Optical Switching and Networking 30 (2018) 10–22

Contents lists available at ScienceDirect

Optical Switching and Networking

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

Robust hierarchical control plane for Transport Software-Defined Networks☆

Rafael B.R. Lourenço [a,*], S. Sedef Savas [a], Massimo Tornatore [a,b], Biswanath Mukherjee [a]

[a] University of California, Davis, USA
[b] Politecnico di Milano, Italy

ARTICLE INFO

Keywords:
Hierarchical control plane
Resilient switch assignment and controller/orchestrator placement
Robust Transport Software-Defined Networks
Disaster survivability
Post-disaster restoration

ABSTRACT

Software-Defined Networking (SDN) enables the separation of data and control planes. Today, it is in use in several practical networks. Research on SDN is yet to focus on Transport Networks in a significant way. To provide efficient services at low cost for future technologies, such as 5G and beyond, the Transport Network will need to support network functions; adapt the network to different applications' traffic; provide high bandwidth at low latencies; integrate with other networks; etc. By fulfilling these requirements, a Transport Software-Defined Network (T-SDN) can become a fundamental part of the telecommunication infrastructure. However, most current SDN solutions have been developed for Layer 3, and cannot be directly applied to a T-SDN without proper adaptations, as Transport Networks have several characteristics that are different from those of Layer 3 networks. In particular, the design of a T-SDN control plane must address heterogeneity in terms of protocols and administrative network areas; as well as high reliability. A hierarchical control plane is suitable to support these characteristics. Accordingly, in this study, we analyze how to design a robust hierarchical control plane for T-SDNs. We discuss how resiliency against random failures can be provided through redundancy; and how survivability against correlated failures (such as disasters) can be achieved by effectively choosing network nodes where to place control-plane elements, and deciding how to route control-plane traffic. We formulate an Integer Linear Program to design a hierarchical, failure- and disaster-resilient T-SDN control plane. We also propose a heuristic for post-failure switch-controller reassignment. We compare our model with a disaster-unaware control-plane design whose objective is to reduce network-resource utilization. Our results show that we can achieve much higher disaster and failure resiliency, at the cost of slightly larger network-resource utilization.

1. Introduction

Software-Defined Networking (SDN) has gained a lot of popularity in recent years as several practical implementations demonstrated performance improvements and lower costs when compared to traditional networks [2,3]. Throughout different protocols and implementations, the common characteristic of SDN is the decoupling of control and data planes.

The data plane of SDNs (in charge of data transmission) is comprised of switches; while the control plane is composed of controllers and other control elements. This separation gives the control plane a network-wide perspective that makes it easier for resources to be managed in creative ways [4,5]. This enables operators to provide better services (latency, bandwidth, etc.) at lower costs.

So far, most SDN-related systems, standards, and solution proposals have focused on Layer 3 of the communication stack, which does not include much of the Transport Network infrastructure. However, two main factors now motivate the use of SDN in Transport Networks: *i)* operators' thinning profit margins; *ii)* evermore bandwidth-hungry, reliability-dependent, and delay-sensitive applications, such as Virtual Reality, Internet of Things, Ultra-High-Definition video formats, etc. These elements also impact other networks. For example, one tool that the future 5G standard (and beyond) will use to deal with these factors is cooperation between different networks. Such cooperation could also be facilitated by Transport Software-Defined Networks (T-SDN) —

☆ A very preliminary version of a part of this work was presented at the High-Performance Switching and Routing (HPSR) 2015 conference in Budapest [1].

* Corresponding author.
E-mail addresses: rlourenco@ucdavis.edu (R.B.R. Lourenço), ssavas@ucdavis.edu (S.S. Savas), tornator@polimi.it (M. Tornatore), bmukherjee@ucdavis.edu (B. Mukherjee).

https://doi.org/10.1016/j.osn.2018.05.004
Received 31 January 2018; Received in revised form 16 April 2018; Accepted 19 May 2018
Available online 22 May 2018

which is another motivation for the use of SDN in Transport Networks.

However, some characteristics specific to Transport Networks hamper a direct use of much of the already-accrued SDN knowledge without proper adjustments. One of the most distinctive characteristic is the heterogeneity of coexisting protocols, architectures, and systems (sometimes specific to each layer) of a Transport Network. In addition, several network element manufacturers differentiate themselves from their competition by introducing proprietary technologies and management systems, which are usually incompatible with other manufacturer's solutions. All this heterogeneity is common in most current operational Transport Networks, and it is evidenced by each network being composed of several administratively-independent *network areas* [6]. This reality contrasts with Layer-3-based SDN, where elements are usually much more homogeneous and inter-operable.

To deal with such heterogeneity, T-SDN efforts have been focusing on a solution where the control plane is structured hierarchically [7–9]. The lower level of this structure is composed of controllers, each capable of managing elements in a certain network area. Controllers, then, communicate with a centralized entity, named orchestrator, so that the network may provide services across different areas.[1] Such controllers and orchestrators can either be physical machines or virtualized systems [10].

Another particular characteristic of Transport Networks is that they are built to be extremely reliable. Many long-distance operators offer services at five 9s availability (sometimes higher [11]). Such highly-reliable systems must survive not only single and double failures (and more) but also large, correlated failures, such as disasters.

Achieving very high reliability in SDN is very challenging. The decoupling of data and control planes introduced by SDN represents a break in the long-standing fate-sharing principle of control and data planes of traditional network elements: i.e., it is possible for SDN data-forwarding elements to lose contact with control elements, and vice versa. This loss of control might lead network elements to be unpredictable and in unstable states. Thus, a major contingency for the successful utilization of SDN concepts in the availability-demanding Transport Network is the resiliency and the survivability of the SDN control plane and its communication with data-plane elements.

In this study, we propose a method to design a robust, hierarchical control plane for T-SDNs. To deal with heterogeneity, our solution assigns switches of different network areas to respective controllers, decides how many and where to place controllers in each area, and connects them to a central, primary orchestrator (and also backup orchestrators), in a hierarchical architecture. Our method also decides how to route switch-to-controller and controller-to-orchestrator traffic, avoiding shortest paths when more reliable, longer paths are also within latency bounds. To provide high reliability, our solution: *i)* provides redundancy of control-plane elements for failure resiliency (i.e., multiple controllers, backup orchestrators, and paths connecting them); *ii)* efficiently places controllers and orchestrators in the network (while choosing what paths to connect them), to minimize the effects of disasters. Our method provides all of the above while respecting other system-related requirements (such as maximum control traffic latencies, controller capacities, etc.).

The method we propose is mathematically formulated into an Integer Linear Program (ILP) that takes less than a few minutes to execute for a US-wide, 24-node network (running on an Intel Core i7, 16 GB RAM machine). Our model is capable of designing a robust control plane; however, severe, unpredicted failures might still disrupt it. Because of that, we also propose a simple heuristic to instruct how to proceed in case such a disrupting failure occurs. Our results show that our *disaster-aware* control plane is much more reliable than a *disaster-unaware* control plane designed to simply minimize control traffic latency, at the cost of slightly higher resource utilization.

The rest of this work is organized as follows: Section 2 briefly reviews the literature related to SDN control-plane design. Section 3 investigates the characteristic of the T-SDN hierarchical control plane. Section 4 details how we achieve high reliability in T-SDN control plane. Section 5 formally describes our problem, and provides a mathematical formulation and a heuristic in case of failures. Section 6 discusses the lessons learned using practical numerical examples. Section 7 concludes the study.

2. Related work

In this section, we briefly review contributions to the general problem of placing controllers in an SDN.[2] Then, we discuss studies that focus on the relationship between where controllers are placed and network resiliency.[3]

2.1. Controller placement

The Controller Placement Problem (CPP) was first proposed in Ref. [13]. Several metrics (such as worst and average case switch-to-controller latency, and number of switches reachable from each controller) and how they affect the optimal location of controllers in a network were considered in this study. The CPP was further investigated in Ref. [14], when they introduced constraints on the controllers' capacity to manage network elements, calling it the Capacitated CPP (or CCPP).

The placement of controllers in Wide-Area Networks (WAN) was studied in Ref. [15]. The authors proposed to first partition the WAN into different components through a clustering algorithm; and, second, decide where to place controllers for each component. This approach is not suitable for brownfield Transport Networks, as the different areas in these environments have usually evolved organically through the years, and it is very costly to completely re-define them.

In a mobile cloud-computing environment, a two-tier control plane is presented in Ref. [16]. The authors propose a solution where a centralized SDN orchestrator monitors and decides when and where to place local controllers depending on the network state. Other works that study some variation of the CPP include: Ref. [17] which investigated a minimum-cost solution to the CPP while limiting the maximum switch-to-controller latency; methods to dynamically grow/shrink the number of controllers according to the network state were proposed in Refs. [18,19]; practical implementations of distributed control planes were presented in Refs. [20–22].

Although distributing the control plane is a first step towards resiliency in SDNs, as it avoids a single point of failure, the works above have not specifically aimed at providing highly-reliable solutions — the focus of our study.

2.2. Controller placement and network resiliency

The Fault Tolerant Controller Placement Problem was introduced in Ref. [23]. The authors proposed a solution to achieve five-nines reliability of the switch-to-controller communication, with link, switch, and controller failures.

