Combined Control and Data Plane Robustness of SDN Networks against Malicious Node Attacks

Dorabella Santos
Instituto de Telecomunicações
Aveiro, Portugal
dorabella@av.it.pt

Amaro de Sousa
Instituto de Telecomunicações
DETI, Universidade de Aveiro
Aveiro, Portugal
asou@ua.pt

Carmen Mas Machuca
Technical University of Munich
Munich, Germany
cmas@tum.de

***Abstract*—In the context of software-defined networking (SDN), we address a variant of the controller placement problem (CPP), which takes into account the network robustness at both control and data plane layers. For given maximum values of switch-controller and controller-controller delays at the regular state (i.e., when the network is fully operational), the aim is to maximize the network robustness against a set of failure states, each state defined as a possible malicious attack to multiple network nodes. We assume that the attacker knows the data plane topology and, therefore, can adopt either one of three commonly considered node centrality attacks (based on the node degree, closeness or betweenness centralities), or an attack to the nodes which are the optimal solution of the critical node detection (CND) problem. We propose a set of robustness metrics which are used to obtain the optimal solutions for the robust CPP variant. We present a set of computational results comparing the average delays and robustness values of the robust CPP solutions against those minimizing only the average switch-controller and controller-controller delays. Moreover, the impact of using the CND based attack in the robustness evaluation of CPP solutions is also assessed in the computational results.**

Keywords— SDN, controller placement, malicious node attacks, integer linear programming

I. INTRODUCTION

In SDN networks, the control plane is decoupled from the data plane, allowing a more efficient network resource management. In its basic configuration, the control plane can be implemented by a single controller that is queried by the switches for data plane routing decisions. In practice, multiple physically distributed controllers must be used to increase availability (*i.e.*, to avoid the single point of failure) and scalability, both in terms of control flow delay (for larger geographical networks) and controller processing load (for higher number of switches and/or higher data flow dynamics). An immediate concern that arises in a SDN network is the number of controllers to be deployed and their placement throughout the network. This problem is known as the controller placement problem (CPP), a facility location problem variant shown to be NP-hard [1].

Large scale failures, caused by natural disasters, technical related issues or malicious human-made activities, can cause serious disruption in telecommunication networks [2]. Malicious attacks are becoming a major concern [3] and such threats have triggered the interest of network operators to evaluate the robustness of their networks.

The SDN control plane can operate either in a logically centralized or a logically distributed mode [4]. In this work, we consider a logically centralized control plane with a flat controller architecture where all controllers are aware of the complete network state (*i.e.*, we consider an active-active back up policy for a faster and seamless restoration). Our CPP considers the following assumptions. Firstly, the controller placement must satisfy given maximum switch-controller (SC) and controller-controller (CC) delays to maintain acceptable control plane performance in the regular state (*i.e.*, when the network is fully operational). Then, since any controller can serve as a backup controller in the logically centralized mode, we assume that each switch connects to its closest controller both in the regular state and in any failure state. Each controller is assumed to be collocated to a switch, which is refereed as controller node.

We address the robust CPP against a set of attacks targeting the simultaneous shut down of up to p nodes assuming that the attacker knows the data plane topology. Moreover, we assume that the shutdown of a node includes the shutdown of the switch and of the collocated controller in the case of a controller node. A vulnerability of SDN networks is that if an attacker shuts down all controller nodes, then the entire control plane fails. So, we assume the operator requires placing $C = p + 1$ controllers compliant with the following robustness property: in any set of p controller nodes failures, a path in the data plane must exist from any surviving switch to the surviving controller. In this way, if all p attacked nodes happen to be controller nodes, at least one controller node survives and the surviving data plane network is still fully connected.

In a recent work [5], we have addressed a similar problem. That work considers node attacks only based on centrality metrics and evaluates the network robustness only at the SDN control plane layer. Here, we extend that work by considering two important issues. Centrality-based metrics are commonly used to model malicious attacks [5–6]. Nevertheless, it is well known that the critical node detection (CND) problem [7] provides solutions that are optimal in the attacker's perspective since they maximally disrupt the data plane network. So, here we include the optimal CND solutions as possible attacks in the evaluation of CPP solutions. Second, the disruption evaluation of a given attack should be performed to both the control plane and the data plane. In general, services between any two SDN switches can only be maintained when both switches can still connect to controllers and also when there is at least one switching path between them in the data plane that

978-3-903176-14-0

survives the attack. So, in this work, the selection of the best robust CPP solutions considers a set of robustness metrics that also includes the data plane disruption evaluation.

In order to assess the delay penalties of robust controller placements when compared with their non-robust counterparts, we also describe how CPP solutions minimizing only the average SC and CC delays can be optimally computed through integer linear programming. Moreover, the impact of using the CND based attacks in the robustness evaluation is also assessed in the computational results.

The paper is organized as follows. Section II describes related work. Section III shows how the non-robust CPP solutions minimizing average SC and CC delays are computed. Section IV describes an enumeration method to obtain a set of robust CPP solutions. Section V describes how CPP solutions are evaluated in order to obtain the best robust solutions. Section VI presents and discusses the computational results. Finally, Section VII presents the main conclusions of the work.

II. Related Work

For a given network, Critical Node Detection (CND) problems aim to optimally remove a subset of nodes (the critical nodes) in order to optimize or achieve a given network degradation metric. CND problems have been considered in different contexts [7–8] and are gaining special attention in the evaluation of telecommunication networks to large-scale disasters [9–10] by using the optimal solution of CND as the network vulnerability metric. Moreover, recent works use CND to enhance the network robustness to multiple node failures. In [11], the authors optimally select a set of r nodes that must be made robust in order to maximally improve the vulnerability of the network given by CND. In [12], the authors consider the addition of new links, within a given budget, to a transparent optical network so that the CND evaluation of the resulting topology is maximally improved.

