On Reliable Controller Placements in Software-Defined Networks

Francisco J. Ros, Pedro M. Ruiz

PII: S0140-3664(15)00342-4
DOI: 10.1016/j.comcom.2015.09.008
Reference: COMCOM 5182

To appear in: *Computer Communications*

Received date: 5 October 2014
Revised date: 31 August 2015
Accepted date: 7 September 2015

Please cite this article as: Francisco J. Ros, Pedro M. Ruiz, On Reliable Controller Placements in Software-Defined Networks, *Computer Communications* (2015), doi: 10.1016/j.comcom.2015.09.008

On Reliable Controller Placements in Software-Defined Networks

Francisco J. Ros[1], Pedro M. Ruiz

Department of Information and Communications Engineering, University of Murcia, Murcia, Spain

Abstract

In order to deploy fault-tolerant Software-Defined Networks (SDN), the logically centralized controller must be physically distributed among different devices. In this paper, we focus on determining *how many* controllers need to be instantiated, *where* they must be deployed, and *what network nodes* are under control of each of them, in order to achieve high reliability in the southbound interface between controllers and nodes. For this, we define the Fault Tolerant Controller Placement problem and develop a heuristic algorithm that computes placements with (at least) the required reliability. We run such algorithm on a set of 124 publicly available network topologies. The results are thoroughly analyzed and provide insight on the feasibility of achieving fault tolerant SDNs by carefully determining the placement of controllers.

Keywords: Software-Defined Networks, Controller Placement, Fault Tolerance, Southbound Reliability

1. Introduction and Motivation

Software-Defined Networks (SDN) provide a clear separation between the control and data planes [1]. While traditional network nodes (switches, routers, middleboxes) integrate both planes within the same device, the SDN

Email addresses: fjros@um.es (Francisco J. Ros), pedrom@um.es (Pedro M. Ruiz)
URL: http://masimum.inf.um.es (Francisco J. Ros), http://ants.inf.um.es/staff/pedrom (Pedro M. Ruiz)
[1]Corresponding author. Phone: +34868884644. Fax: +34868884151.

paradigm proposes the use of central *controllers* that dictate the behavior of the various forwarding elements (*nodes*) in the network. Control applications take advantage of the controller by querying and modifying the network state through a *northbound interface*. Similarly, the controller implements a *southbound interface* with the network nodes it is in charge of. By means of these open interfaces, SDN offers huge opportunities for network programmability, which in turn facilitates innovation in the field.

The controller in charge of a network domain is a logically centralized construct, although it must be physically distributed in order to meet performance, scalability, and fault tolerance constraints. To elaborate on the former, let us consider a large network with a single controller responsible for all forwarding elements. The communication with the farthest node might require excessive latency, therefore limiting the responsiveness of the control plane. In addition, a single controller might not be enough to service all nodes in the network. It would become a bottleneck in terms of processing power, memory, or input/output bandwidth. Furthermore, the controller itself might fail and therefore leave the network without its "brain". Note that even when the controller is operational, some of the network nodes might get disconnected from the controller due to failures in links or in other nodes within the path between the two.

In order to alleviate the aforementioned issues, multiple controllers which are physically distributed throughout the network are required. Along such line, some controller platforms have been leveraged as a distributed control plane (e.g. Onix [2]) while others have been designed with clustering as an integral feature from the beginning (e.g. OpenDaylight [3]). They exploit the fact that a single node can be connected to multiple controllers to provide reliability in the southbound interface (e.g. with OpenFlow v1.2 [4]). However, the random placement of an arbitrary number of controllers does not provide satisfactory performance [5], so some planning must be done to build an effective and reliable SDN. In particular, a decision must be made on *how many* controllers are instantiated, *where* are they deployed, and *which nodes* are under control of each of them. Despite each node should connect to several controllers for the given reasons, it is advisable to minimize the number of controllers per node in order to limit processing within the node. The more controllers, the more switch resources (cpu, memory) and bandwidth are consumed. Furthermore, we should also strive to optimize the overall number of controllers in the network, since each deployed controller has an inherent cost in terms of capex, opex, and state synchronization with the

rest of controllers.

In this paper, we focus on building reliable software-defined networks and elaborate on the *fault tolerant controller placement* (*FTCP*) problem (§3). We extend our initial work on the subject [6] by enhancing the proposed algorithm and providing further insight on the characteristics of reliable controller placements.

In our formulation, each node must be guaranteed that an operational path towards any of the controllers it connects to exists with at least a given probability. The goal of this optimization problem is to find out the controller placement that minimizes the associated cost. Given the combinatorial nature of the problem, we develop a heuristic algorithm (§4) which computes solutions that meet the reliability constraint, while trying to minimize the number of controllers per node along with the overall number of controllers.

The motivation for our proposed approach is twofold. First of all, synchronization traffic among controllers is the main source of overhead in multi-controller deployments. Some of our preliminary experiments with two OpenDaylight controllers show spikes of aggregated synchronization traffic of about 300-600 Kbps (9-node scenario), 1 Mbps (20-node scenario) and 1.8 Mbps (36-node scenario). So, large number of controllers has an important performance impact. Sencondly, each controller connection consumes switch resources (e.g. CPU and memory to maintain TLS sessions, network bandwidth for controller-to-switch commands and related reply or error messages, etc.). Such consumption of resources might be small under some assumptions. However, our proposed work does not focus on any particular southbound protocol and non-OpenFlow protocols might impose a higher burden on the network node. Thus, it seems relevant optimizing the number of controllers per switch as well.

We exploit such algorithm to explore controller placements that lead to different levels of southbound reliability in 124 publicly available network topologies (§5). In the evaluated scenarios, we find that the number of required controllers is positively correlated with the number of *spokes* (nodes with degree one) in the network topology. As expected, the total number of controllers varies greatly and is more related to the network topology than to the network size. Nonetheless, 8 controllers or less are enough for 75% of the most interesting cases even when we look for **five nines** of southbound reliability. Interestingly, each node is required to connect to just 2 controllers, which typically provide more than the required reliability threshold. On the controller side, the solutions found by our algorithm do not provide fair load

balancing unless the network topology features zero spokes. However, the load on each controller is kept to a reasonable level that can be easily handled by current controller platforms. In addition, we also note that relaxing the reliability constraint does not lead to a dramatic reduction in the number of required controllers, unless the target reliability is below the operational probabilities of most nodes, links and controllers.

We conclude this paper by discussing the main results of our analysis (§6), which indicate that fault tolerant SDNs are achievable by carefully choosing controller placements within the target network topology. Next section summarizes previous relevant works in the literature.

2. Related Work

The physical distribution of the control plane in SDN has been addressed in different works. Levin *et al.* explore the state distribution trade-offs between strongly consistent and eventually consistent models [7], given that network state is logically centralized but physically distributed among different controllers. While strongly consistent models ensure that all controllers perceive the same network view, they limit responsiveness in the control plane during the process of achieving consensus. On their hand, eventually consistent models improve liveness at the cost of possibly incorrect behavior due to different network views at the various controllers. The Onix distributed control platform [2] provides applications with flexibility to partition the network state and apply different storage mechanisms with distinct features. In order to reduce processing and synchronization overhead among controllers, Onix and other approaches like Kandoo [9] propose to organize them into hierarchical layers. A different philosophy is followed by HyperFlow [10], which defines a publish/subscribe framework to selectively propagate network events across controllers. In addition, controller load can be dynamically shifted [11] by changing the switch-controller mapping based on the actual load on each controller.