In Ref. [24], the authors proposed an approach that finds controller locations such that multiple possible switch-to-controller paths

[1] Other reasons for a hierarchical control plane include: backwards compatibility with brownfield Transport Networks; higher resiliency due to the distributed control plane; scalability; etc.

[2] In our study, we refer to the dual problem of deciding where to place controllers and what switches to assign to them as a single problem, because these two aspects are intrinsically connected.

[3] We understand *reliability* as the metric which expresses the percentage of the control plane (and its interactions within itself and with switches) that is not lost due to failures. *Resiliency* is the capacity of the control plane to sustain failures while remaining operational (partially or fully) [12].

are possible, minimizing the chances of single-element failures interrupting communication between switches and controllers. In Ref. [25], a method to compute a set of Pareto optimal placements based on different failure cases is presented. Once the set is calculated, the network operator is in charge of deciding the placement it wants. In Ref. [26], the authors study how to design an SDN control plane that is robust against malicious node attacks.

The authors of [27] solve the CCPP while also pre-defining a backup controller for each switch. Thus, if a controller fails, the backup controller starts managing the switches that were initially assigned to the failed controller. The authors aim at minimizing the latency from switch to primary controller plus the latency from the primary to the backup controller. Thus, the proposed method places primary and backup controllers very close together — a solution that is not ideal for geographically-correlated failures, such as disasters.

In a previous version of this work, we studied how to place controllers while accounting for disaster-prone regions in the network [1]. By minimizing the risk of control-plane disruptions, our solution decides where to place controllers, how to define the inter-controller topology, and switch-to-controller connections. Thus, our previous and current works differ from most literature in that we not necessarily use shortest paths to connect switches to controllers. Instead, we find paths such that the impact of disasters on the control plane is minimized, while respecting the maximum allowable switch-to-controller and controller-to-orchestrator latencies.

To the best of our knowledge, works on resilient controller placement have not investigated this problem in a Transport Network environment. As our work focuses on these networks, we study a solution capable of handling heterogeneous environments while providing high reliability. For that, we focus on a robust hierarchical control plane, which has not been considered by the works above. Also, our work is capable of designing a control plane robust not only against random failures but also against disasters, an aspect that has not been explored by the works listed here (except for [1]).

3. T-SDN: the hierarchical control plane

Transport Networks have several characteristics that are not necessarily common to other network domains (e.g., L3) where the SDN paradigm has been implemented so far. The following characteristics have the strongest impact on the control-plane design for T-SDNs [6,7]:

1. *Heterogeneity:* Several diverse technologies and architectures (such as SONET/SDH, OTN, etc.) exist side-by-side in Transport Networks. These technologies may be common to geographical regions of the network, and/or to services with more or less stringent bandwidth, latency, or availability requirements. Also, manufacturers have historically differentiated their products by introducing proprietary technologies and by specializing their management systems. The result of all these factors is that, in most practical Transport Networks, several *areas*[4] of the network are administratively isolated from each other and cannot be all directly managed by a single common entity.
2. *High-Reliability:* As Transport Networks commonly connect several densely-populated areas and/or in general high-bandwidth infrastructures (such as Data Centers), a link or node failure can be extremely harmful. Single or even double element failure preparedness is sometimes not sufficient to fulfill some *Service Level Agreements*, which may require five 9s of availability or higher. In fact, it is important for these networks to be able to survive even unlikely events such as natural or human-made disasters [28,29].

The above two factors motivated us to focus on a hierarchical control plane. Motivations for a hierarchical structure include those of a generic distributed control plane (such as lower control traffic latency, avoiding computational bottlenecks, increasing resiliency, etc.); as well as others specific to a hierarchical system, such as: scalability [30]; backwards compatibility with current Transport Networks [6,7]; and current efforts (mainly by ONF [8] and OIF [9]) into standardization of T-SDNs.

Our model assumes that the hierarchical control plane is structured as follows. First, each network element belongs to some *network area*. All elements of a certain *area* are managed (e.g., through the controllers' Southbound Interfaces [6,7]) by one or more controllers specific to elements of that *area*. Controllers are then assigned to a centralized orchestrator (e.g., through their Northbound Interfaces [6,7]). Fig. 1a shows the logical structure of a hierarchical control plane.

The SDN orchestrator is an important part of the control infrastructure as it allows for different areas of the network to operate together. Each controller is responsible for communicating with the orchestrator by providing it with information of its managed portion of the network. A possible example of how this system works is as follows: if a connection request requires traversing multiple areas, it is the job of the orchestrator to decide what should be the appropriate route for such a connection; and to inform (or negotiate with) the different controllers to provision the necessary segments of the connection's path. For that, the orchestrator needs to maintain an updated view of the entire network (which may be a simplified abstraction).

Given the premise of a hierarchical control plane (as described above) which must be made as robust against failures as possible, we focus on optimally deciding:

i. where to place controllers and how to assign switches;
ii. where to place orchestrators; and
iii. how to route switch-to-controller and controller-to-orchestrator traffic.

A major concern when placing controllers in an SDN is the latency which traffic between switches and controllers are subject to [13,14,27]. As this latency directly impacts the performance of the network, there are stringent requirements that must be observed when deciding where to place controllers and how to route switch-to-controller traffic. Given that the orchestrator is a fundamental entity in a hierarchical control plane, it is important that traffic between orchestrators and controller also take into consideration latency. However, orchestration of different networks (or network areas) is an active research topic [31,32], and bounds on the latency of controller-to-orchestrator traffic are still not defined (to the best of our knowledge). For this reason, in Section 5, we formulate our problem such that switch-to-controller traffic must be subject to a certain latency bound, while controller-to-orchestrator traffic is subject to another (these bounds can also be made equal in our formulation).

In the example of Fig. 1a, a possible solution for the control-plane placement is shown in Fig. 1b. This placement is a disaster-unaware placement and choses controller locations and paths such as to minimize resource utilization (in terms of number of links used for control traffic). Note that such an approach tends to concentrate many control-plane elements towards the center of the network topology (i.e., close to nodes 9 and 12).

4. Resiliency and the hierarchical control plane

As discussed before, in traditional non-SDN networks, the fate of control and data planes of an element are intrinsically tied together: if one fails, the other fails as well. With SDN, this fate sharing is not necessarily true. Thus, a major contingency for the success of the Transport-SDN is the robustness against failures of the control plane, and the paths through which it interacts within itself and with data-plane elements.

In a general SDN, in case a link carrying switch-to-controller traffic fails, it is necessary to re-route such traffic through a different path, either by pre-provisioning a protection path (protection) or by utilizing

[4] In Ref. [6], *domain* is the equivalent of a *network area*.