Most literature on CPP aims at minimizing SC delays without addressing failure resilience. In [13], the authors aim at minimizing combinations of average and maximum delays and also consider load balancing of the controllers. In [14], the authors aim at optimizing several objectives (number of controllers, maximum SC delay, maximum CC delay and controller load imbalance) providing solutions based on search methods for different objective combinations, on a software platform named POCO. The CC delay is also an important requirement in the logically centralized control plane, supported for example by the two major SDN implementations ONOS [15] and ODL [16], to guarantee efficient controller synchronization for consistency reasons. However, minimizing the average SC and CC delays are conflicting objectives [17] since considering more controllers will, in general, decrease the average SC delay but increase the average CC delay.

Different works have addressed the issue of making the CPP more resilient to failures. In [18], the authors first address the CPP for the regular state, minimizing the number of controllers, while guaranteeing maximum values for the SC and CC delays. Then, they address a resilient CPP variant assuming that switches reconnect to the closest surviving controller when they lose connectivity with their primary controller. Controllers are assumed to fail with a given probability and the average SC delays take into account these failure probabilities. The objective is a combination of minimizing the number of controllers and the average SC delays. This work considers only failures on the control plane.

In [19], a CPP is proposed in order to guarantee two node disjoint paths from each switch to its primary controller, and another CPP is proposed guaranteeing node disjoint paths from each switch to its primary and to its backup controller. These CPP solutions show enhanced robustness with small SC delay penalties. In [20], the resilient capacitated CPP is addressed considering multiple controller failures where each switch has a given traffic load and controllers have an associated capacity. An Integer Linear Programming (ILP) model minimizing the number of controllers is proposed. The ILP guarantees (i) the assignment of r controllers to each switch and (ii) given maximum values for the SC and CC delays. Then, the ILP can be extended to ensure that all control paths of each switch are link-disjoint. Both [19–20] consider only single link and/or node failures at the SDN data plane.

The authors in [6] address targeted attacks to a SDN network. Assuming that the attacker has knowledge of the data plane topology but is unaware of the controller locations, the network vulnerabilities to centrality-based attacks are studied. The controller placements are proposed to be the least critical nodes, *i.e.*, the nodes less chosen by the different attacks. In [21], the CPP is addressed for a multiple failure scenario where the SDN controller locations are based on a failure correlation assessment of network nodes and links. The authors consider different types of minimal cut sets composed of nodes and/or links, to assess the network unavailability.

III. The Non-Robust CPP

Consider a data plane network represented by a directed graph $G_A = (N, A)$, where N is the set of nodes and A is the set of directed links. The number of nodes is given by $n = |N|$ and the directed link from node i to node j is given by arc (i, j). The set of the adjacent nodes of i is denoted as $V(i)$. Given the propagation delay of each arc, the shortest path delay between nodes i and j is pre-computed and denoted as d_{ij}.

The non-robust CPP focuses on the regular state, aiming to optimize the control plane performance, by minimizing either the average SC delay or the average CC delay [5]. We further assume that the SC delay between any switch and its primary controller does not exceed a given D_{sc}, and that the CC delay between any two controllers does not exceed a given D_{cc}. Consider the decision variables given by:

$y_i \in \{0,1\}$	binary variable that is 1 if a controller is placed in i (*i.e.*, node i is a controller node) and 0 otherwise (*i.e.*, node i is a switch)
$z_{ij} \in \{0,1\}$	binary variable that is 1 if the primary controller of switch i is placed in node j, and 0 otherwise
$c_{ij} \in \{0,1\}$	binary variable that is 1 if one controller is placed in i and another controller is placed in j, and 0 otherwise (this means that $c_{ij} = y_i \cdot y_j$)

The following ILP constraints define the set of all feasible CPP solutions:

$$\sum_{i\in N} y_i = C \quad (1)$$

$$\sum_{j:d_{ij}\le D_{sc}} y_j \ge 1 \qquad i \in N \quad (2)$$

$$y_i + y_j \le 1 \qquad i \in N, j \in N\backslash\{i\}: d_{ij} > D_{cc} \quad (3)$$

$$\sum_{j:d_{ij}\le D_{sc}} z_{ij} = 1 \qquad i \in N \quad (4)$$

$$z_{ij} \le y_j \qquad i \in N, j \in N \quad (5)$$

$$c_{ij} \le y_i \qquad i \in N, j \in N: i < j \quad (6a)$$

$$c_{ij} \le y_j \qquad i \in N, j \in N: i < j \quad (6b)$$

$$c_{ij} \ge y_i + y_j - 1 \qquad i \in N, j \in N: i < j \quad (6c)$$

$$y_i \in \{0,1\} \qquad i \in N \quad (7a)$$

$$z_{ij} \in \{0,1\} \qquad i \in N, j \in N: d_{ij} \le D_{sc} \quad (7b)$$

$$c_{ij} \in \{0,1\} \qquad i \in N, j \in N: i < j \quad (7c)$$

Constraint (1) guarantees the placement of C controllers. Constraints (2) guarantee that for each node $i \in N$, there must exist at least one controller placed in a node distanced at most D_{sc} from i (guaranteeing the maximum SC delay), and constraints (3) guarantee that any pair of controllers are not placed in nodes distanced more than D_{cc} from each other (guaranteeing the maximum CC delay). Constraints (4–5) guarantee that each node i has exactly one primary controller in a node distanced at most D_{sc} and, when used, render constraints (2) redundant. Constraints (6) are the linearization of the equalities $c_{ij} = y_i \cdot y_j$ and constraints (7) are variable domain constraints. For each feasible solution, the average SC delay of the $n - C$ nodes not hosting a controller is given by

$$f_{sc} = \frac{1}{n-C} \sum_{i\in N} \sum_{j\in N\backslash\{i\}} d_{ij} z_{ij}$$

while the average CC delay of the $C(C-1)/2$ controller node pairs is given by

$$f_{cc} = \frac{2}{C(C-1)} \sum_{i\in N} \sum_{j\in N: i<j} d_{ij} c_{ij}$$

The aim is to optimize the SC and CC delays, which are conflicting objectives [17]. The joint optimization of these objectives is a bi-objective optimization problem, which has multiple optimal solutions, known as Pareto solutions. In this work, we consider only the two Pareto opposites defined as:

MinAvgSC optimization problem that first minimizes f_{sc} subject to (1–7) and, then, minimizes f_{cc} subject to (1–7) and guaranteeing the minimum value of f_{sc}

MinAvgCC optimization problem that first minimizes f_{cc} subject to (1–7) and, then, minimizes f_{sc} subject to (1–7) and guaranteeing the minimum value of f_{cc}

The solution of each problem is determined by solving in sequence two ILP models. In our instances, both problems were efficiently solved (we used CPLEX 12.6.1 ILP solver) with total runtime always below 6 seconds in all cases.

IV. Enumeration of Robust CPP Solutions

In order to enumerate the robust CPP solutions with $C = p + 1$ controllers, we first describe through integer linear programming the set of feasible solutions where the robustness property is imposed: in any set of p controller nodes failures, a path in the data plane must exist from any surviving switch to the surviving controller. Consider the previous variables y_i and the following integer variables:

$x_{ij}^k \in N_0^+$	non-negative integer variable indicating the number of paths that include arc (i,j) from switch k to all controller nodes

Following [5], the following constraints define the set of all robust CPP solutions:

(1–3), (7a)

$$\sum_{j\in V(i)} \left(x_{ij}^k - x_{ji}^k\right) \le y_i \qquad k \in N, i \in N\backslash\{k\} \quad (8)$$

$$\sum_{j\in V(i)} \left(x_{ij}^k - x_{ji}^k\right) \ge 0 \qquad k \in N, i \in N\backslash\{k\} \quad (9)$$

$$\sum_{j\in V(i)} x_{ij}^k \le C(1 - y_i) \qquad k \in N, i \in N \quad (10)$$

$$\sum_{j\in V(i)} x_{ji}^k \ge y_i - y_k \qquad k \in N, i \in N\backslash\{k\} \quad (11)$$

$$x_{ik}^k = 0 \qquad k \in N, i \in V(k) \quad (12)$$

$$x_{ij}^k \in \mathbb{N}_0^+ \qquad k \in N, (i,j) \in A \quad (13)$$

Constraints (8–13) guarantee the robustness property. For each node $k \in N$:

- if node $i \in N\backslash\{k\}$ is not a controller node (*i.e.*, $y_i = 0$), constraints (8–9) become the usual path conservation constraints and constraints (10–11) become redundant;
- if node $i \in N$ is a controller node (*i.e.*, $y_i = 1$), constraints (10) guarantee that there is no outgoing arc in node i (to ensure that no path includes intermediate controller nodes) and constraints (8–9,11) guarantee that there is exactly one path ending at node i (to ensure that there is a path from k to each controller node).

In addition, constraints (12) guarantee that there is no path originating at k that ends at k, and constraints (13) are the variable domain constraints of variables x_{ij}^k.

In order to enumerate all robust CPP solutions, the approach in [5] is used. A flowchart with the enumeration method is presented in Fig. 1. First, the optimal solution of the following ILP model is computed:

Maximize $\sum_{i\in N} \gamma_i y_i$ (14)

Subject to

(1–3), (7a), (8–13)

where coefficients γ_i of the objective function (14) are the nodes' closeness centrality metric (motivated by the assumption that nodes with higher closeness centrality are more promising candidates for placing controllers). Then, we use a random walk procedure to compute many new robust CPP solutions. The random walk randomly swaps a controller from its current node location to an available adjacent node that

does not yet host a controller. If the generated solution is new (*i.e.*, it has not been previously found) and feasible (given by the feasibility test), the random walk procedure continues with the new solution; otherwise, it discards the solution and goes back to the previous one. The random walk ends when a maximum number $I_{\max}$ of solutions are discarded.

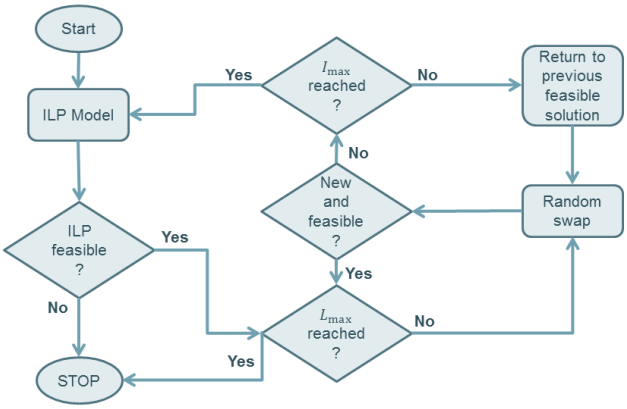

Fig. 1. Flow chart of the enumeration method.