The controller placement issue has been addressed in a few papers. The first study on the subject is due to Heller *et al.* [5]. They introduce the *controller placement* problem and explore the trade-offs when optimizing for minimum latency between nodes and controllers. They find that, in most topologies, average latency and worst latency cannot be both optimized. In some cases, one controller is enough to meet common network expectations with respect to communications delay (but obviously not fault tolerance).

Additional controllers provide diminishing returns in most topologies. Yao *et al.* build upon this work to formulate the *capacitated controller placement* problem [12]. They account for the different capacities of controllers, so that the number of nodes that connect to a controller cannot generate a load that the controller is not able to deal with. Contrary to our work, they assume that each node only connects to one controller. Another preliminary work on *controller placement* is that of Bari *et al.* [17]. They propose two heuristics to dynamically provisioning controllers. However, their goals are related to minimizing flow setup time, control traffic and switch-to-controller reasignments and are not related to reliability.

The *Pareto-Optimal Controller Placement* (POCO) framework [13][14] also extends Heller *et al.*'s work [5] to consider additional aspects other than network latency. In particular, the whole solution space for placements of k controllers is analyzed with respect to node-controller latency, controller-controller latency, load balancing among controllers, and resilience in arbitrary double link or node failure scenarios. By generating all possible placements for k controllers, the trade-offs between the former metrics are explored. Recently, Lange *et al.* [19] extended the POCO framework with heuristics to make it work in large or dynamic topologies. In any case, POCO does not target any given reliability threshold β that paths between nodes and controllers must fulfill. Instead, the authors focus on one and two failure scenarios for nodes and links. Therefore, they do not consider the operational probability of every node, link, and controller in their formulation. In addition, the number of controllers k to be deployed in the network is an input parameter in the POCO framework. In our work, k is a variable to be optimized. Another difference is that POCO assumes that each node connects to one controller (the closest one), while the assignment of nodes to (possibly multiple) controllers is another variable to be optimized in our evaluation. To summarize, POCO and the algorithm presented in this paper solve different problems and both tools can be used together to get insight onto controller placements.

Another work that deals with controller placements from a reliability viewpoint is due to Hu *et al.* [15]. In this case, the authors compare through simulation different placement algorithms. Recently, the same authors extended that work in [16]. They propose a metric called expected percentage of control path loss to measure reliability. Moreover, they proof that the reliability-aware controller placement is NP-hard and again they test different placement algorithms analyzing the trade-off between reliability and

latency. However, similar to [13][14], the number k of controllers is given as input, each switch only connects to its nearest controller, and only one-failure scenarios are considered.

We introduced the FTCP problem in [6], where we provided a first version of the algorithm described in this paper. Our formulation is more general than other works in the literature, and network operators might exploit the heuristic algorithm to find controller placements that meet a given reliability constraint. We have enhanced the performance and the results of our previous work by simplifying the flow network our algorithm operates on. In [6], the computed maximum flow contains non-disjoint paths that are pruned afterwards. By slightly changing edge capacities as specified in §3.2, in this work we directly obtain disjoint paths different than those in our previous implementation. They provide higher operational probabilities and therefore lead to fewer controllers in the network and fewer southbound connections. In addition, our previous work only considers these two metrics when the objective is to achieve five nines reliability. We have extended our previous work by analyzing the topological properties related to the total number of required controllers (§5.2), the controller load in our placements (§5.4), other reliability levels (§5.5), the impact of our algorithm's parameter τ (§5.6), and the runtime of the proposed heuristic (§5.7). These new findings further shed light on the reliable placement of controllers in SDN.

There are other works that focus on reliability in SDN without addressing the controller placement problem. Among them, Akella *et al.* suggests that highly available fabrics can be built on top of some primitives like reliable flooding, global snapshots, and replicated controllers [18]. On the contrary, in this paper we do not propose any new SDN design. Both the problem formulation and the results are applicable to general network designs. The following section discusses the FTCP problem in detail.

3. The FTCP Problem

3.1. Problem formulation

Let $G(V = N \cup F; E)$ be the graph that represents the topology of a network, where N is the set of network nodes, F consists of the different facility locations where a controller can be deployed, and E corresponds to the links among them. Each vertex $v \in V$ has a given probability p_v of being operational (up and running). Analogously, links $(u, v) \in E$ are operational

with probability $p_{u,v}$. We assume different i.i.d. operational probabilities for links, nodes, and controllers.

Let us define binary variables y_i that equal 1 if facility $i \in F$ holds a controller, 0 otherwise. The cost of deploying the controller in such facility is given by ψ_i. Let x_{ij} be binary variables that equal 1 is node $j \in N$ connects to the controller in facility $i \in F$, 0 otherwise. The cost of serving node j with the controller in i is given by ω_{ij}. Costs ψ_i and ω_{ij} can be measured in the terms more relevant to the network operator, for instance they could mean economic cost.

Every node j has to connect to a subset $I_j \subseteq F$ of deployed controllers such that the probability of having at least an operational path in G between j and I_j is higher than a given threshold β. Following conventional network reliability terminology [20], we refer to such probability as the *k-terminal reliability* $R(G, j, I_j)$ between j and I_j, where $k = |I_j|$.

Hence, the **fault tolerant controller placement** (FTCP) problem can be formulated as:

$$\begin{aligned}
\min \quad & \sum_{i \in F} \psi_i y_i + \sum_{j \in N} \sum_{i \in F} \omega_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{\forall i \in F} x_{ij} > 0, \quad \forall j \in N \\
& x_{ij} \leq y_i, \quad \forall j \in N, \forall i \in F \\
& R(G, j, I_j) \geq \beta, \quad \forall j \in N \\
& x_{ij}, y_i \in \{0, 1\}
\end{aligned}$$

This formulation is related to the *fault tolerant facility location* (FTFL) problem [21], which belongs to the NP complexity class. One of the main differences between both problems is that FTFL predetermines the minimum number of facilities to which each node connects. Given that this is a variable under study in our work, FTCP does not impose such constraint. In addition, we introduce a non-linear constraint on the network reliability R[2]. Computing R is an NP-hard problem[3] by itself, so for the remainder of

[2]Let R be a shorthand for $R(G, j, I_j)$.

[3]Intuitively, computing R requires enumerating all possible states of the network (controllers, nodes and links which are up/down), checking whether each state allows for (at

this paper we consider a lower bound $\hat{R}$ on network reliability that can be obtained in polynomial time.

3.2. Reliability lower bound

G can be applied a series of reliability-preserving transformations [20] that allow us to compute $\hat{R}$ in a simple and intuitive way. Figure 1 shows an example.

In first place, we build a directed graph G' by creating two anti-symmetric edges $(u, v), (v, u)$ for every undirected edge (u, v) in G. The same operational probability $p_{u,v}$ is assigned to every anti-symmetric edge. Then, we transform G' from a network with unreliable nodes and links, to an equivalent one with perfectly reliable nodes and unreliable links. To do this, we split each vertex v (node or facility) into two vertices v_0, v_1 which are connected by means of a directed edge (v_0, v_1). The operational probability p_{v_0,v_1} equals the operational probability p_v. Directed edges (u, v) become (u_1, v_0) with $p_{u_1,v_0} = p_{u,v}$. After these transformations, the following equality holds: $R(G, j, \{i : i \in I_j\}) = R(G', j_0, \{i_1 : i \in I_j\})$. Finally, we add a virtual sink t and perfectly reliable directed links $(i, t), \forall i \in I_j; p_{i,t} = 1$. Hence, we get $R(G, j, I_j) = R(G', j_0, t)$. The latter expression is referred to as *two-terminal reliability*, and a lower bound $\hat{R}$ can be computed in polynomial time [22].