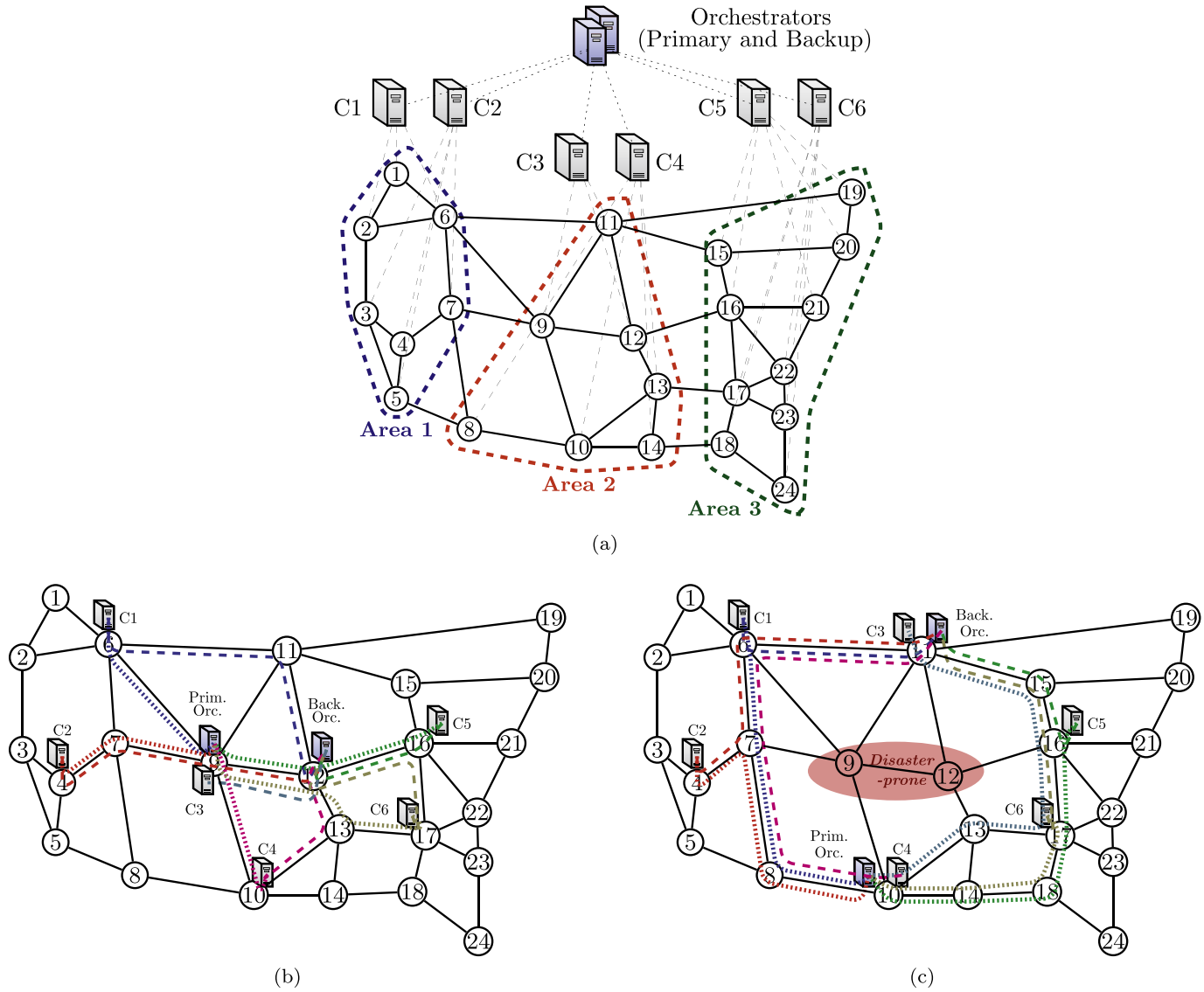


Fig. 1. (a) Logical topology of a hierarchical control plane of a T-SDN with three *network areas*. In this example, each area has two controllers. All controllers communicate with the primary and backup orchestrators simultaneously. Dashed black lines represent the controller's communication with switches. Dotted black lines represent the controller's interactions with orchestrators. (b) A possible disaster-unaware control-plane placement (note that switch-to-orchestrator traffic is omitted). Dotted paths connect controllers to the Primary Orchestrator. Dashed paths connect controllers to the Backup Orchestrator. (c) A disaster-aware control-plane placement. If nodes 9 and 12 are within a disaster-prone region, it is important to avoid them when placing control-plane elements and when routing control traffic.

some re-routing strategy (restoration). The same is true if a node in the path of the control traffic fails. *Observation 1: Note, however, that a backup path might not respect the latency constraints that were considered when initially solving the controller-placement problem.*

In terms of node failures, another possible scenario is if the node hosting the controller itself fails (or the node to which the controller is attached). In this scenario, if the control plane is distributed, it is necessary to reassign those switches to other controllers in the network. Thus, a distributed control plane is a first safeguard against random node failures. *Observation 2: In such case, however, two possible undesired situations might happen:*

i. a switch being reassigned to a controller through a path that is above latency constraints; or
ii. a switch being reassigned to a controller that is already managing the maximum number of switches that it can handle (i.e., the controller is already operating at maximum capacity).

As explained, failure resiliency can be achieved through redundancy. However, to remedy the observations listed above, proper considerations during the placement of the control plane can help [23,27]. In fact, the placement of the control plane can also help lessen the impact of large-scale, correlated, predictable failures (such as natural and man-made disasters) [1]. This is important since disasters can disrupt large portions of the network, and redundancy that would be enough for uncorrelated failure resiliency might not be sufficient to provide disaster-survivable services [28,29].

In the Transport-SDN, however, the hierarchical control plane creates a second level of interaction that must be considered when designing a robust control plane (i.e., controller-to-orchestrator). Thus, the observations above are true not only for the switch-to-controller but also for the controller-to-orchestrator levels of the hierarchy. In addition, the T-SDN control plane must be resilient not only to random, uncorrelated failures, but also to large-scale disasters. Thus, in Section 5, we propose a method to design and place the T-SDN hierarchical control plane such that it provides redundancy against uncorrelated failures, as well as disaster survivability, while considering the observations listed in this section. In Fig. 1c, the same control-plane hierarchy as that of Fig. 1a is placed in a disaster-aware manner. In this example,

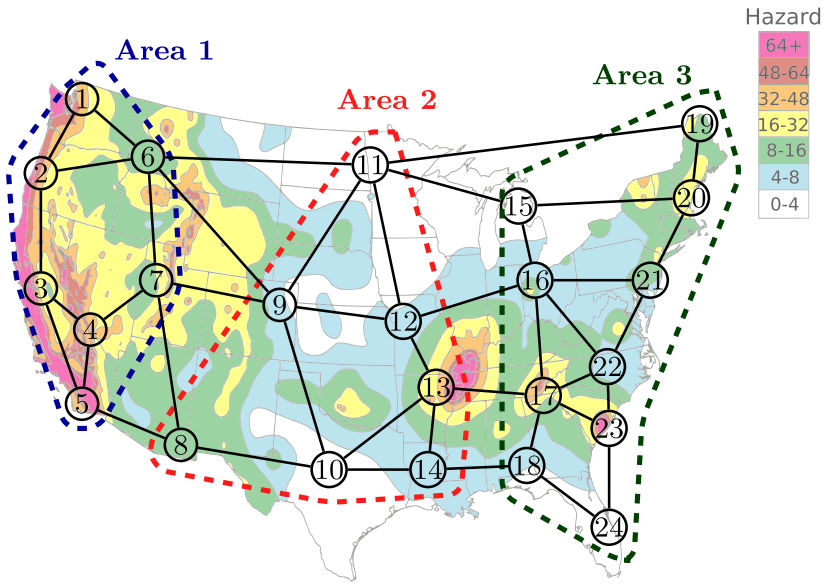


Fig. 2. A 24-node network topology over the US seismic hazard map [36]. Colored regions of the map represent possible disasters and their color-coded hazards is proportional to the *disaster risks*, as defined in Section 4.1. Note that the network has three *network areas*. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

since nodes 9 and 12 are within a disaster-prone region, these nodes should be avoided when placing control-plane elements and when routing control-plane traffic. Note how such disaster-aware control-plane placement differs from the disaster-unaware placement of Fig. 1b.

Before going into detail about our formulation, we first need to investigate how to measure the impact of disasters in the T-SDN infrastructure.

4.1. Disaster risk assessment

To make the control plane resilient to disasters, it is necessary to first determine the vulnerability of the network to them. We consider that the network operator is capable of assessing what elements might be affected by each disaster (e.g., a map of the disaster-affected regions such as the one in Fig. 2). Also, we consider that the network operator is capable of computing a probabilistic metric called *disaster risk* [33]. This metric encompasses the probability of a disaster occurring in the observation period of interest to the operator; and, given that such a disaster occurs, the probability that it will cause damage to the network elements within the affected region.

For the set of all disasters that may occur, we compute a metric called *Disaster Impact*. This metric captures the expected loss of control-plane connectivity (i.e., the links carrying switch-to-controller and controller-to-orchestrator traffic) due to disasters weighted by the risk of each disaster. It can be calculated as:

$$I = \sum_{y \in Y} \sum_{p \in P^c} R_y \cdot \left(1 - U_p^y\right)$$

where Y is the set of all disasters y, P^c is the set of all paths through which control traffic is carried, R_y is the risk of disaster y, U_p^y is 1 if path p is not affected by disaster y, and I is the impact of disasters. Note that node failures can also be accounted for by considering all paths that traverse a node.

5. Robust hierarchical control-plane design

With the considerations discussed in the previous sections, we now formalize the problem in Section 5.1 and mathematically formulate it into an ILP in Section 5.2. Our method is able to design a robust control plane; however, severe failures might still disrupt it. To overcome the effects of these failures, in Section 5.3, we also present a post-failure/disaster dynamic procedure that complements the safeguard measures of our mathematical formulation.

5.1. Problem statement

Given: Network topology; network nodes capable of hosting SDN controllers and/or orchestrators; for each network node, the network area it belongs to; disaster-prone regions and their risks; maximum switch-to-controller and controller-to-orchestrator latencies (possibly different); how many switches each controller can manage (also called *controller capacity*); pre-calculated paths for each pair of nodes; number of orchestrators to be deployed in the network.

Output: How many controllers and where to place them; where to place orchestrator(s); what switches to assign to each controller; and how to route switch-to-controller, and controller-to-orchestrator connections.

Objective: First: Minimize the risk of control-plane disruption due to disasters. Second: Provide the a solution with minimal resource utilization (in terms of number of controllers and links utilized to route control traffic).

5.2. Mathematical formulation

Now, we present an ILP formulation to be used offline when deciding where to place the T-SDN hierarchical control plane, i.e., to solve the problem presented in Section 5.1. Once the placement of control plane is done, in Section 5.3, we present a heuristic to dynamically overcome failures if they occur.

We refer to our model as Disaster-Aware Hierarchical Control-Plane Placement. In Section 6, we will compare our method to a *Disaster-Unaware* solution. The constraints of the Disaster-Unaware solution are the same as those of the Disaster-Aware case (which will be presented in the following sections), except for constraints 8 and 9 and 19–27 (the objective function of the Disaster-Unaware model is only equation λ).

Table 1
Inputs.