When no new solution can be computed by the random walk, for each obtained CPP solution $\{\rho_1, \dots, \rho_C\}$, a constraint is added to the ILP model in the form:

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

to remove it from the feasible set of solutions and the whole process is repeated (*i.e.*, we solve again the ILP model and based on its optimal solution, we run again the random walk). The whole enumeration method stops when the ILP model is infeasible (which means the complete set of robust feasible CPP solutions was found) or when a pre-defined number of CPP solutions, defined as $L_{\max}$, is reached.

The feasibility test of a solution has polynomial complexity. First, the maximum D_{sc} and D_{cc} delays in the regular state are straightforwardly checked with the shortest path delays d_{ij} between all pairs of nodes of graph G. Then, we start by eliminating from G all outgoing arcs (i,j) of all nodes i hosting a controller, and then running a shortest path algorithm between all node pairs in the new graph, which has complexity $\mathcal{O}(n^3)$. To check if there is a path in the SDN data plane between any switch and any controller node, we just need to verify if the shortest path from every switch to every controller node is less than infinity.

V. Evaluation of the Robust CPP Solutions

In order to evaluate the robustness of a controller placement solution, as the robust ones determined by the enumeration method (Section III) or the *MinAvgSC* and *MinAvgCC* solutions (Section II), we first select the malicious attacks of interest targeting up to p node shutdowns. Then, we propose a set of robustness metrics and evaluate each controller placement by computing its robustness metric values for the considered attacks of interest. At the end, based on the robustness metric values of each solution, the selection of the controller placements with the best robustness values is straightforward.

A. Modelling Malicious Node Attacks

We assume the attacker has full knowledge of the SDN data plane topology, but is unaware of the controller locations. Under this assumption, we consider that the attacker can adopt either a node centrality based attack or an attack to the nodes which are the optimal solution of the CND problem.

The different centrality measures give different information concerning network connectivity. Three node centrality measures are considered: (i) node degree, *i.e.*, nodes with the largest number of neighbors are preferred; (ii) closeness centrality, *i.e.*, nodes closest to all other nodes are preferred; and (iii) betweenness centrality, *i.e.*, nodes serving the largest number of shortest paths are preferred. In each of these attacks, the nodes with greatest centrality value are preferred since shutting them down should cause major network disruption in from the attacker's perspective. To select the p nodes for shut down, the centrality measures are first computed in the complete network graph. Then, for each centrality measure, the node with greatest centrality value is selected and removed from the graph, the centrality node values are recomputed for the remaining graph and the process is repeated until p nodes are selected.

In the context of our problem, we consider the CND problem variant aiming to identify a set of p nodes in the data plane network that when shutdown maximizes the number of node pairs that become disconnected. CND can be formulated as an ILP model as follows. Consider the data plane network represented by the undirected graph $G_E = (N, E)$, where E is the set of undirected links and each link is given by $(i,j) \in E$ with $i < j$. Consider $V(i)$ as the set of neighboring nodes of i in G_E. Also consider the following auxiliary sets: the set of all node pairs $T = \{(i,j) \in N \times N : i < j\}$; the complementary set of E given by $E_c = \{(i,j) \in T : (i,j) \notin E\}$; and set $V(i,j)$ defined as $V(i)$ if $|V(i)| \leq |V(j)|$, or $V(j)$ otherwise. Consider the decision variables given by:

$v_i \in \{0,1\}$	binary variable that is set to 1 if node i is selected as a critical node, and 0 otherwise
$u_{ij} \in \{0,1\}$	binary variable that is set to 1 if nodes i and j are connected when the critical nodes are removed, and 0 otherwise ($i < j$)

For readability purposes, both u_{ij} and u_{ji} appear in the following formulation but they represent the same variable u_{ij} with $i < j$. Following [10], CND is defined by:

Minimize $\sum_{(i,j)\in T} u_{ij}$ (14)

Subject to:

$$\sum_{i\in N} v_i = p \quad (15)$$

$$u_{ij} + v_i + v_j \geq 1 \qquad (i,j) \in E \quad (16)$$

$$u_{ij} \geq u_{ik} + u_{jk} - 1 + v_k \qquad (i,j) \in E_c, k \in V(i,j) \quad (17)$$

$$v_i \in \{0,1\} \qquad i \in N \quad (18a)$$

$$u_{ij} \in \{0,1\} \qquad (i,j) \in T \quad (18b)$$

The objective function (14) is the minimization of the number of node pairs that remain connected when the critical

nodes are removed. Constraint (15) imposes that exactly p critical nodes are selected. Constraints (16) guarantee that if there is a link between nodes i and j, then they are connected if none of them are critical. Constraints (17) guarantee that for nodes i and j that are not critical and do not share a link, then they are connected if a neighbor node k of one of them is connected to the other. Constraints (18) are the variable domain constraints. In our computational instances, CND is efficiently solved (we used CPLEX 12.6.1 ILP solver) with total runtime always below 4 seconds in all cases.

Note that each malicious attack is defined by a set of p nodes that are selected by the attacker to be shut down and this set is computed based only on the data plane topology. So, the set of p nodes of each attack are computed only once for each data plane topology. Consider the set M = {1,2,3,4} representing all attacks such that $m = 1,2,3$ represent the node degree, closeness and betweenness based attack, respectively, and $m = 4$ represents the CND based attack. In order to compare the robustness evaluation with and without the CND based attack, we represent by M' the subset composed by only the centrality-based attacks, *i.e.*, M' = {1,2,3}.

B. Evaluation of CPP Solutions

Recall that for a given data plane network, a CPP solution is a set of C nodes that host a controller each. Consider G_m as the surviving graph to the malicious attack $m \in M$, *i.e.*, when the p nodes of attack m are eliminated from the data plane graph. To evaluate each CPP solution, we consider three robustness metrics.