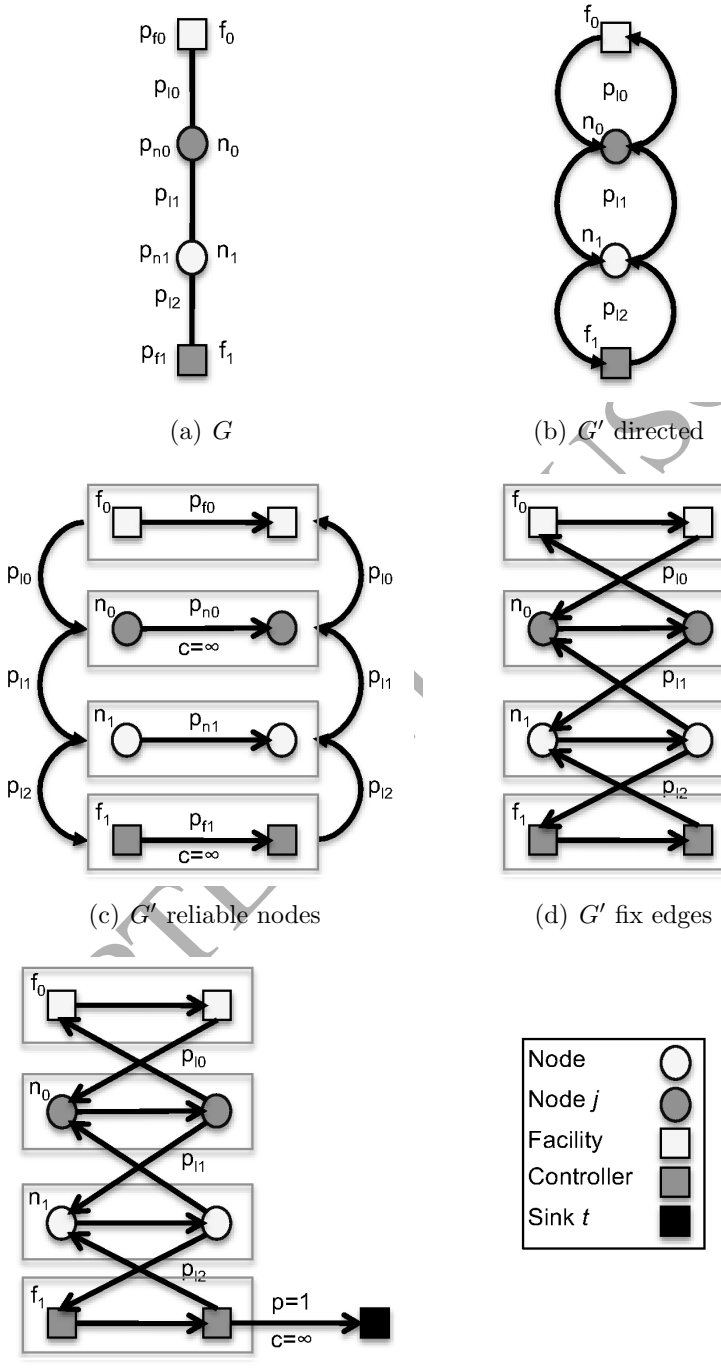
The rationale behind the bound is that the probability of not having an operational path from the source towards the sink, cannot be larger than the product of the failure probabilities of λ disjoint paths between the two (where λ is the *local edge-connectivity*[4] between j_0 and t in G'). Figure 2 clearly shows the concept. Edge (u, m) is not taken into account to compute $\hat{R}$, given that it is not part of any disjoint path between j and t. However, it contributes positively to the network reliability R, so $\hat{R} < R$ in this case.

We denote the set of disjoint paths from j_0 to t in G' as $\Pi_{j_0,t}$. Given that we assumed statistical independence among operational probabilities, the lower bound $\hat{R}$ on the southbound reliability between j and I_j is given by:

---

least) a path from j to I_j, and computing the probability of each state. Refer to [20] and the references therein for a more elaborated discussion.

[4] In other words, λ is the minimum number of edges that must be removed from G' to disconnect j_0 and t.



(a) G (b) G' directed

(c) G' reliable nodes (d) G' fix edges

(e) G' add sink (f) Legend

Figure 1: Topology transformation from original graph G to flow network G' with reliable nodes.

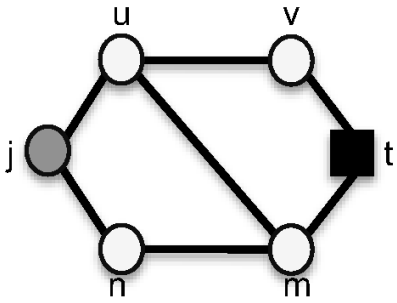


Figure 2: Example topology with two disjoint paths (j, u, v, t) and (j, n, m, t). Edge (u, m) is not considered to compute lower bound $\hat{R}$.

$$\begin{aligned} \hat{R}(G', j_0, t) &= 1 - \prod_{\forall \pi \in \Pi_{j_0,t}} (1 - \prod_{\forall (u,v) \in \pi} p_{u,v}) \\ &\leq R(G', j_0, t) \end{aligned}$$

In order to find a set $\Pi_{j_0,t}$, we take advantage of a well-known result from optimization theory. We can find the edge-disjoint paths from a source to a sink in a graph by solving the *max-flow* problem in the associated *flow network* in which all edges have unit capacity. Given the transformations we perform to get G' from G, additional considerations must be taken when creating the flow network from G', as detailed in the following.

In our case, j_0 is the source and t is the sink of the flow. All edges in G' are decorated with a capacity attribute, as shown in Figure 1. The source is allowed to issue as much flow as possible (infinite capacity), since the virtual edge (j_0, j_1) should be part of multiple paths without breaking the disjointness property in G (note that it comes from a previous transformation and is not a real edge). Similarly, facilities can receive flow from multiple disjoint paths and all of them should be allowed to traverse the facility. Therefore, the virtual edge (v_0, v_1) of a facility also has infinite capacity. The same reasoning applies to the virtual edge that links avery facility with the sink t, while the remaining edges have capacity 1 (not shown in Figure 1).

By solving the *max-flow* problem on the capacitated network G', we directly find all disjoint paths that contribute to network reliability. Our previous work [6] took a different approach to decorate G', and it required an additional step to get $\Pi_{j_0,t}$.

In the following section we take advantage of the bound $\hat{R}$ to develop a placement algorithm for the FTCP problem.

4. Heuristic Algorithm

Given the complexity of FTCP, we develop a heuristic algorithm that computes subsets $I_j, \forall j \in N$ that meet the reliability constraint $\hat{R} \geq \beta$. Since $\hat{R}$ is a lower bound, the solutions found by our algorithm provide an upper bound on the number of controllers required for β-reliability in the southbound interface.

The objective function of the FTCP optimization problem depends on the total number of controllers that are instantiated and on the number of controllers to which every node connects. Hence, our algorithm strives to find placements with "desirable properties", namely with few overall controllers and few connections between nodes and controllers. The design of the heuristic is based on the next observations.

In first place, our algorithm respects the reliability constraint $\hat{R}(G', j_0, t) \geq \beta$. $\hat{R}$ is computed as explained in the former section, what determines the facilities where a controller must be deployed. In fact, only those facilities which are part of the disjoint paths between j_0 and t are required for meeting the reliability constraint. Therefore, we can keep the number of connections of a node at a minimum by avoiding unnecessary controllers at facilities that do not contribute to increased reliability.