Input	Description
$G(N_t, E)$	Network topology, where N_t is the set of nodes with area information, and E is the set of directed links.
F	Set of nodes capable of hosting controllers and/or orchestrators. We consider that each node in this set can host as many orchestrators and/or controllers as necessary.
$T = \{t \mid t \in \{0,1,2,\ldots\}\}$	Set of areas in the network.
$D_{ij} \in \{0,1\}$	1 if node i is within reach of node j, under maximum switch-to-controller latency.
$L_{ij} \in \{0,1\}$	1 if node i is within reach of node j, under maximum controller-to-orchestrator latency.
P_{ij}	Set of possible paths from i to j within latency limits.
$Y = \{y \mid y = < E_y, R_y >\}$	Set of disasters, E_y links they affect, and the risk R_y of y occurring.
$U_p^y \in \{0,1\}$	1 if path p survives disaster Y.
$B^c \in \{1,2,3,\ldots\}$	How many switches each controller can manage.
$B^o \in \{1,2,3,\ldots\}$	How many controllers an orchestrator can manage.
$K \in \{1,2,3,\ldots\}$	Minimum number of controllers that must be reachable from any switch within latency constraints (as defined by D_{ij}). We suggest using a number larger than 1, as otherwise there might not be a reachable backup controller for a switch if its assigned controller fails.
$N \in \{1,2,3,\ldots\}$	Minimum number of paths between a switch and its controller. We suggest a number larger than 1, as otherwise there might not be a backup path in case a failure happens.
$Q \in \{1,2,3,\ldots\}$	Minimum number of paths between a controller and the orchestrators.
$D \in \{0,1,2,\ldots\}$	Number of backup orchestrators that should be deployed.
$V_i^y \in \{0,1\}$	1 if node i survives disaster y. If a disaster completely isolates a node from the network (even though it may remain working), we consider that such a node does not survive the disaster.

5.2.1. Input parameters

The input parameters are summarized in Table 1.

5.2.2. Variables

All variables used in this formulation are binary. They are summarized in Table 2.

5.2.3. Objective

The objective of our formulation is stated as:

$$minimize\,(\omega + \delta + \lambda)$$

where

$$\omega = \sum_{y\in Y}\sum_{i\in N}\sum_{f\in F}\sum_{p\in P_{if}} R_y \cdot \left(1 - U_p^y\right) \cdot \left(\Omega \cdot w_{if}^p + \sum_{t\in T} \Gamma \cdot h_{if}^{tp}\right)$$

is the average number of controller-to-orchestrator and switch-to-controller paths that fail due to disasters, weighted by the risk of each disaster occurring (note that this equation is, in fact, the Disaster Impact metric of Section 4.1);

$$\delta = \Delta \cdot \left(\sum_{i\in F} j_i + \sum_{y\in Y}\sum_{i\in F} R_y \cdot (1 - V_i^y) \cdot j_i\right)$$

is the total number of nodes that host controllers (for all areas in the network) plus the average number of controllers that fail due to disasters, weighted by the risk of each disaster occurring. This term ensures that not only the least amount of controllers are deployed, but also that the deployed set of controllers is the one least impacted by disasters. And

$$\lambda = \Lambda \cdot \sum_{i\in N}\sum_{j\in N}\sum_{p\in P_{ij}} len(p) \cdot \left(w_{if}^p + \sum_{t\in T} h_{if}^{tp}\right)$$

is the resource utilization measured by the number of links used for switch-to-controller and controller-to-orchestrator communication ($len(p)$ is the length of path p).

In the functions above Ω, Γ, Δ, and Λ are sufficiently large constants whose values may affect the overall solution of the problem. In this study, we first focus on making the controller-to-orchestrator communication disaster-resilient (thus, $\Omega \gg \Gamma$); then, the switch-to-controller (thus, $\Gamma \gg \Delta$); and, finally, on first minimizing number of controllers and then the number of links used for control traffic (thus, $\Delta \gg \Lambda$).

5.2.4. Constraints

In constraints 2, 12, 18, and 22–26, M is a sufficiently large number.

Controller Placement and Switch Assignment: We start by defining the constraints that define the switch-to-controller level of the hierarchical control plane:

$$c_f^t \le \sum_{i\in N} a_{if}^t, \qquad \forall f \in F, \forall t \in T \tag{1}$$

$$c_f^t \ge \frac{\sum_{i\in N} a_{if}^t}{M}, \qquad \forall f \in F, \forall t \in T, M\ large \tag{2}$$

$$\sum_{i\in N} a_{if}^t = 1, \qquad \forall f \in F, \forall t \in T \tag{3}$$

Table 2
Binary variables.

Variable	Description
c_f^t	1 if one controller of area t is located in node f (also, if nodes f is not of area t, then $c_f^t = 0$). Note that each area might have multiple controllers.
a_{if}^t	1 if switch i is assigned to controller f (also, if nodes i or f are not of type t, then $a_{if}^t = 0$).
j_f	1 if any controller is active in node f.
h_{if}^{tp}	1 if path p is used for communication between switch i and controller f, both of area t.
o_f^t	1 if orchestrator is located in node f.
b_{if}^t	1 if all controller in i are assigned to orchestrator f.
w_{if}^p	1 if path p is used for communication between controller i and orchestrator f.
z_{if}^y	1 if at least one path from controller i to orchestrator f survives disaster y.
s_f^y	1 if all controllers that survive disaster y remain connected to orchestrator f.

$$a_{if}^t \le d_{if}, \qquad \forall i \in N, f \in F, \forall t \in T \tag{4}$$

$$a_{if}^t = c_f^t, \qquad \forall i, f \in N, \forall t \in T, i = f \tag{5}$$

$$a_{if}^t + c_i^t \le 0, \qquad \forall i, f \in N, \forall t \in T, i \ne f \tag{6}$$

$$a_{if}^t \le c_f^t, \qquad \forall i, f \in N, \forall t \in T, i \ne f \tag{7}$$

Constraints 1 and 2 enforce that a controller will be deployed in node f if at least one node i of area t is set to be controlled by such a controller. Constraint 3 enforces that every node must be controlled by exactly one primary controller. Constraint 4 enforces that nodes can only be assigned to a controller reachable within their latency limit. Constraints 5–7 enforce that, if there is a controller in some node, the switch in such node must be assigned to the controller in that node (i.e., to the *local* controller); otherwise, this switch must be assigned to a controller in some other node.

Controller Failure Resiliency: Resiliency against uncorrelated failures of the switch-to-controller level of the hierarchy is enforced by the following constraints:

$$\sum_{i \in N} a_{if}^t + 1 \le \frac{B^c}{1 + \frac{1}{K}}, \qquad \forall f \in F, \forall t \in T \tag{8}$$

$$\sum_{p \in P_{if}} h_{if}^{tp} \ge N \cdot a_{if}^t, \qquad \forall i, f \in N, \forall t \in T, i \ne f \tag{9}$$

$$\sum_{f \in F} c_f^t \cdot D_{if} \le K, \qquad \forall i \in N, \forall t \in T \tag{10}$$

Constraint 8 enforces that every controller will have spare capacity, thus allowing it to accommodate extra switches if a neighboring controller fails. This is a simplification that considers controllers uniformly placed across the network, such that, if one controller fails, the switches assigned to it will be reassigned to one of its K closest neighboring controllers. We argue that the maximum capacity of a controller is a soft limit that can be infringed for brief amounts of time, if necessary [34]. In fact, if a controller ends up managing more switches than its capacity B^c, the matter should be resolved quickly since it would be due to a serious network failure or a short-term traffic burst.

Constraint 9 enforces that at least N paths must be setup between each node and its controller. Generally, N should be a small number greater than one. This helps to avoid disconnections due to single failures in a path carrying switch-to-controller traffic. Our tests show that, since multiple controllers are placed close to each switch, link or node disjointedness among these paths only provides marginal resiliency benefits (if any) at very high computational costs. Thus, we do not enforce path disjointedness but a sub-optimal solution might be achieved if the user provides a set P_{if} of only disjoint paths.

Constraint 10 enforces that at least K controllers will be reachable from any node, within latency bounds, such that, if a controller fails, the switches it was controlling can fall-back to other close-by controllers. For implementation, the fall-back procedure can be encoded in each switch by setting the initial controller as master, and the other controllers in the region as slaves (ordered by their distance to the switch) [27].

Orchestrator Placement: The next set of constraints refers to the controller-to-orchestrator level of the T-SDN control plane hierarchy:

$$o_f \le \sum_{i \in N} b_{if}, \qquad \forall f \in F \tag{11}$$

$$o_f \ge \frac{\sum_{i \in N} b_{if}}{M}, \qquad \forall f \in F, \forall t \in T \tag{12}$$

$$b_{if} \le L_{if}, \qquad \forall i, f \in N \tag{13}$$

$$b_{if} \le o_f, \qquad \forall i, f \in F, \forall t \in T \tag{14}$$

$$b_{if} \le j_i, \qquad \forall i, f \in F, \forall t \in T \tag{15}$$

$$\sum_{f \in F} \sum_{t \in T} c_f^t \le B^o \tag{16}$$

$$j_i \le \sum_{t \in T} c_i^t, \qquad \forall i \in N \tag{17}$$

$$j_i \ge \frac{\sum_{t \in T} c_i^t}{M}, \qquad \forall f \in F, \forall t \in T \tag{18}$$

Constraints 11 and 12 enforce that an orchestrator will be deployed in node f if at least one controller i (of any area) is set to be managed by such an orchestrator. Constraint 13 enforces that all orchestrators must be within the latency limits of every controller. Constraints 14 and 15 enforce that controllers must be managed by nodes that host orchestrators (and vice versa). Since every controller must be assigned to all orchestrators, constraint 16 enforces that no more than B^o controllers be deployed in the network. Research into abstracting network details to simplify the orchestrator responsibilities [31] suggest that the maximum number of controllers that an orchestrator can manage (i.e., B^o) could be very large. Constraints 17 and 18 enforce that variable j_i is one if at least one controller of any area t is placed on node i.