The first metric is the switch pair connectivity metric n_{sp}. To compute this metric, we first determine n_{sp}^m, given by the number of switch pairs that are connected in G_m and both switches can connect to a SDN controller. Then, we define $n_{sp} = \min_{m \in M} n_{sp}^m$. This is the most important robustness metric. It represents the minimum number of switch pairs that are still able to support data flows amongst all considered malicious attacks and the aim is to find the CPP solution that maximizes it.

If multiple solutions have the maximum metric value n_{sp}, then the second metric, the switch-controller metric n_{sc}, is considered. To compute this metric, we first determine n_{sc}^m given by the number of switches that can connect to a surviving controller distanced at most D_{sc} in G_m. Then, we define $n_{sc} = \min_{m \in M} n_{sc}^m$. This is the second most important robustness metric. It represents the minimum number of switches with available SDN control plane within the maximum SC delay amongst all considered malicious attacks.

Finally, if multiple solutions have the maximum metric values n_{sp} and n_{sc}, the third proposed metric is the primary controller metric n_{pc}. To compute this metric, we first determine n_{pc}^m given by the number of switches whose closest controller in G_m is its primary controller (*i.e.*, is the closest controller before attack m). Then, we define $n_{pc} = \min_{m \in M} n_{pc}^m$. This metric represents the number of switches that do not require a change of their primary controller avoiding temporary control plane disruption.

Note that the determination of n_{sp}^m, n_{sc}^m and n_{pc}^m for each $m \in M$ is polynomial since these values can be computed with the shortest path lengths between every node pair and these lengths are computed by running a shortest path algorithm between all node pairs in G_m (which has complexity $\mathcal{O}(n^3)$).

VI. Computational Results

In this section, we present a comparison analysis between the non-robust optimal CPP solutions and the best robust CPP solutions in terms of average delays versus robustness to malicious node attacks. Moreover, the influence of using the CND based attack in the robustness evaluation of all CPP solutions is also assessed. All methods were implemented in C++, using CPLEX 12.6 callable libraries running 8 threads. All problem instances were solved on a 16 core server with 64 GB of RAM running Windows OS. To define the instances for the computational results, two topologies were considered as SDN data plane networks: Germany50 with 50 nodes, 88 undirected links and an average node degree of 3.52, depicted in Figure 2 (sndlib.zib.de) and CORONET CONUS with 75 nodes, 99 undirected links and an average node degree of 2.64, depicted in Figure 3 (www.monarchna.com/ topology.html).

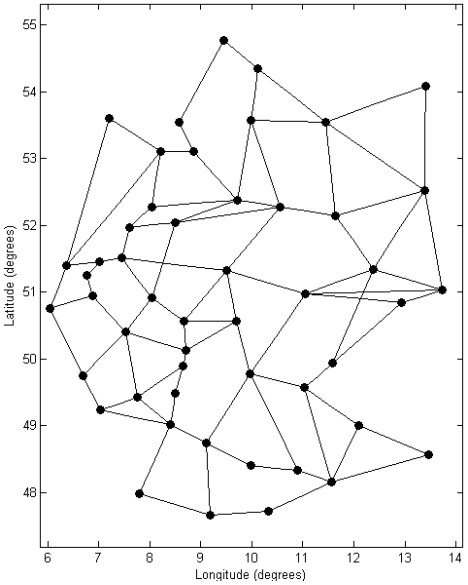

Fig. 2. Germany50 with 50 nodes and 88 links.

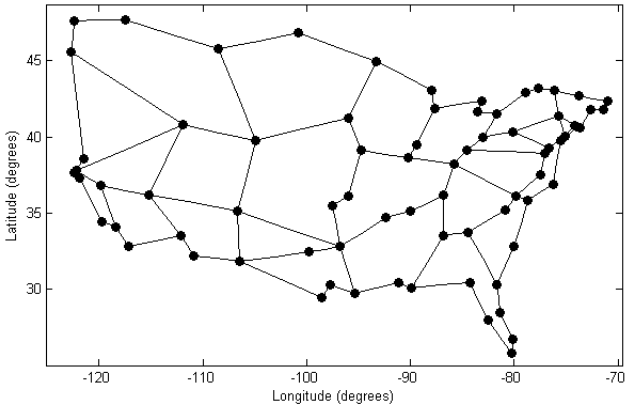

Fig. 3. CORONET CONUS with 75 nodes and 99 links.

As in [5], given the geographical coordinates of the nodes, each link length was computed as the shortest distance between its two end nodes, over the Earth's surface. Moreover, the delay between two nodes d_{ij} is given by the shortest path length between them. The maximum delay requirements D_{sc} and D_{cc} are given as percentages of the graph diameter, which is given by the largest delay between any pair of nodes in the network. The graph diameter for Germany50 is 934 km, while for CORONET CONUS is 6472 km. The values used for the number of attacked nodes p are 3, 5 and 7. Since $C = p + 1$, the number of controllers is 4, 6 and 8, respectively. For each value of C, three sets of D_{sc},D_{cc} values have been considered (presented in Table I), representing different tradeoffs between the SC and CC delays. These sets of values were chosen to be tight, while guaranteeing that the non-robust and robust problems were still feasible resulting in a total number of 9 problem instances for each network topology.

TABLE I. MAXIMUM DELAYS OF EACH PROBLEM INSTANCE

Instance ID		1	2	3	4	5	6	7	8	9
C		4	4	4	6	6	6	8	8	8
Germany50	D_{sc} **(%)**	30	35	40	25	30	35	20	25	30
	D_{cc} **(%)**	60	40	35	65	60	40	75	65	60
CORONET CONUS	D_{sc} **(%)**	30	35	40	20	25	30	20	25	30
	D_{cc} **(%)**	55	40	30	80	55	50	65	55	50

Based on preliminary tests, the maximum number $L_{\max}$ of CPP solutions in the enumeration method (see Section IV) was set to 100000 and the maximum number $I_{\max}$ of consecutive solution discards by the random walk was set to 10000.