In order to minimize the total number of deployed controllers, we heuristically rank facilities according to their expected contribution to southbound reliability for as most nodes as possible. The rationale here is to prefer facilities that could be useful to many nodes. We employ connectivity as a surrogate for reliability, given the intuition that a controller that can be reached through different paths can be employed by more nodes. Hence, we assume that a facility i ranks better than another if the degrees D_j of its neighbor nodes $j \in n(i)$ are higher, where $n(\cdot)$ is the neighborhood function in G. Once it has been decided that facility i holds a controller to which node j connects, we increase the rank of i so that other nodes prefer to reuse a controller in such facility, instead of deploying an additional controller in a different location. For this, our algorithm employs an incentive parameter τ that determines how much the score of a facility gets augmented.

Algorithm 1 details our proposal. Initially, all facilities i are assigned a score S_i which depends on the connectivity of their neighborhood $n(i)$ (line

1). Then, we iterate through all nodes j to find subsets of facilities I_j that meet the reliability constraint (lines 2-14). Facilities are considered from highest to lowest score (lines 4-5). Current facility is added to I_j (line 6) and the corresponding reliability bound $\hat{R}$ is computed (lines 7-10). If it is enough to satisfy the reliability constraint, I_j is finally assigned to the facilities that are part of the disjoint paths employed to calculate $\hat{R}$ (lines 11-13). The scores of all facilities in I_j are multiplied by τ to try to reuse the same controllers for subsequent nodes (line 14).

Algorithm 1: Heuristic algorithm for FTCP.

Input: Graph G, nodes N, facilities F, reliability threshold $\beta < 1$, incentive parameter $\tau > 1$

1 $S_i \leftarrow \sum_{j \in n(i)} D_j, \forall i \in F$;
2 **foreach** $j \in N$ **do**
3 $\quad I_j \leftarrow \emptyset$;
4 $\quad rank \leftarrow [i \in F]$ in reverse order of S_i;
5 $\quad$ **foreach** $i \in rank$ **do**
6 $\quad\quad I_j \leftarrow I_j \cup \{i\}$;
7 $\quad\quad$ Build G' from G and I_j;
8 $\quad\quad f \leftarrow max\_flow(G', j_0, t)$;
9 $\quad\quad \Pi_{j_0,t} \leftarrow \{\pi_k\}_{k=1}^{\lambda}, \pi_k = \{(u,v) : f(u,v) = 0\}$;
10 $\quad\quad \hat{R} \leftarrow 1 - \prod_{\forall \pi \in \Pi_{j_0,t}} \left(1 - \prod_{\forall (u,v) \in \pi} p_{u,v}\right)$;
11 $\quad\quad$ **if** $\hat{R} \geq \beta$ **then**
12 $\quad\quad\quad I_j \leftarrow \{k \in I : (\exists \pi \in \Pi_{j_0,t} : k \in \pi)\}$;
13 $\quad\quad\quad$ break;
14 $\quad S_i \leftarrow \tau S_i, \forall i \in I_j$;

5. Analysis of Fault-Tolerant Placements

Our interest in developing Algorithm 1 lies on its ability to analyze existing network topologies from the viewpoint of deploying fault-tolerant SDNs. Given a topology, if network nodes implement an SDN southbound protocol (e.g. OpenFlow), there is a set of facilities where controllers can be deployed, and we require that each node is effectively connected to at least one controller with high reliability, we want to answer the following questions: *How*

	Links	Nodes	Controllers
η	0.999981	0.999983	0.999975
σ	15526	46047.5	13879.7

Table 1: Scale parameter η and shape parameter σ of Weibull distributions for the operational probabilities employed in our evaluation.

many controllers must be deployed at least? *Where*? *What nodes* must be connected to each controller? In other words, we want to explore solutions to the FTCP problem. The remainder of this section describes our evaluation setup and tries to shed some light onto these issues.

5.1. Evaluation setup

The network topologies under consideration have been obtained from a publicly available repository, namely the Internet Topology Zoo [23]. From the set of networks reported in [23] we exclude all disconnected ones, what leaves us with 124 topologies at the Point-of-Presence (PoP) level. Hence, in our analysis every PoP corresponds to one network node. In addition, we consider each PoP as a candidate for hosting a controller. Therefore, every node links with one facility in the topology graph G.

The operational probabilities of links, nodes, and controllers are assumed to follow different i.i.d. Weibull distributions (Table 1). Such configuration is intended to reproduce some results reported in the literature, namely long tails in the downtime distribution of WAN links with four nines of median reliability [24], higher reliability for nodes [25], and lower reliability for higher layer devices [25]. For each network topology, we run twenty independent experiments in which the operational probabilities of links, nodes, and controllers are randomly drawn from the previous Weibull distributions (different seed per run).

Unless otherwise stated, in the following we fix parameter $\beta = 0.99999$ in order analyze controller placements when we look for five nines of southbound reliability. Section 5.5 extends the analysis to other reliability levels. Similarly, we also assume parameter $\tau = 5$. We perform a sensitivity analysis of such parameter in §5.6, to provide the rationale of our choice.

5.2. Number and location of controllers

Our results indicate that, not surprisingly, the number of controllers required for high southbound reliability is dependent on the network topology.

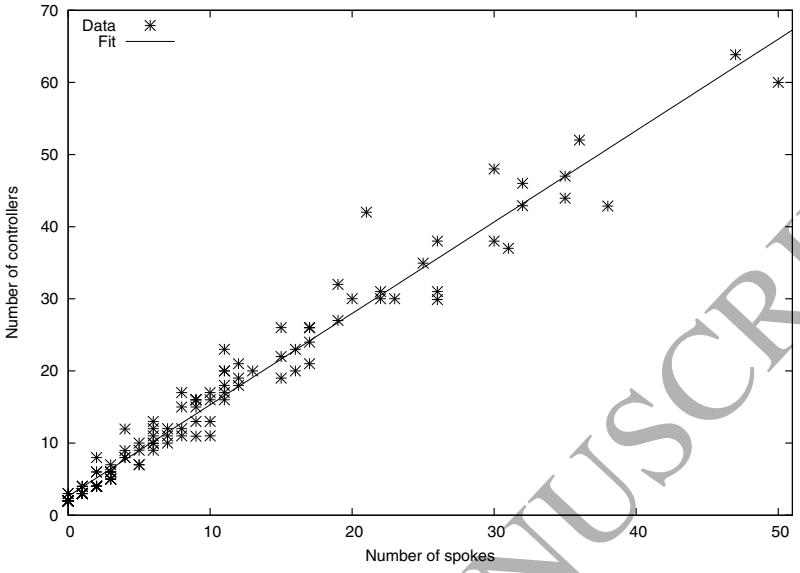


Figure 3: Scatterplot and linear fit of the number of spokes and the average number of required controllers ($\beta = 0.99999$, $\tau = 5$).

A non-reliable network, i.e. a network without redundant paths, requires many more controllers than a redundant network of similar size. One interesting aspect that we observe regarding the topological properties of the analyzed networks, is that the number of nodes with degree one (spokes) is highly related to the total number of controllers determined by Algorithm 1 (see Figure 3). In fact, the correlation coefficient of these two variables is higher than 0.98.