Orchestrator Failure Resiliency: In the controller-to-orchestrator level of the hierarchy, resiliency against random element failures is enforced by the following constraints:

$$\sum_{p \in P_{if}} w_{if}^p \ge Q \cdot b_{if}, \qquad \forall i, f \in F \tag{19}$$

$$\sum_{f \in F} b_{if} = (1 + D) \cdot j_i, \qquad \forall i \in N \tag{20}$$

Constraint 19 enforces that at least Q paths must be setup between each controller and each orchestrator, similar to traditional protection against element failures (when $Q > 1$). This protects the controller-to-orchestrator traffic from single failures. Similar to the observation of constraint 9, we do not enforce disjointedness among these paths because multiple orchestrators are deployed.

Constraint 20 enforces that every controller must be managed by exactly $1 + D$ (primary and backup) orchestrator(s). Typically, D should be a small number, such as one or two. The redundancy enforced by this constraint is important because, in the T-SDN hierarchical control plane, failures that cause controllers to lose contact with the orchestrator can be critical, as they might render the different networks areas unable to work with one another. As all controllers must communicate with a central orchestrator, this element still constitutes a single point of failure. To avoid this, we consider deploying one primary and, possibly, multiple backup orchestrators, as shown in constraint 20. All orchestrators (primary and backups) must maintain a synchronized, coherent network view at all times. We assume that every controller sends information to all orchestrators simultaneously, and primary-to-backup orchestrator communication to propagate or synchronize network information is not needed.

We assume that every orchestrator, then, contains a copy of the same information, and whenever the operator changes configuration in one of them, it also changes in the others. These orchestrators are effectively a type of content-replication system across the network. The main benefits of this scheme are: *i*) just a few orchestrators are enough to provide high reliability (a small price for the benefit), as will be shown; *ii*) in case of orchestrator failure, the controllers can instantaneously fall back to the backup orchestrator, and it will already be knowledgeable of the updated network state and history. In fact, content replication has been studied in the literature [35] and is an enabler for future,

highly-reliable services, such as 5G and beyond. Note that the assumption that all orchestrators simultaneously receive information from all controllers is not a requirement if there exists a method to synchronize different orchestrators.

Orchestrator Disaster Resiliency: In terms of the controller-to-orchestrator relationship, when a disaster happens, the orchestrator placement must be such that at least one orchestrator survives such disaster, for any disaster. This is yet another reason for the deployment of multiple orchestrators. The following constraints enforce that the orchestrators will be placed such that at least one will survive any given disaster:

$$z_{if}^{y} \leq V_i^{y} \cdot o_f, \qquad \forall i,f \in N, \forall y \in Y \tag{21}$$

$$z_{if}^{y} \leq V_i^{y} \cdot j_i, \qquad \forall i,f \in N, \forall y \in Y \tag{22}$$

$$z_{if}^{y} \leq \sum_{p \in P_{if}} w_{if}^{p} \cdot U_p^{y}, \qquad \forall i,f \in F, \forall y \in Y \tag{23}$$

$$z_{if}^{y} \geq \frac{\sum_{p \in P_{if}} w_{if}^{p} \cdot U_p^{y}}{M}, \qquad \forall i,f \in F, \forall y \in Y \tag{24}$$

$$\sum_{i \in N} z_{if}^{y} \geq \sum_{i \in N} V_i^{y} \cdot j_i - M\left(1 - s_f^{y}\right), \qquad \forall f \in N, \forall y \in Y \tag{25}$$

$$\sum_{i \in N} V_i^{y} \cdot j_i \geq \sum_{i \in N} z_{if}^{y} - M \cdot s_f^{y} + 1, \qquad \forall f \in N, \forall y \in Y \tag{26}$$

$$\sum_{f \in F} s_f^{y} \geq 1, \qquad \forall y \in Y \tag{27}$$

As the orchestrator needs to be knowledgeable of the entire network state, these constraints enforce that, not only at least one orchestrator must survive any one disaster, but also that at least one surviving orchestrator will remain connected to every surviving controller. Constraints 21–24 enforce that z_{if}^{y} is one if at least one controller-to-orchestrator path survives disaster y (i.e., if both controller and orchestrator also survive such disaster).

In constraints 25 and 26, note that $\sum_{i \in N} z_{if}^{y}$ is the total amount of controller-hosting nodes that survive disaster y and remain connected to orchestrator f; and $\sum_{i \in N} V_i^{y} \cdot j_i$ is the total amount of controller-hosting nodes that survive disaster y. Thus, the second is an upper bound of the first. These constraints enforce that s_f^{y} is one only if all controllers that survive disaster y remain connected to orchestrator f. Constraint 27 enforces that, after any disaster y, all surviving controllers remain connected to at least one surviving orchestrator. If more than one orchestrator survives, at least one of them will remain connected to every surviving controller.

5.3. Post-disaster and post-failure procedures

The method in Section 5.2 can be used to statically decide how to place the hierarchical control plane in a robust manner. However, severe failures might still affect the normal operation of the T-SDN control plane. Also, our formulation only guarantees that a second controller will be within reach, without exactly pre-determining a path between switches and the backup controller. In this section, we explore heuristics to overcome control-plane disruptions in case a severe failure affects either switch-to-controller or controller-to-orchestrator interactions.

5.3.1. Surviving switch-to-controller failures

In case a failure either in the path that carries switch-to-controller traffic or a controller failure occurs, we propose the following approach:

A. *Link Failure:* in case of failure in the primary path between a switch and its assigned controller:
   (a) *if* $N > 1$: try the shortest pre-computed path (as per constraint 9). If the path is working, use it (it becomes the new primary path); if not, try the next, until all pre-computed paths have been examined;
   (b) *if* $N = 1$ *or all N paths have been exhausted:* use Algorithm 1.

B. *Controller failure:* use Algorithm 1.

5.3.2. Surviving controller-to-orchestrator failures

The controller-to-orchestrator relationship is slightly different than that of the switch-to-orchestrator. In case a failure that affects the controller-to-orchestrator level of the hierarchical control plane occurs, we propose:

A. *Link Failure:* in case of failure in the primary path between a controller and an orchestrator:
   (a) *if* $Q > 1$: try the shortest pre-computed path (as per constraint 19). If the path is working, use it (it becomes the new primary path); if not, try the next, until all pre-computed paths have been examined;
   (b) *if* $Q = 1$ *or all Q paths have been exhausted:* Find the shortest path to that orchestrator.

Note that, if $D > 0$, some other orchestrator will become primary while a new path is not found and provisioned. If $D = 0$, the controller that no longer can communicate with the orchestrator will face difficulties in cooperating with other controllers.

B. *Orchestrator failure:* If $D > 0$, another orchestrator will become primary.

As the procedure above demonstrates, not deploying backup orchestrators can leave controllers without contact to the orchestrator in case of failures. Thus, **we recommend using** $D > 0$.

6. Illustrative examples

In this section, we will analyze the control-plane placement solutions found through our method (i.e., *Disaster-Aware*). We will also compare them to the control-plane placement whose objective does not take into consideration disaster resiliency, only focusing on minimizing resource utilization (i.e., *Disaster-Unaware*, briefly described at the beginning of Section 5). We will demonstrate how our solution is capable of providing high disaster and failure resiliency at the cost of slightly-larger resource consumption and control-plane path latencies when compared to the Disaster-Unaware placement.

In both disaster-aware and unaware approaches, we enforce that every switch will have at least a second controller within reach, in case the first fails. As each area needs at least two controllers, the minimum number of controllers is six in our example, which has three areas (see Fig. 2).

To evaluate our solution's resiliency against disasters, we consider the topology and earthquake hazard map presented in Fig. 2. Although we are using this map for our results, risk maps for different types of disasters (other than seismic) can also serve as input. Nodes and links are susceptible to damage according to the disaster zones which they are located in or which they traverse. We consider that each disaster region has a different disaster risk, proportional to the hazard information from the figure.

The first results, in Sections 6.1, 6.2, and 6.3, analyze how the Disaster-Aware and Disaster-Unaware solutions behave under different latency constraints, both for switch-to-controller and controller-to-orchestrator traffic. For these results, we use the configuration described in Table 3.