The optimal non-robust CPP solutions were obtained by solving *MinAvgSC* and *MinAvgCC* (see Section II). For each of these two solutions, besides the average SC and CC delays (given by the optimal values of the objective functions), their robustness metrics n_{sp}, n_{sc} and n_{pc} were also computed as described in Section V.B. The robustness parameters were computed both against the set M of attacks (which include the CND based attack) and against the subset M' (composed only by the centrality-based attacks). Then, the best robust CPP solutions were computed. For each solution, besides the robustness metrics n_{sp}, n_{sc} and n_{pc} (computed as described in Section V), their average SC and CC delays were also computed. Like in the non-robust CPP solutions, the robustness parameters were computed both against the set M of attacks and against the subset M'.

Table II presents the obtained robustness metric values of all solutions for the nine Germany50 network instances. Column 'ID' identifies the instance ID (as defined in Table I) and column 'p' identifies the number of attacked nodes (recall that $C = p + 1$). The last row of Table II presents the average values of each column which enable us to compare the robustness of the different CPP solutions.

The results presented in Table II highlight the impact of using the CND based attack in the robustness evaluation of all controller placement solutions. When only centrality-based attacks are considered, the main robustness metric value n_{sp}^m is much higher than that value when considering all attacks, misleading us to conclude that the network is much more robust to malicious node attacks than it really is. As a consequence, when only centrality-based attacks are considered, all robustness metric values are very similar, on average, for the *MinAvgSC* and *MinAvgCC* non-robust solutions. Moreover, the best robust CPP solution can only improve the least important robustness metric values n_{sc}^m and n_{pc}^m when compared with the non-robust solutions.

On the other hand, when all attacks are considered (*i.e.*, including CND), we observe that the *MinAvgSC* solution is more robust in the main robustness metric value n_{sp}^m than the *MinAvgCC* solution. Note that minimizing the average SC delay spreads out the controllers while minimizing the average CC delay tends to concentrate controllers in the center of the network. So, the results indicate that having more spread out controllers makes the solutions more robust to malicious node attacks. Finally and more importantly, the best robust CPP solutions are now able to improve, on average, all robustness metric values, when compared with the previous non-robust CPP solutions. Although these improvements are more significant for the robustness metric values n_{sc}^m and n_{pc}^m, there are also cases where the main robustness metric value n_{sp}^m is also improved. Overall, by including the CND based attack in the robustness evaluation, besides obtaining a more accurate evaluation, we are able to compute more robust CPP solutions.

Table III presents the obtained robustness parameters of all solutions for the nine CORONET CONUS network instances. Like in the previous case, when only centrality-based attacks are considered (set *M'*), the main robustness metric value n_{sp}^m is much higher than that value when considering all attacks (set *M*). Now, in both cases (either considering *M'* or *M*), we observe that the *MinAvgSC* solution is more robust in all robustness metric values than the *MinAvgCC* solution. Recall that CORONET CONUS topology has a much lower average node degree than Germany50. This means that more spread out controllers make the solutions even more robust to malicious node attacks for network topologies with lower average node degrees. Moreover, when all attacks are considered, the best robust CPP solutions are able to improve only slightly, on average, the main robustness metric value n_{sp}^m when compared with the *MinAvgSC* solution but it improves significantly the other two robustness metric values.

TABLE II. ROBUSTNESS PARAMETERS OF THE NON-ROBUST AND ROBUST SOLUTIONS FOR GERMANY50

		Centrality based Attacks (M')									***Centrality + CND based Attacks (M)***								
		MinAvgSC			***MinAvgCC***			***Robust CPP***			***MinAvgSC***			***MinAvgCC***			***Robust CPP***		
ID	p	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}
1		1081	44	42	1081	47	41	1081	47	47	711	36	28	711	40	39	711	44	41
2	**3**	1081	34	29	1081	34	29	1081	34	29	666	34	29	666	34	29	666	34	29
3		1081	47	29	1081	47	37	1081	47	46	666	37	30	666	37	34	711	45	32
4		990	28	24	990	39	37	990	43	39	496	28	24	496	37	34	496	43	39
5	**5**	990	33	28	990	38	29	990	44	43	496	33	28	496	38	29	496	44	40
6		990	37	29	990	39	30	990	42	26	496	37	29	171	19	18	496	40	26
7		441	29	24	441	23	18	441	34	31	301	29	24	181	23	18	301	34	31
8	**7**	441	31	24	441	30	25	441	40	37	301	31	24	301	30	25	301	40	35
9		441	26	19	441	31	16	441	40	36	301	26	19	301	31	16	301	40	35
Avg:		837.3	34.3	27.6	837.3	36.4	29.1	837.3	41.2	37.1	492.7	32.3	26.1	443.2	32.1	26.9	497.7	40.4	34.2