In order to facilitate the analysis, we arbitrarily partition the topology dataset into two disjoint groups: topologies that require on average less or as many controllers as half the number of nodes, and those that require more than half the number of nodes. From now on we focus on the first group, since it contains the most interesting scenarios. Clearly, it is expensive (in terms of controllers) to achieve southbound reliability in an inherently unreliable network topology. Although we do not show detailed numbers for our second group of topologies, let us discuss them briefly. Such group is mostly comprised of tree-like networks (mainly hub-and-spoke) [23], so few (if any) redundant paths exist in the topology. This means that the number of required controllers approaches the number of nodes. For instance, the

	Mode	Median	75-prctile	Mean	Std. Dev.
$\beta = 0.99999$	3	4	8	6.58	6.30
$\beta = 0.9999$	2	4	8	6.54	6.07
$\beta = 0.999$	1	1	1	1.18	0.54

Table 2: Statistics for the total number of required controllers in topologies requiring on average less or as many controllers as half the number of nodes ($\tau = 5$).

`Itnet` network has a star topology and it needs as many controllers as nodes for five nines southbound reliability.

Figure 4 provides insight on the overall number of required controllers in network topologies with certain degree of redundancy, ordered from lower to higher size along the X axis. Note that for every given scenario, the number of controllers obtained is quite similar regardless the different operational probabilities that are assigned to nodes, links and controllers. Therefore Figure 4 just shows the mean value per network topology, given that it is significative. Hence, in our evaluation the network topology plays a more important role than the particular probability configuration. Overall, the number of controllers required for highly reliable southbound connections remains below a reasonable level (Table 2). In fact, 75% of the scenarios we discuss here require 8 controllers or less for five nines reliability. On average, 6.58 controllers are needed, despite the dispersion of the data is high (due to the corner case of the `Cogentco` topology, discussed later). All these results improve the ones obtained in our previous evaluation [6].

In redundant enough topologies, controllers must be usually located at central facilities with high connectivity, according to the solutions found by our algorithm. However, some of them might be devoted to provide specific nodes with increased reliability. This is because some nodes (e.g. spokes) do not have enough paths with acceptable operational probability to other potential controllers. Hence, at least one controller must be deployed near such nodes.

As an example of a small redundant network, let us consider the `Sprint` topology that features 11 nodes (Figure 5(a)). In all cases, three controllers suffice to provide five nines reliability to every node. One controller must be placed on a central PoP with high connectivity, such as Stockton. Given that the node at Boulder is a spoke and therefore connects to the rest of the network by means of a single link, the reliability bound $\hat{R}$ on such placement

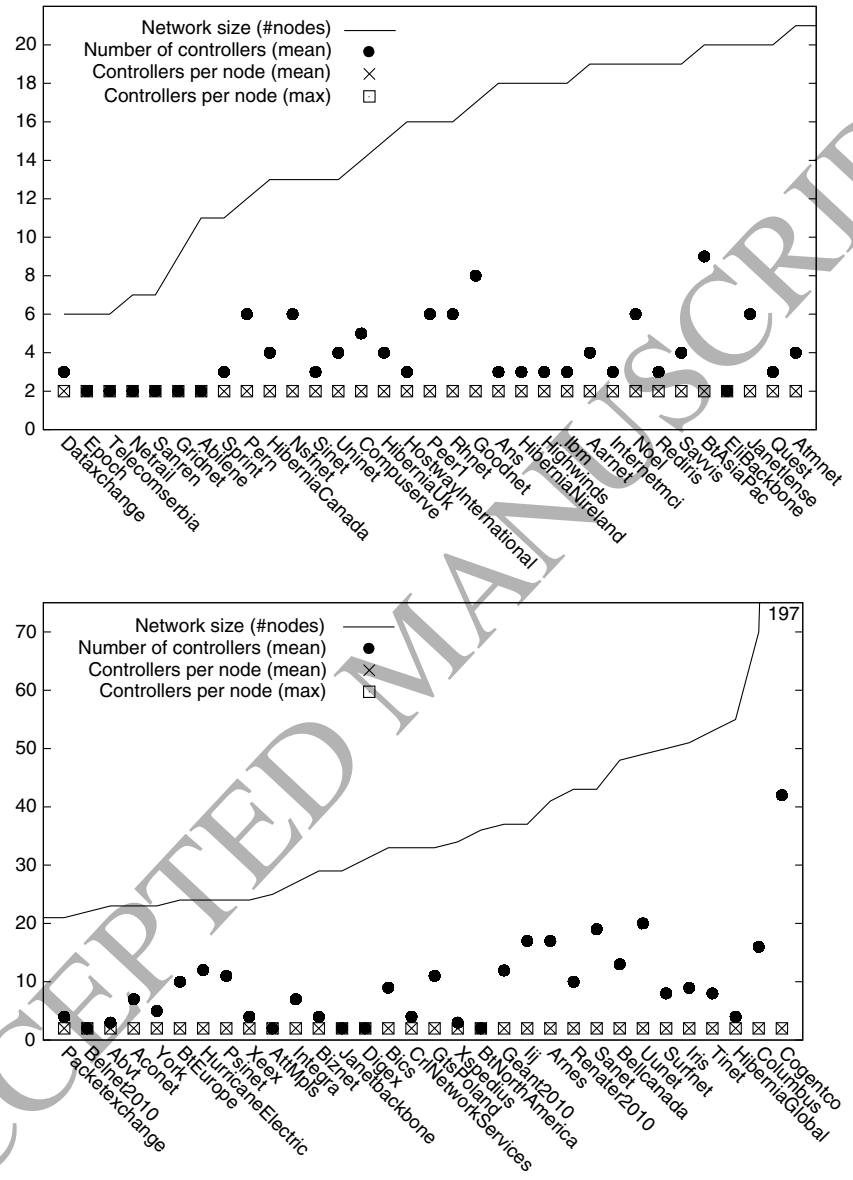


Figure 4: Overall number of controllers (mean) and controllers per node (mean, maximum) in topologies requiring on average less or as many controllers as half the number of nodes ($\beta = 0.99999$, $\tau = 5$).

is restricted by the operational probabilities of the node at Cheyenne and the link between the two. If these are not high enough, close-by controllers must be deployed (in Cheyenne and Boulder itself in the case of the instance shown in the figure).

Bigger networks like `BtNorthAmerica` (36 nodes, see Figure 5(b)) meet the reliability constraint with just two controllers. On its hand, the `Cogentco` network (197 nodes) requires 42 controllers. As reported in [23], this network has an average node degree (< 2.5) close to the lower bound for a connected graph of the same size (1.99). It is mainly comprised of network rings, some hubs and quite a few spokes, so there is certain degree of redundancy that allows for reducing the number of controllers although the overall amount remains high. This might be an issue for some applications if strong consistency of network state is required among controllers. However, an intercontinental network like this one would require multiple controller domains for management and operational reasons, so strong consistency between domains might be relaxed to achieve better liveness [7].

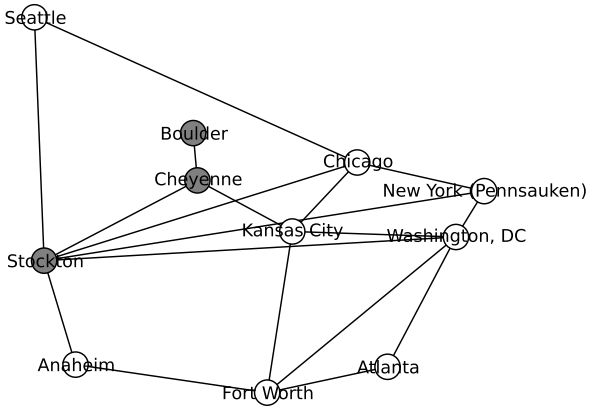
5.3. Southbound connections

Regardless the topology, each node only needs to establish a connection with two controllers at maximum (Figure 4). Note that the same is true for the remainder of the evaluated topologies not shown in the figure. Again, this result is more optimistic than our previous evaluation, in which some nodes required three connections [6].

