.3	9	5	3	1	0	6	1275.39*	148.6	10	7	3	0	0
	7	1337.08	12.0	9	5	3	1	0	7	1194.99*	22.7	9	6	3	0	0
	8	1337.08	23.2	9	5	3	1	0	8	1118.74*	9.9	9	6	3	0	0
	9	1337.08	16.1	9	5	3	1	0	9	1081.31*	6.5	9	7	2	0	0
	10	**1556.12**	4.4	9	6	2	1	0	10	1081.31	5.7	9	7	2	0	0
	11	**1556.12**	4.1	9	6	2	1	0	11	1081.31	4.8	9	7	2	0	0
	12	–	0.1	–	–	–	–	–	12	1081.31	3.9	9	7	2	0	0
									13	–	0.1	–	–	–	–	–
40%	2	3076.88*	600.4	15	5	6	3	1	2	3076.88*	1093.3	15	5	6	3	1
	3	2672.08*	5214.6	12	5	3	4	0	3	2672.08*	2508.3	12	5	3	4	0
	4	2068.35*	2565.1	14	7	6	1	0	4	2068.35*	3379.3	14	7	6	1	0
	5	1477.79*	293.1	10	5	4	1	0	5	1477.79*	627.5	10	5	4	1	0
	6	1337.08*	95.6	9	5	3	1	0	6	1165.87*	141.8	7	1	5	1	0
	7	1337.08	70.6	9	5	3	1	0	7	955.85*	51.2	7	2	5	0	0
	8	1337.08	94.0	9	5	3	1	0	8	886.54*	8.6	7	3	4	0	0
	9	1337.08	19.3	9	5	3	1	0	9	817.22*	9.0	6	2	4	0	0
	10	**1556.12**	10.9	9	6	2	1	0	10	817.22	10.0	6	2	4	0	0
	11	**1556.12**	7.0	9	6	2	1	0	11	817.22	7.5	6	2	4	0	0
	12	–	0.1	–	–	–	–	–	12	817.22	2.2	6	2	4	0	0
									13	–	0.1	–	–	–	–	–

5 Conclusions

In this paper, we address the control plane design optimization problem of controller placement and selecting a spanning tree, such that the cost of upgrading the links on the spanning tree is minimized, to achieve the imposed availability requirements. We propose an ILP model and show an exact method for linearizing the inherent availability constraints.

Computational results were conducted with two small networks, to assess the trade-off between the number of controllers deployed in the network and the cost of upgrading the links on the spanning tree. The ILP becomes time-consuming for larger networks, justifying a heuristic for these cases, in future work. Having a minimum number of controllers in the network is desired, from the network operator's perspective, to minimize the intercontroller communication overhead. However, this generally implies larger SC delays, decreasing the availability of the control paths. To meet the required availability constraints, more links need to be upgraded and to higher levels, increasing the upgrade cost. Deploying a few additional controllers can drastically decrease the upgrade cost, without jeopardising the intercontroller communication overhead too seriously, On the other hand, having too many controllers not only jeopardizes the communication overhead, but can lead to an increase in upgrade cost, as shown in the computational results.

Acknowledgements

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. Embedded network design to support availability differentiation. *Annals of Telecommunications*, 74(9-10):605–623, 2019.

[3] A. de Sousa, T. Gomes, R. Girão-Silva, and L. Martins. Minimization of the network availability upgrade cost with geodiverse routing for disaster resilience. *Optical Switching and Networking*, 31:127–143, 2019.

[4] A. de Sousa, D. Santos, and P. Monteiro. Determination of the minimum cost pair of D-geodiverse paths. In *The 2017 International Conference on Design of Reliable Communication Networks (DRCN 2017)*, Munich, March 8-10 2017.

[5] U. Franke. Optimal it service availability: Shorter outages, or fewer? *IEEE Transactions on Network and Service Management*, 9(1):22–33, 2012.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. SNDlib 1.0–Survivable Network Design library. *Networks*, 55(3):276–286, 2010. http://sndlib.zib.de.

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

[12] 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.

[13] D. Santos, A. de Sousa, and F. Alvelos. Traffic engineering of telecommunication networks based on multiple spanning tree routing. *Lecture Notes in Computer Science*, 5464:114–129, 2009.

[14] D. Santos, A. de Sousa, and C. Mas Machuca. Robust SDN controller placement to malicious node attacks. In *DRCN 2018, co-located with ICIN 2018*, pages 1–8, Paris, France, 2018.

[15] D. Santos and T. Gomes. Controller placement and availability link upgrade problem in SDN networks. In *RNDM*, pages 1–8, Nicosia, Cyprus, 2019.

[16] 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.

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

[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.