TABLE III. ROBUSTNESS PARAMETERS OF THE NON-ROBUST AND ROBUST SOLUTIONS FOR CORONET CONUS

		Centrality based Attacks (M')									***Centrality + CND based Attacks (M)***								
		MinAvgSC			***MinAvgCC***			***Robust CPP***			***MinAvgSC***			***MinAvgCC***			***Robust CPP***		
ID	p	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}	n_{sp}	n_{sc}	n_{pc}
1		2485	69	62	2485	67	37	2485	71	65	2016	64	62	2016	64	37	2016	64	64
2	**3**	2485	49	33	2485	66	52	2485	68	63	2016	49	33	2016	64	52	2016	64	63
3		2485	56	39	2485	56	38	2485	56	39	2016	56	39	2016	56	38	2016	56	39
4		1191	49	45	1191	60	57	1191	61	59	983	49	45	983	60	57	983	61	59
5	**5**	1191	59	48	1191	60	49	1191	60	49	973	59	48	973	60	49	973	60	49
6		1191	65	53	561	32	31	1191	65	54	973	63	53	561	32	31	974	64	56
7		849	50	48	849	63	60	849	64	61	610	50	48	620	63	60	620	64	61
8	**7**	849	64	53	849	61	48	849	64	55	620	64	53	610	61	48	620	64	54
9		771	55	47	561	32	30	771	55	52	620	55	47	430	32	30	620	55	52
Avg:		1499.7	57.3	47.6	1406.3	55.2	44.7	1499.7	62.7	55.2	1203.0	56.6	47.6	1136.1	54.7	44.7	1204.2	61.3	55.2

So far, we have shown that the method proposed to compute the best robust CPP solutions can provide solutions more robust than those minimizing only the average SC and CC delays. Now, let us focus on the average SC and CC delay penalties of the best robust CPP solutions.

Figure 4 presents as bar charts the average SC delays of *MinAvgSC* solutions, best robust CPP solutions against set *M'* and against set *M* for all instances of both networks. As expected, the average SC delays of the robust CPP solutions are always higher than the optimal values of the *MinAvgSC* solutions. Nevertheless, the delay penalties are small in all cases showing that the improved robustness against malicious node attacks does not significantly degrade the SDN control plane performance in the regular state concerning switch-controller delays. Another interesting observation from these results is that the robust CPP solutions obtained by including the CND based attack do not necessarily have higher average SC delays since there are some cases where the delay penalties of the CPP solutions against set *M* are lower than those of the CPP solutions against set *M'*.

Figure 5 presents as bar charts the average CC delays of *MinAvgCC* solutions, best robust CPP solutions against set *M'* and against set *M* for all instances of both networks. As before, the average CC delays of the robust CPP solutions are always higher than the optimal values of the *MinAvgCC* solutions. In these cases though, the CC delay penalties are more significant in some instances than in the previous cases, showing that, as already observed before, the more robust solutions tend to require more spread out controllers, which in turn, impose higher average delays between controllers. Note though, that the average CC delays might not be so relevant in practice

since the maximum CC delay (parameter D_{cc} imposed in all CPP solutions) is the main parameter that impacts the synchronization efficiency between controllers [17] and the average CC delays of the solutions are all well below the maximum value considered for each instance. As a final remark, the conclusion drawn before between the delay penalties of the CPP solutions computed against set *M* and against set *M'* also stands in this case, *i.e.*, the robust CPP solutions obtained by including the CND based attack have lower average CC delays in some of the instances.

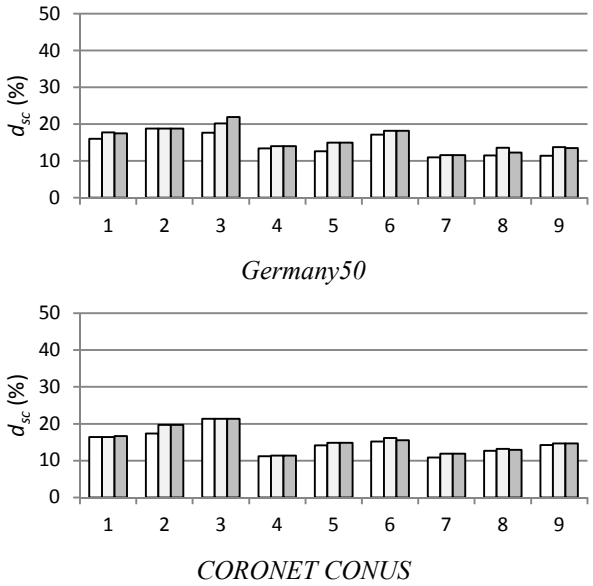

Fig. 4. Average SC delays of the *MinAvgSC* solutions (white columns), best robust CPP solutions against set M' (grey columns) and against set *M* (dark grey columns) for all instances of Germany50 (top) and CORONET CONUS (bottom).

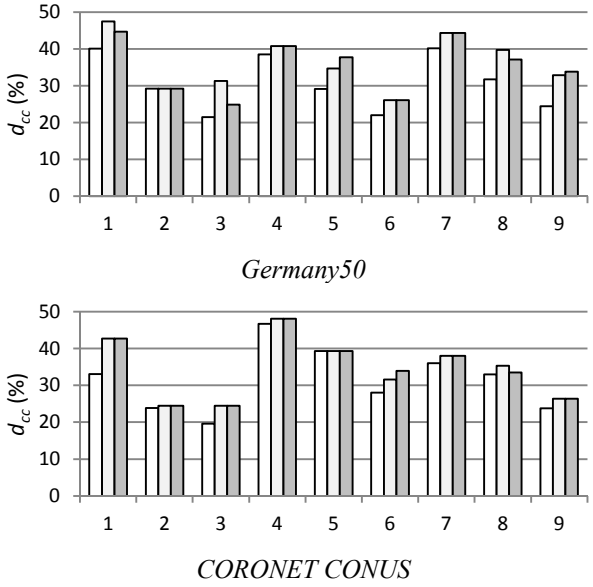

Fig. 5. Average CC delays of the MinAvgCC solutions (white columns), best robust CPP solutions against set M' (grey columns) and against set *M* (dark grey columns) for all instances of Germany50 (top) and CORONET CONUS (bottom).