Our proposed ILP was solved using CPLEX [37]. Other calculations, algorithms, and simulations were implemented in Java. Results were generated on an Intel Core i7, 16 GB RAM machine. Millisecond laten-

Algorithm 1 Emergency Switch Reassignment.
Input: switch n_i^t and current network topology
Output: what controller c_j^t to assign n_i^t through path $p_i j$
1: Create set C^t of surviving controllers of type t
2: Create empty set S^t of tuples $< c_j^t; p_{ij}; usedCapacityOf(c_j^t) >$
3: **for all** $c_j^t \in C^t$ **do**
4: Find shortest path p_{ij} from n_i^t to c_f^t
5: Insert tuple $< c_j^t; p_{ij}; usedCapacityOf(c_j^t) >$ in S^t
6: **end for**
7: Create set $S^t_{good} \subseteq S^t$ of tuples with p_{ij} within latency bounds, and $usedCapacityOf(c_j^t) < B$
8: Sort S^t_{good} first by length of p_{ij}, then by $usedCapacityOf(c_j^t)$
9: **if** $S^t_{good} \neq \emptyset$ **then**
10: **return** First tuple in ordered S^t_{good}
11: **else**
12: Sort S^t length of p_{ij} multiplied by $usedCapacityOf(c_j^t)$
13: **return** First tuple in ordered S^t
14: **end if**

Table 3
Illustrative example inputs.

Input	Value
Network topology $G(N_r, E)$	See Fig. 2.
Controller-capable nodes F	All nodes in Fig. 2.
Network areas T	3 areas.
Disasters Y	See Fig. 2.
Controller capacity B^c	8 switches.
Orchestrator capacity B^o	Unlimited.
Reachable controllers K	2
Switch-to-controller paths N	1
Controller-to-orchestrator paths Q	1
Number of orchestrators D	2

cies were calculated with the approximation:

$$latency(l) = 0.005 \cdot l$$

where l is the length of the path in kilometers.

6.1. Disaster impact

In Fig. 3, we show how our *Disaster-Aware* solution compares to the *Disaster-Unaware* solution with regards to the Disaster Impact metric introduced in Section 4.1. As an example, given two alternative control-plane designs for the network of Fig. 2 (with the same number of switch-to-controller and controller-to-orchestrator paths), a control plane whose Disaster Impact is 8 has paths that traverse areas twice as risky as those of a control plane whose Disaster Impact is 4, or traverse twice the amount of disaster-prone regions of similar risk. In Fig. 3a, we vary the maximum possible switch-to-controller latency (input D_{ij}) while not imposing limits to the controller-to-orchestrator latency. In Fig. 3b, we do the opposite.

Our results show that our method provides a solution whose Disaster Impact is much smaller than that of the Disaster-Unaware model, in all scenarios. Note, however, that the lengths of switch-to-controller and controller-to-orchestrator paths directly affect the placement solution. As these paths are allowed to be longer, the disaster-aware solution is able to find better paths, allowing the Disaster Impact to decrease. As expected, the disaster-unaware solution does not have a clear relationship with this metric. Note that the red bars (which represent the controller-to-orchestrator paths in the disaster-aware model) are always smaller as the latency limits increase, which is not necessarily true for the blue bars (i.e., the switch-to-controller paths). That is because, in our optimization objective, we place higher importance on the paths that connect controllers to orchestrators.

6.2. Path latencies

In Fig. 4, we show how the disaster-aware average control-plane path latencies compare to the disaster-unaware control-plane path latencies. As the disaster-unaware model simply minimizes network-resource consumption, this solution will always find the shorter paths possible that satisfy this model's constraints. In Fig. 4a, we vary the maximum possible switch-to-controller latency (input D_{ij}) while not imposing limits to the controller-to-orchestrator latency. In Fig. 4b, we do the opposite.

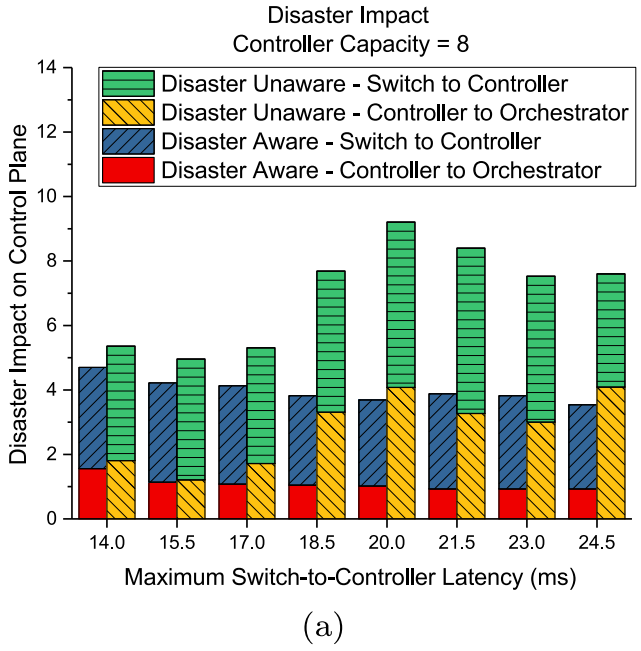
Fig. 4 shows how our disaster-aware solution's paths tend to be slightly longer than the disaster-unaware solution's. Note that our solution is more affected by constraints in the lengths of controller-to-orchestrator paths — which causes it to compensate with longer switch-to-controller paths, as demonstrated in the longer average switch-to-controller path latencies of Fig. 4b when compared to those of Fig. 4a. Also, note that in Fig. 4a, our solution achieves almost the same summed average path lengths as the disaster-unaware approach for a maximum switch-to-controller path latency of 20 ms. This is because, with this particular latency limit, the model cannot find another solution that makes the Disaster Impact metric smaller; and, thus, simply minimizes path latencies as much as possible, which ends up coinciding (the summed average, as the individual switch-to-controller and controller-to-orchestrator averages differ) with the disaster-unaware solution.

6.3. Controllers and their locations

In this section, we analyze how the number of controllers placed is affected by different maximum allowable latencies between controller and switch, and controller and orchestrator. Our results are presented in Fig. 5, and they show that although the number of controllers deployed tends to be similar for both disaster-aware and unaware models (due to constraint 9), the disaster-aware solution places less controllers within disaster-prone zones of the network. In the disaster-aware approach, the number of controllers placed inside disaster-prone regions also decreases as the switch-to-controller and controller-to-orchestrator maximum latencies increases. The disaster-unaware solution, however, does not try to avoid disaster-prone regions.

6.4. Post-disaster switch-controller reassignment

We now analyze how the heuristics proposed in Section 5.3 perform. Our mathematical formulation enforces that at least K controllers will be reachable from any switch within latency bounds. Note that the network topology might change such that a controller that was initially reachable within latency bounds cannot be reached anymore if a failure



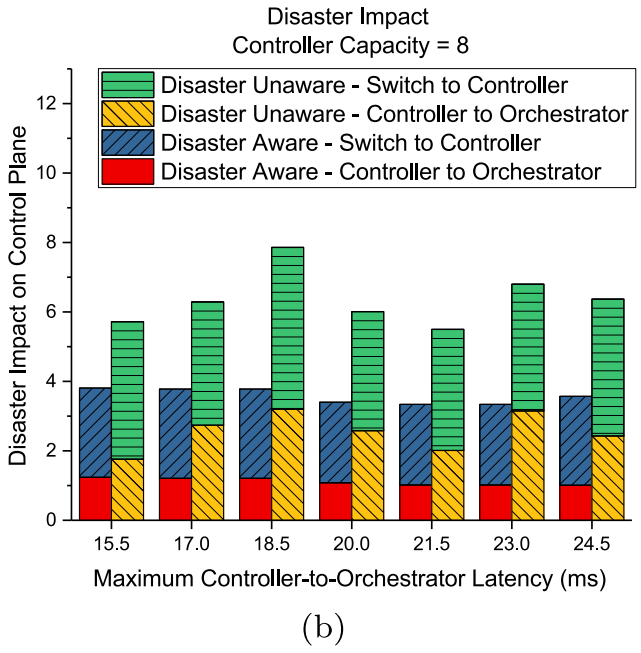
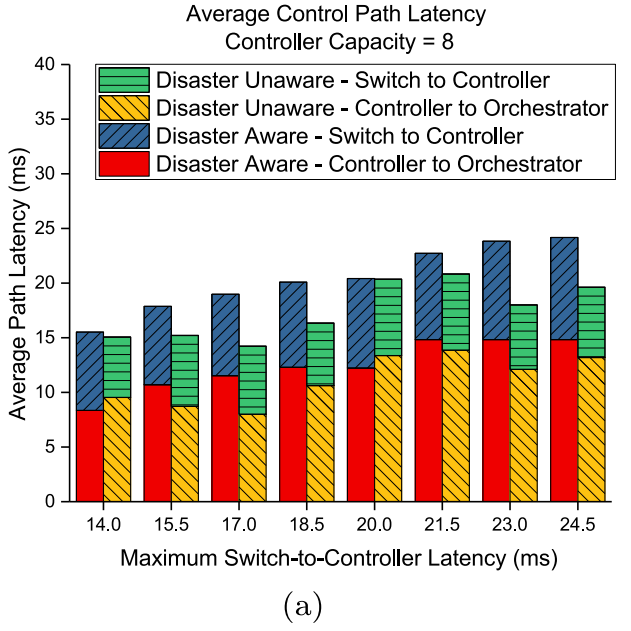


Fig. 3. Comparison of the Disaster Impact of a disaster-unaware solution with our disaster-aware proposal. (a) Shows how the maximum switch-to-controller latency affects the Disaster Impact on the control plane. (b) Shows how the maximum controller-to-orchestrator latency affects the Disaster Impact on the control plane.