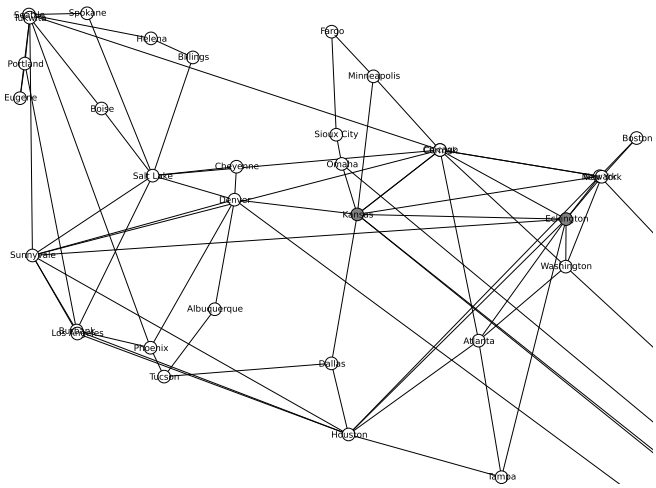
Interestingly, with such few connections, nodes typically achieve more than five nines reliability as reported in Table 3. This is the minimum reliability level that nodes get, since we actually compute a lower bound $\hat{R}$ on network reliability. This result suggests that SDN operators might relax the reliability constraint to try to reduce the number of deployed controllers, while still providing most nodes with very high southbound reliability. Section 5.5 digs into this possibility.

5.4. Controller load

We have just seen that fault-tolerant placements can be achieved with minimum overhead at the network node, that only requires two connections to different controllers. On the other hand, there is an associated overhead at the controller with respect to the number of nodes that connect to it.



(a) `Sprint`



(b) `BtNorthAmerica`

Figure 5: Location of controllers (highlighted in grey) in two evaluated topologies ($\beta = 0.99999$, $\tau = 5$).

$\hat{R}$	3 nines	4 nines	5 nines	6 nines	7 nines	$\geq$ 8 nines
$\beta = 0.999$	93.43%	4.09%	1.17%	1.25%	0.04%	0.02%
$\beta = 0.9999$	-	4.47%	7.50%	56.40%	29.84%	1.79%
$\beta = 0.99999$	-	-	7.78%	57.50%	31.88%	2.84%

Table 3: Percentage of nodes that can find an operational path towards any of their connected controllers with different reliabilities ($\tau = 5$).

Figure 6 shows the number of nodes served by the controller with the fewest and most connections[5]. We can clearly distinguish between two different groups of controller placements: those that achieve very high load balancing (controllers with the fewest and most connections serve approximately the same number of nodes), and those that feature very high load imbalance (some controllers only manage one node). In the figure, it is shown that placements in the first group always correspond to network topologies without spokes. Spokes usually require in-premises controllers mostly devoted to that particular node, therefore increasing controller load imbalance.

However, the actual load on the controller is dependent on a number of factors other than the number of nodes that connect to it. If the controller operates proactively and pre-populates flows, controller load is kept at a minimum and does not suppose a scalability issue. On the other hand, a completely reactive scheme in which every new flow generates a request for the controller may challenge controller performance. In practice, a mix of proactive and reactive flows are to be encountered in any given SDN implementation. In our evaluation, the maximum number of nodes per controller is well below 50 in the great majority of cases (Figure 6). Previous studies suggest that a controller can satisfactorily handle such load [8], even under a purely reactive scheme with 100k end hosts. In addition, each node connects to multiple controllers and they can play different roles, e.g. 'master', 'slave' and 'equal' semantics in OpenFlow 1.2 and beyond. Slave controllers do not receive asynchronous messages such as flow requests, thus their associated load is still lower.

[5]Each point in the figure is the result of a single run. Given that there is so little variability, in most experiments results are identical in the 20 randomized runs.

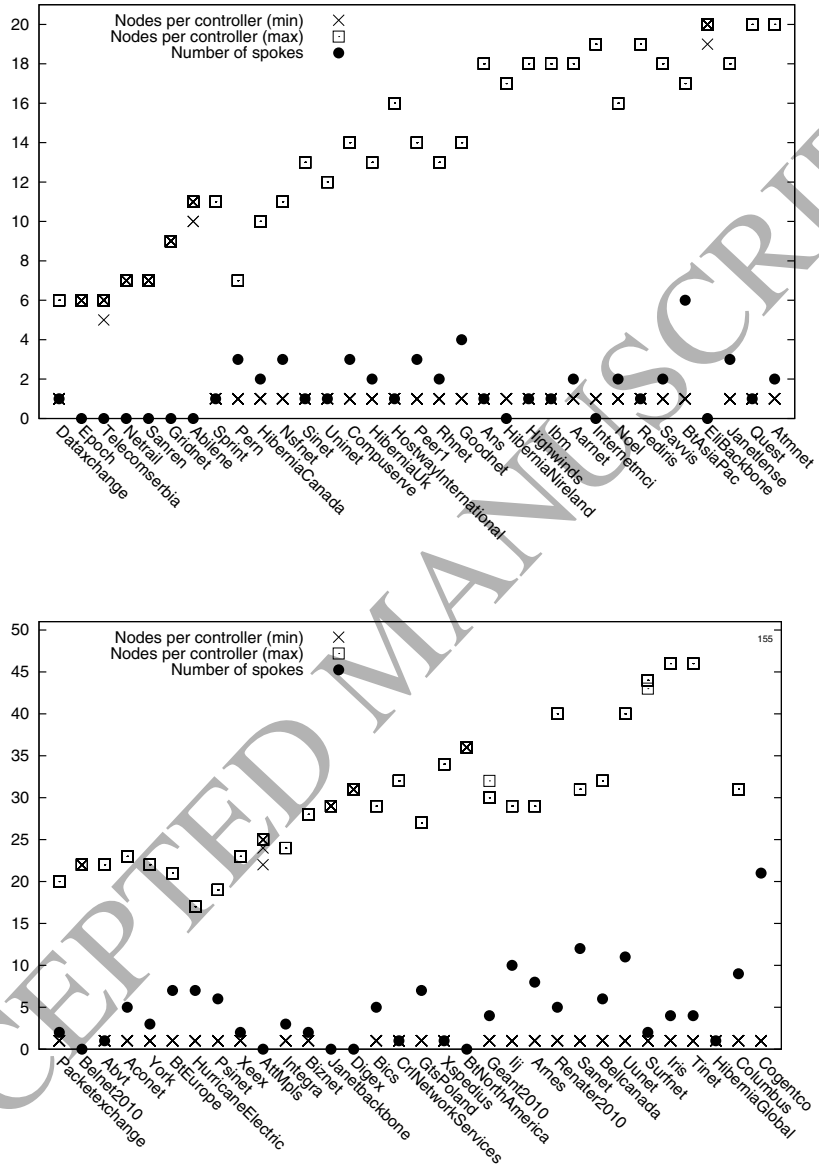


Figure 6: Minimum and maximum number of nodes served by a controller in topologies requiring on average less or as many controllers as half the number of nodes ($\beta = 0.99999$, $\tau = 5$).

5.5. Relaxing the reliability constraint

So far, we have limited our study to the case in which five nines of southbound reliability ($\beta = 0.99999$) are required. In this section we relax the reliability constraint, in order to evaluate controller placements when we look for just three or four nines.