VII. Conclusions

The robustness of SDN networks to malicious node attacks requires both the data plane and the control plane disruption evaluation. This is because services between any two SDN switches can only be maintained when both switches can still connect to controllers, and also when there is at least one path between them in the data plane that survives the attack. To this aim, we have addressed a variant of CPP which takes into account the network robustness at both control and data plane levels. For given maximum values of SC and CC delays at the regular state, the aim of our CPP variant was to maximize the SDN robustness against a set of failure states, each one given by a possible malicious attack to multiple nodes.

We have assumed that the attacker knows the data plane topology and, therefore, can adopt either one of three commonly considered node centrality attacks (node degree, node centrality of node betweenness) or an attack to the nodes which are the optimal solution of the CND problem.

We have also described how CPP solutions minimizing only the average SC and CC delays can be optimally computed through integer linear programming. Then, in the computational results, a comparison analysis has been conducted between the non-robust and best robust CPP solutions in terms of average delays versus robustness to malicious node attacks. The computational results have shown that enhanced robustness against malicious attacks to multiple nodes are obtained with small average SC delay penalties while the CC delay penalties might be more significant although well below the required maximum values.

Concerning the impact of using the CND based attack in the robustness evaluation of controller placements, the computational results have shown that it enables to compute more robust CPP solutions. Moreover, the results have also shown that the robustness evaluation becomes more accurate with the CND based attack since when we only consider centrality-based attacks, the switch pair connectivity metric n_{sp} is much higher than that value when considering all attacks, misleading us to conclude that the network is much more robust to malicious node attacks than it really is.

Acknowledgment

This paper is based upon work from COST Action CA15127 ("Resilient communication services protecting end user applications from disaster-based failures – RECODIS") supported by COST (European Cooperation in Science and Technology). The work was financially supported by FCT ("Fundação para a Ciência e Tecnologia"), Portugal, under the projects CENTRO-01-0145-FEDER-029312 and UID/EEA/50008/2013 and through the postdoc grant SFRH/BPD/111503/2015.

References

[1] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem", ACM HotSDN, New York, USA, pp. 7-12 (2012).

[2] J. Rak, D. Hutchison, E. Calle, T. Gomes, M. Gunkel, P. Smith, J. Tapolcai, S. Verbrugge, and L. Wosinska, "RECODIS: Resilient communication services protecting end-user applications from disaster-based failures", ICTON, Trento, Italy, We.D1.4 (2016).

[3] M. Furdek, L. Wosinska, R. Goscien, K. Manousakis, M. Aibin, K. Walkowiak, S. Ristov, and J. Marzo, "An overview of security challenges in communication networks", RNDM, pp. 43-50, Halmsatd, Sweden (2016).

[4] M. Karakus, and A. Durresi, "A survey: control plane scalability issues and approaches in Software-Defined Networking (SDN)", Computer Networks, vol. 112, pp. 279-293, 2017.

[5] D. Santos, A. de Sousa, and C. Mas Machuca, "Robust SDN Controller Placement to Malicious Node Attacks", DRCN, co-located with ICIN, Paris, France, pp. 1-8 (2018).

[6] D. F. Rueda, E. Calle and J. L. Marzo, "Improving the Robustness to Targeted Attacks in Software Defined Networks (SDN)", DRCN, Munich, Germany, pp. 78-85 (2017).

[7] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos, "Detecting Critical Nodes in Sparse Graphs", Computers & Operations Research, vol. 36, pp. 2193–2200, 2009.

[8] A. Veremyev, V. Boginski, and E. Pasiliao, "Exact Identication of Critical Nodes in Sparse Networks via New Compact Formulations", Optimization Letters, vol. 8, pp. 1245–1259, 2014.

[9] T. Dinh, Y. Xuan, M. Thai, P. Pardalos, and T. Znati, "On new approaches of assessing network vulnerability: hardness and approximation", IEEE/ACM Trans. on Networking, vol. 20, pp. 609–619, 2012.

[10] D. Santos, A. de Sousa, and P. Monteiro, "Compact Models for Critical Node Detection in Telecommunication Networks", Electronic Notes in Discrete Mathematics, vol. 64, pp. 325-334, 2018.

[11] A. de Sousa, D. Mehta, and D. Santos, "The Robust Node Selection Problem aiming to Minimize the Connectivity Impact of any Set of *p* Node Failures", DRCN, Munich, Germany, pp. 138-145 (2017).

[12] F. Barbosa, A. de Sousa, and A. Agra, "Topology Design of Transparent Optical Networks Resilient to Multiple Node Failures", RNDM Longyearbyen, Norway, pp. 1-8 (2018).

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

[14] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks", ITC, Shanghai, China, pp. 1-9 (2013).

[15] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS", ACM HotSDN, New York, USA, pp. 1-6 (2014).

[16] OpenDaylight: A Linux foundation collaborative project. [Online]. Available: http://www.opendaylight.org.

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

[18] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture", DRCN, Paris, France, pp. 145-151 (2016).

[19] P. Vizarreta, C. Mas Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane", RNDM, Halmstad, Sweden, pp. 253-259 (2016).

[20] M. Tanha, D. Sajjadi, R. Rubyy, and J. Pan, "Capacity-aware and Delay-guaranteed Resilient Controller Placement for Software-Defined WANs", IEEE Trans. on Network and Service Management, vol. 15, no. 3, pp. 991-1005, 2018.

[21] G. Nencioni, B. E. Helvik, and P. E. Heegaard, "Including Failure Correlation in Availability Modeling of a Software-Defined Backbone Network", IEEE Trans. on Network and Service Management, vol. 14, no. 4, pp. 1032-1045, 2017.