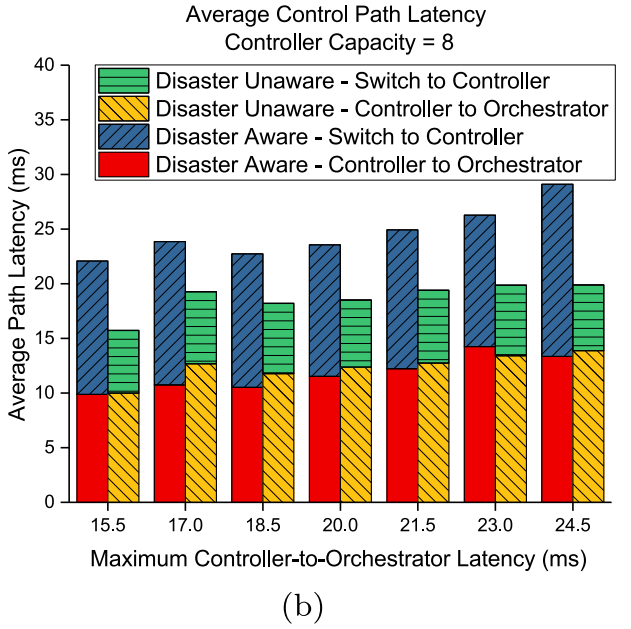


Fig. 4. Average control-plane latencies. (a) Shows how the maximum switch-to-controller latency affects the average control-plane latencies. (b) Shows how the maximum controller-to-orchestrator latency affects the average control-plane latencies. Note that each column is composed of the switch-to-controller and controller-to-orchestrator average latencies; and, in each graph, the maximum latencies only refer to either one or the other (not the sum).

occurs. However, in all of our tested post-disaster scenarios, the paths used for the new switch-to-controller assignment rarely infringed the maximum latency allowed for switch-to-controller communication.

In this result, we focus on how the post-disaster reassignment of a switch to other controllers affected the controller utilization (instead of post-disaster path length, as it never crosses latency bounds). This is because, when controllers fail, the remaining controllers need to manage all the surviving switches. This might change the number of switches which each controller was initially managing (either lowering or increasing the controller capacity utilization).

In Fig. 6, we show the cumulative distribution of controllers at different capacity utilizations, before and after disasters. The capacity utilization of a controller is directly proportional to the number of switches assigned to it. These results reflect how many controllers are there at each capacity in the network. We only consider controllers with capacity of 8.

In Fig. 6, we compare the disaster-aware solution when no disaster occurs with the disaster-aware solution after each disaster presented in Fig. 2, averaged across all disasters. After each disaster, we utilize the post-disaster procedures discussed in Section 5.3 to reassign switches to

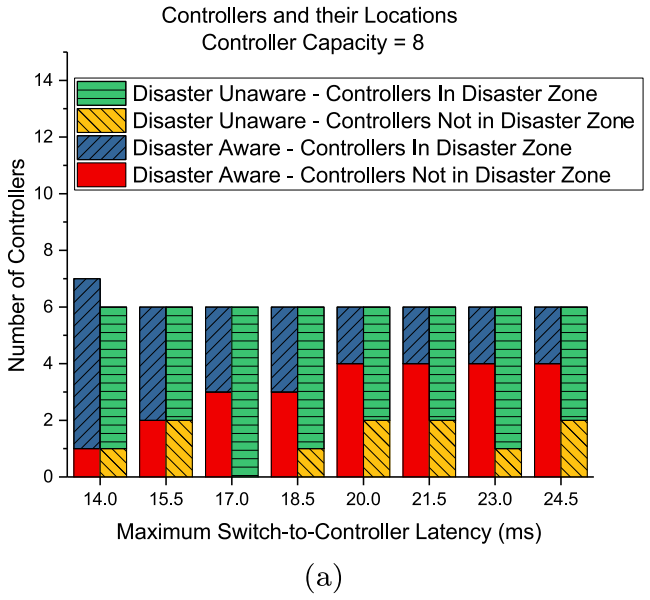


Fig. 5. Number of deployed controllers inside/outside disaster-prone zones. (a) Shows how the maximum switch-to-controller latency affects where controllers are deployed. (b) Shows how the maximum controller-to-orchestrator latency affects where controllers are deployed. Note that, as every switch must reach at least two controllers in its area (i.e., $K = 2$), the minimum number of controllers in the network is six because it has three areas (see Fig. 2).

controllers.

The results of Fig. 6 demonstrate that, without any disaster, all controllers are under 70% utilization. In the post-disaster scenario, however, a few percent of controllers (around 3%) sees around 90% utilization.

6.5. Number of orchestrators

We now study how the number of orchestrators deployed in the network affects the failure resiliency of the hierarchical control plane.

Our mathematical formulation always enforces a solution which, for the foreseen disasters, necessarily places at least one orchestrator that survives any single one of them. Thus, to investigate how the number of orchestrators impacts the control-plane resiliency, we analyze the effect

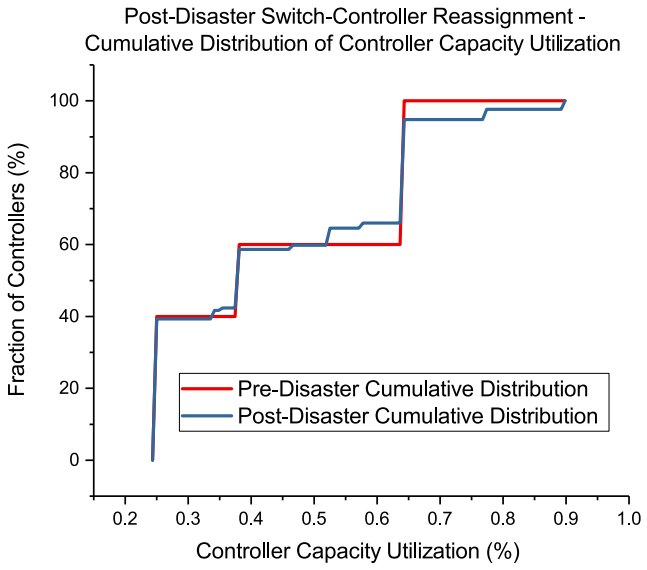


Fig. 6. Cumulative distribution of controllers by capacity utilization.

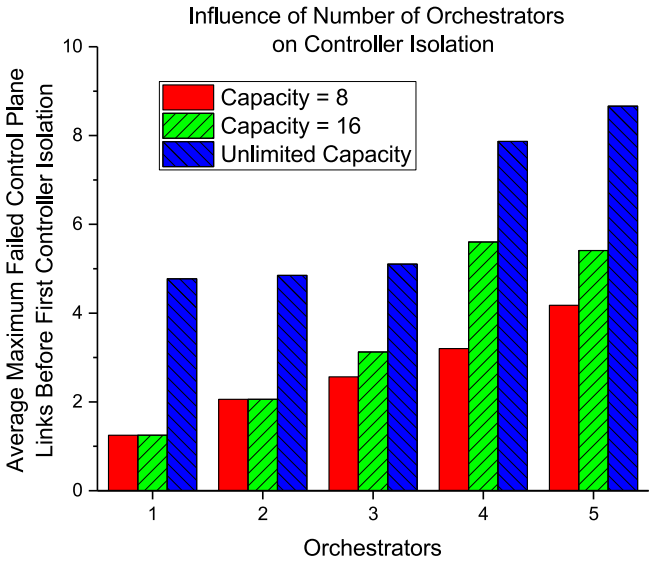


Fig. 7. Relationship between number of orchestrators and how many control-plane links must fail (on average) in order for a controller to lose contact with all orchestrators.

of random failures (i.e., which are not necessarily included among the foreseen disasters).

In Fig. 7, we ran our optimization for three controller capacities (i.e., 8, 16, or unlimited switches). In this calculation, given our disaster-aware placement solutions, we analyzed all possible combinations of failures in *control-plane links*.[5] Among all those combinations, we checked which of them caused at least one controller to be isolated from all orchestrators.

Our results show that the number of links that must fail in order for a controller to lose contact with all orchestrators increases as the number of orchestrators deployed also increases. In fact, this relationship is complex and is highly dependent on the control-plane placement solution (even though all of our simulated scenarios follow the same increasing trend as the number of orchestrators increase).

[5] We refer to links carrying control traffic (both between controller and switch, and between controller and orchestrator), as *control-plane links*.

Table 4
Execution time: Switch-to-controller.

Switch-to-controller Max. Latency (ms)	Execution Time (s)
14.0	31.60
15.5	35.67
17.0	132.23
18.5	255.18
20.0	203.94
21.5	219.39
23.0	199.35
24.5	217.24

Table 5
Execution time: Controller-to-orchestrator.