Figure 7 shows the number of controllers that are required per evaluated topology, while Table 2 provides aggregate statistics. In both cases we can see that the amount of controllers to be deployed for five or four nines is very similar. Even in large networks, the savings obtained when planning for $\beta = 0.9999$ is lesser than three controllers. Furthermore, most nodes still achieve reliability levels around six and seven nines, as it is the case for $\beta = 0.99999$ (Table 3). However, for $\beta = 0.999$ the number of controllers dramatically decreases and most nodes actually get three nines reliability. This is because the operational probabilities we use for controllers, nodes and links are mostly above such threshold. This makes finding an operational path towards the controller fairly easy in most cases.

Hence, in this section we can see the trade-off between the two alternatives to design a reliable system, namely deploying few reliable components or relying on more components with less reliability. Since hardware vendors provide higher quality products over time, and our evaluation setup does not make optimistic assumptions about the different operational probabilities, many software-defined networks can be expected to require less controllers than the numbers reported in this paper.

5.6. Sensitivity analysis

The placements found by Algorithm 1 are dependent on the incentive parameter τ. So far, we have used a fixed value of $\tau = 5$ without further explanation. To conclude our analysis, we conduct a sensitivity analysis of the impact of such parameter on the total number of required controllers when we look for five nines reliability. This provides the rationale for our choice.

Figure 8 shows a comparison for values of $\tau = 1, 2, 5$, providing the mean number of controllers returned by our algorithm along with the 95% confidence interval. As expected, higher values of τ provide solutions with less overall controllers, since this is the purpose of this parameter. In some scenarios, all values of τ return the same number of controllers. But in general, the case for $\tau = 1$ performs worse than higher values of the parameter. In many cases there is no statistical difference between experiments with $\tau = 2$

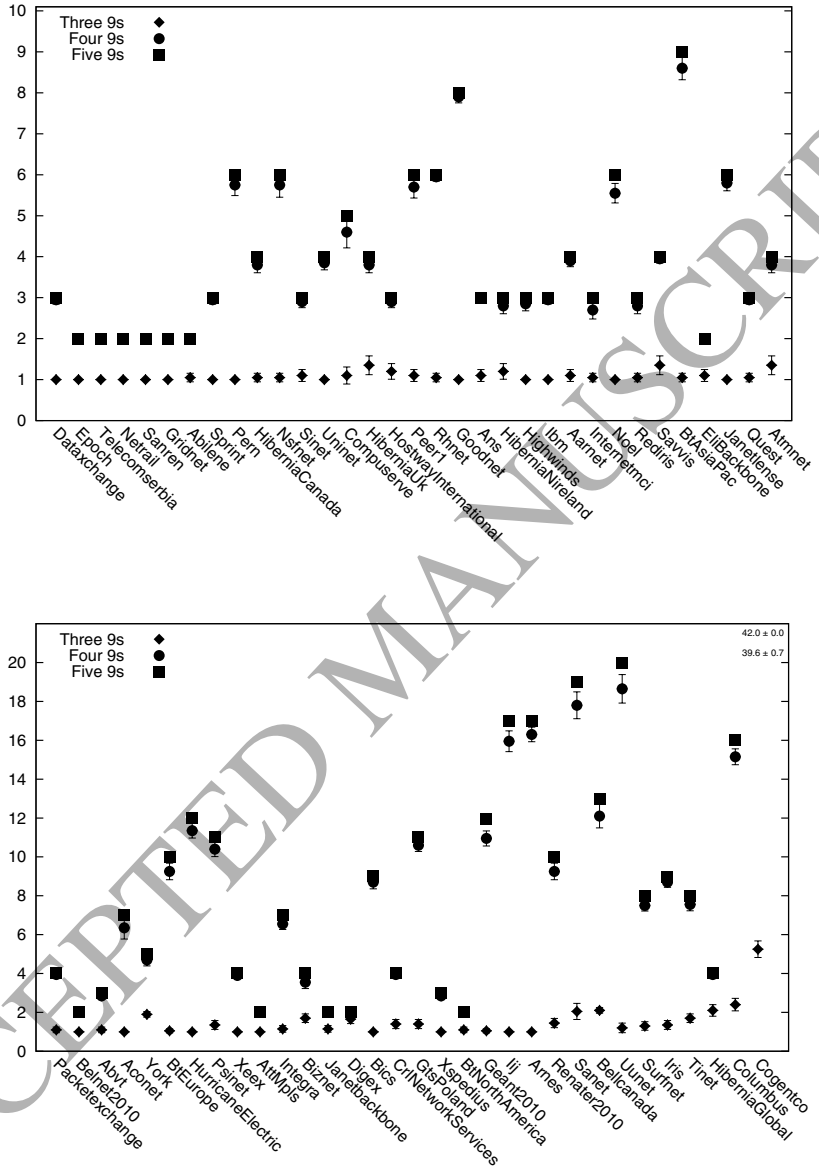


Figure 7: Overall number of controllers (mean ± 95% confidence interval) for different β reliabilities in topologies requiring on average less or as many controllers as half the number of nodes ($\tau = 5$).

	Mode	Median	{90,95,99}-prctile	Mean
$\beta = 0.99999$	0.04	1.58	{27.9, 43.3, 107.5}	16.9
$\beta = 0.9999$	0.01	1.57	{19.7, 36.5, 94.2}	16.8
$\beta = 0.999$	0.01	0.23	{1.29, 1.80, 3.87}	1.09

Table 4: Runtime statistics (seconds) of our heuristic in all simulated scenarios ($\tau = 5$).

and $\tau = 5$, despite the latter provides slightly smaller solutions in six network topologies. Hence, in this paper we decided to stick with $\tau = 5$ for the overall evaluation.

5.7. Runtime

We finish this section by providing some statistics on the runtime required by Algorithm 1 to compute reliable controller placements. For these experiments we run our heuristic for all simulated scenarios in a Linux computer with an Intel Xeon CPU at 2.67 GHz and 14GB of RAM. Controllers are launched in virtual machines using VirtualBox as hypervisor. Such virtual machines have 1 virtual CPU and 1GB of RAM, and they control network topologies created with mininet. Table 4 shows that our heuristic usually provides a solution in just a few seconds. In the worst-case scenarios for the algorithm, it still computes a reliable placement in less than two minutes. Hence, we can conclude that the algorithm is practical to use.

6. Discussion and Conclusion

When nodes get disconnected from controllers, SDN applications lose visibility and control of part of the network. Therefore, in order to exploit the full potential of the SDN paradigm, it is important to design reliable southbound interfaces between nodes and controllers. Our work in this paper addresses such concern.

We evaluated solutions to the fault tolerant controller placement problem in a wide range of network topologies. Such solutions provide SDN operators with guidance on *how many* controllers should be deployed, *where* in the network topology they should go, and *what network nodes* connect to each of them. While answers are dependent on the topology itself, many networks require a reasonable number of controllers to achieve very high reliability. The number of spokes in the topology is directly correlated to the total number of required controllers, so it is less expensive to implement a highly reliable

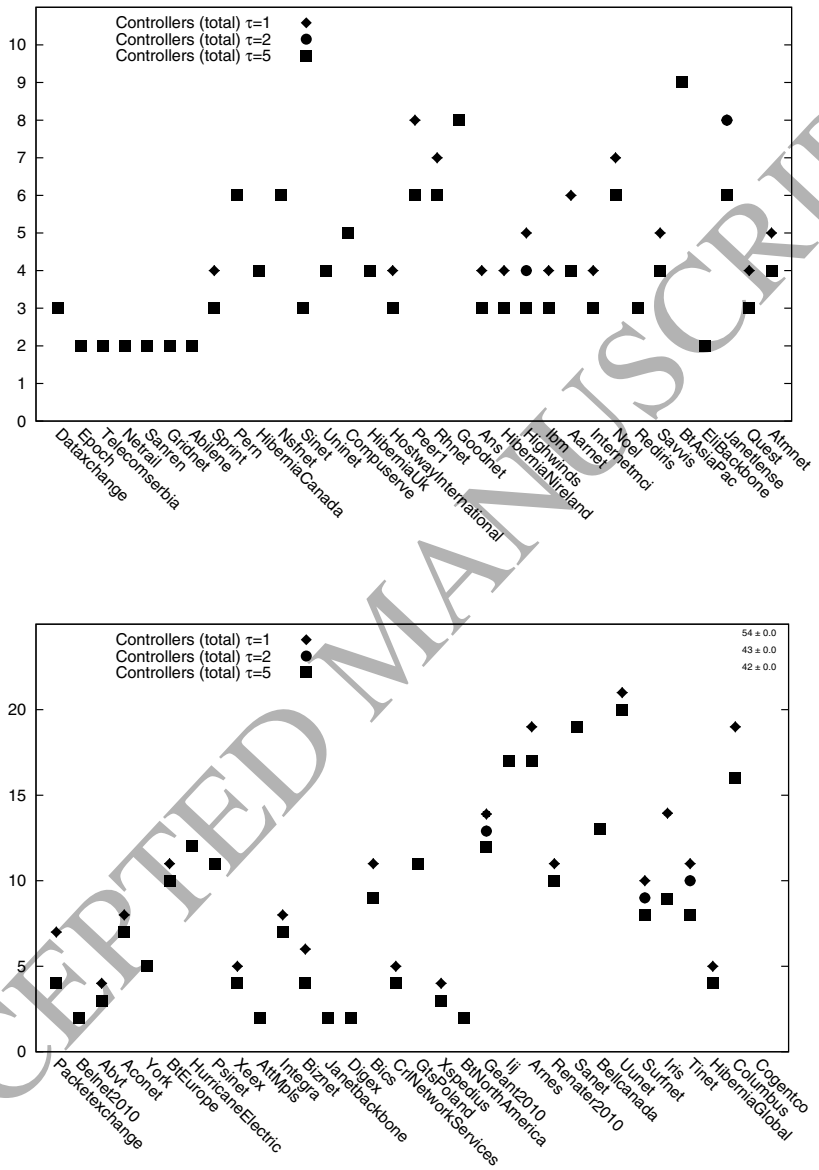


Figure 8: Overall number of controllers (mean ± 95% confidence interval) for different values of the incentive parameter τ ($\beta = 0.99999$).

southbound interface in topologies with few nodes with degree one. However, adding additional links in a wide-area network is much more expensive than deploying a new controller. Thus, careful controller placements can help minimize the total deployment cost. In addition, the number of controllers reported in this work is an upper bound since our proposed algorithm works on a lower bound on southbound reliability. These placements put a very light load on network nodes, requiring at most two connections with different controllers. On their hand, controllers must handle very different numbers of nodes, but in the evaluated topologies the processing overhead of the most challenged controller is well below the limit of reported controller performances.

In our evaluation we assumed in-band control. However, the proposed algorithm can also be applied for out-of-band or hybrid control networks. Since the algorithm takes the network topology as an input graph, our approach does not impose any restriction in this regard. On the other hand, it requires controllers, nodes and links to be tagged with their corresponding operational probability. This information can be gathered from network monitoring and/or issue tracking systems that log downtimes of the different network elements. Along with controller virtualization, the former can be leveraged to develop an automatic controller provisioning system that employs an algorithm to deploy controllers on-demand for highly reliable SDNs.

Putting altogether, we hope the work in this paper to help prospective SDN operators mitigate their concerns about having a logically centralized control plane in their networks, as well as to promote research on the topic.

Acknowledgment

This work has been partially supported by the European Commission through the Multi-Gigabit European Research and Education Network and Associated Services (GN3plus) project under grant agreement no. 605243.

References

[1] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks", *ONF White Paper*, April 2012.

[2] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix:

A Distributed Control Platform for Large-scale Production Networks", in *Proc. of the 9th USENIX Conference on Operating Systems Design and Implementation* (OSDI'10), pp. 1–6, 2010.

[3] The OpenDaylight Project, "OpenDaylight - An Open Source Community and Meritocracy for Software-Defined Networking", *White Paper*, April 2013.

[4] Open Networking Foundation, "OpenFlow Switch Specification" (Version 1.2.0), December 2011.

[5] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem", in *Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks* (HotSDN'12), pp. 7–12, 2012.

[6] F. Ros, and P. Ruiz, "Five Nines of Southbound Reliability in Software-Defined Networks", in *Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (HotSDN 2014), pp. 31–36, August 2014.

[7] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized? State Distribution Trade-offs in Software Defined Networks", in *Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks* (HotSDN'12), pp. 1–6, 2012.

[8] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-Defined Networks", in *Proc. of the Second USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (Hot-ICE'12), 2012.

[9] S. H. Yeganeh, and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications", in *Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks* (HotSDN'12), pp. 19–24, 2012.

[10] A. Tootoonchian, and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow", in *Proc. of the Internet Network Management Conference on Research on Enterprise Networking* (INM/WREN'10), pp. 3–3, 2010.

[11] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an Elastic Distributed SDN Controller", in *Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks* (HotSDN'13), pp. 7–12, 2013.

[12] G. Yao, J. Bi, Y. Li, and L. Guo, "On the Capacitated Controller Placement Problem in Software Defined Networks", *IEEE Communications Letters*, Vol. 18, No. 8, pp. 1339–1342, August 2014.

[13] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks", in *Proc. of the 25th International Teletraffic Congress* (ITC'13), pp. 1–9, 2013.

[14] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks", in *Proc. of the IEEE/IFIP Network Operations and Management Symposium* (NOMS'14), pp. 1–2, 2014.

[15] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware Controller Placement for Software-Defined Networks", in *Proc. of the IEEE/IFIP International Symposium on Integrated Network Management* (IM'13), pp. 672–675, 2013.

[16] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "On Reliability-optmized Controller Placement for Software-Defined Networks", *Communictions, China*, Vol. 11, No. 2, pp. 38–54, February 2014.

[17] 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 9th International Conference on Network and Service Management* (CNSM 2013), pp.18–25, 2013.

[18] A. Akella, and A. Krishnamurthy, "A Highly Available Software Defined Fabric", in *Proc. of 13th ACM Workshop on Hot Topics in Networks* (HotNets-XIII), pp. 21, October 2014.

[19] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks", *IEEE Transactions on Network and Service Management*, Vol. 12, No. 1, pp. 4–17, March 2015.

[20] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network Reliability", *Technical Research Report TR 92-74*, June 1992.

[21] C. Swamy, and D. B. Shmoys, "Fault-Tolerant Facility Location", *ACM Transactions on Algorithms*, Vol. 4, No. 4, pp. 51:1–51:27, August 2008.

[22] T. B. Brecht, "Lower Bounds for Two-Terminal Network Reliability", *Discrete Applied Mathematics*, Vol. 21, pp. 185–198, 1985.

[23] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo", *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 9, October 2011.

[24] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California Fault Lines: Understanding the Causes and Impact of Network Failures", in *Proc. of the ACM SIGCOMM 2010*, pp. 315–326, 2010.

[25] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications", in *Proc. of the ACM SIGCOMM 2011*, pp. 350–361, 2011.