Controller-to-orchestrator Max. Latency (ms)	Execution Time (s)
15.5	77.09
17.0	95.87
18.5	99.65
20.0	151.07
21.5	154.02
23.0	236.82
24.5	200.02

6.6. Execution time

Finally, we provide average execution time of our ILP model in Tables 4 and 5. These numbers are based on the same setup (and maximum latency constraints) as that of results of Figs. 3–5. For both tables, the general trend is that, when longer latencies are allowed, the solution space increases, which causes the solver to take more time to find the optimum placement. Even in the slowest situation, the execution time is still less than five minutes.

7. Conclusion

In this work, we studied how to design robust control planes for Transport Software-Defined Networks (T-SDNs). As T-SDNs are heterogeneous and require high reliability, we considered a hierarchical control plane that contains in its top orchestrators, and in its base, controllers. Switches of a certain network area communicate with controllers, and controllers communicate with orchestrators.

We studied how disaster resiliency is achieved by minimizing a metric, called Disaster Impact (which captures the network's susceptibility to disasters). Minimizing Disaster Impact ends up placing controllers, orchestrators, and switch-to-controller and controller-to-orchestrator paths in a disaster-survivable manner. We also proposed providing resiliency against random failures through redundant entities available in the network, particularly by deploying several orchestrators, all simultaneously synchronized. Multiple orchestrators allow for readily-available backups in case of failures. We presented an Integer Linear Program formulation (ILP) capable of designing a hierarchical control plane that is resilient to both disasters and random failures. We also proposed a fast heuristic to reassign switches to controllers in case failures disrupt the switch communication with its assigned controller.

Results from our illustrative examples where a Transport Network is susceptible to earthquake damage shows that our model achieves much higher disaster and failure resiliency with minimal extra resource consumption, when compared to a disaster-unaware approach. Results also showed that our heuristic for post-disaster switch-to-controller reassignment can reassign switches without violating controller capacity, for all disasters considered during the control-plane placement calculation. Finally, we investigated how the number of orchestrators deployed affects the chances of a controller being isolated from all active orchestrators. Results demonstrated that increasing the number of orchestrators (primary plus backups) effectively means that more control-plane links must fail for any controller to be isolated from orchestrators. Thus, it is important to deploy at least one backup orchestrator in the Transport-SDN hierarchical control plane.

Acknowledgments

We thank the anonymous reviewers for their constructive comments in improving the paper. R. Lourenço was funded by CAPES Foundation (Proc. 13220-13-6). M. Tornatore acknowledges the research support from COST Action CA15127. This work has been supported by the Defense Threat Reduction Agency (DTRA) Grant Number HDTRA1-14-1-0047.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.osn.2018.05.004.

References

[1] S.S. Savas, M. Tornatore, M.F. Habib, P. Chowdhury, B. Mukherjee, Disaster-resilient control plane design and mapping in software-defined networks, in: Proc. of the IEEE HPSR, 2015.
[2] S. Jain, et al., B4: experience with a globally-deployed software defined WAN, in: Proc. of the ACM SIGCOMM, New York, NY, USA, 2013.
[3] A. Roy, et al., Inside the social network's (datacenter) network, SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 123–137.
[4] C.-Y. Hong, et al., Achieving high utilization with software-driven WAN, in: Proc. of the ACM SIGCOMM, New York, NY, USA, 2013.
[5] R. B. R. Lourenco, M. Tornatore, C. U. Martel, B. Mukherjee, Running the Network Harder: Connection Provisioning with Degradation under Resource Crunch, CoRR abs/1710.00115. arXiv:1710.00115. URL http://arxiv.org/abs/1710.00115.
[6] R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, C. Cavazzoni, Comprehensive survey on T-SDN: software-defined networking for transport networks, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2232–2283.
[7] A.S. Thyagaturu, A. Mercian, M.P. McGarry, M. Reisslein, W. Kellerer, Software defined optical networks (SDONs): a comprehensive survey, IEEE Commun. Surv. Tutor. 18 (4) (2016) 2738–2786.
[8] ONF-OTWG, Openflow-enabled Transport SDN, 2014, https://www.opennetworking.org.
[9] OIF, OIF Carrier Wg Requirements on Transport Networks in SDN Architectures, 2013, http://www.oiforum.com/public/documents/OIF_Carrier_WG_Requirements_on_Transport_Networks_in_SDN_Architectures_Sept2013.pdf.
[10] A. Blenk, A. Basta, M. Reisslein, W. Kellerer, Survey on network virtualization hypervisors for software defined networking, IEEE Commun. Surv. Tutor. 18 (1) (2016) 655–685.
[11] I. Morris, CenturyLink Targets 'six Nines' Reliability, 2016, http://www.lightreading.com/data-center/centurylink-targets-six-nines-reliability/d/d-id/723711.
[12] G. Wang, Y. Zhao, J. Huang, W. Wang, The controller placement problem in software defined networking: a survey, IEEE Netw. 31 (5) (2017) 21–27.
[13] B. Heller, et al., The controller placement problem, in: Proc. of the ACM HotSDN, New York, NY, USA, 2012.
[14] G. Yao, J. Bi, Y. Li, L. Guo, On the capacitated controller placement problem in software defined networks, IEEE Commun. Lett. 18 (8) (2014) 1339–1342.
[15] P. Xiao, W. Qu, H. Qi, Z. Li, Y. Xu, The SDN controller placement problem for WAN, in: Proc. of the IEEE/CIC ICCC, 2014.
[16] A. Aissioui, A. Ksentini, A. Gueroui, An efficient elastic distributed SDN controller for follow-me cloud, in: Proc. of the IEEE WiMob, 2015.
[17] A. Sallahi, M. St-Hilaire, Optimal model for the controller placement problem in software defined networks, IEEE Commun. Lett. 19 (2015) 30–33.
[18] M.F. Bari, A.R. Roy, S.R. Chowdhury, Q. Zhang, M.F. Zhani, R. Ahmed, R. Boutaba, Dynamic controller provisioning in software defined networks, in: Proc. of the IEEE CNSM, 2013.
[19] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, Towards an elastic distributed SDN controller, in: Proc. of the ACM HotSDN, New York, NY, USA, 2013.
[20] P. Berde, et al., ONOS: towards an open, distributed SDN OS, in: Proc. of the ACM HotSDN, ACM, New York, NY, USA, 2014.
[21] A. Krishnamurthy, S.P. Chandrabose, A. Gember-Jacobson, Pratyaastha: an efficient elastic distributed SDN control plane, in: Proc. of the ACM HotSDN, New York, NY, USA, 2014.
[22] A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, R.R. Kompella, ElastiCon; an elastic distributed SDN controller, in: Proc. of the ACM/IEEE ANCS, 2014.
[23] F.J. Ros, P.M. Ruiz, Five nines of southbound reliability in software-defined networks, in: Proc. of the ACM HotSDN, New York, NY, USA, 2014.

[24] L.F. Müller, et al., Survivor: an enhanced controller placement strategy for improving SDN survivability, in: Proc. of the IEEE GLOBECOM, 2014.
[25] D. Hock, et al., Pareto-optimal resilient controller placement in SDN-based core networks, in: Proc. of the 25th International Teletraffic Congress (ITC), 2013.
[26] D. Santos, A. de Sousa, C.M. Machuca, Robust SDN controller placement to malicious node attacks, in: Proc. of IEEE DRCN, 2018.
[27] B.P.R. Killi, S.V. Rao, Capacitated next controller placement in software defined networks, IEEE Trans. Netw. Service Manage. 14 (3) (2017) 514–527.
[28] B. Mukherjee, M.F. Habib, F. Dikbiyik, Network adaptability from disaster disruptions and cascading failures, IEEE Commun. Mag. 52 (5) (2014) 230–238.
[29] S. Neumayer, G. Zussman, R. Cohen, E. Modiano, Assessing the vulnerability of the fiber infrastructure to disasters, IEEE/ACM Trans. Netw. 19 (6) (2011) 1610–1623.
[30] OIF/ONF, SDN Transport API Interoperability Demonstration, 2017, http://www.oiforum.com/.
[31] M. Fiorani, A. Rostami, L. Wosinska, P. Monti, Transport abstraction models for an SDN-controlled centralized RAN, IEEE Commun. Lett. 19 (8) (2015) 1406–1409.
[32] A. Rostami, et al., Orchestration of RAN and transport networks for 5G: an SDN approach, IEEE Commun. Mag. 55 (4) (2017) 64–70.
[33] F. Dikbiyik, M. Tornatore, B. Mukherjee, Minimizing the risk from disaster failures in optical backbone networks, J. Lightwave Technol. 32 (18) (2014) 3175–3183.
[34] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: Proc. of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, San Jose, CA, 2012.
[35] A. Hmaity, F. Musumeci, M. Tornatore, Survivable virtual network mapping to provide content connectivity against double-link failures, in: Proc. of the IEEE DRCN, 2016.
[36] M. D. Petersen, et al., Documentation for the 2008 Update of the united states National Seismic Hazard Maps, US Geologic Hazards Science Center.
[37] IBM ILOG CPLEX v12, 1: User's Manual for CPLEX, International Business Machines Corporation, 2009, p. 157.