Five Nines of Southbound Reliability in Software-Defined Networks

Francisco J. Ros
University of Murcia
E-30100 Facultad de Informatica
Murcia, Spain
fjros@um.es

Pedro M. Ruiz
University of Murcia
E-30100 Facultad de Informatica
Murcia, Spain
pedrom@um.es

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 present our initial work 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 at least five nines reliability in the southbound interface between controllers and nodes. For this, we introduce 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. We find that each node is required to connect to just 2 or 3 controllers, which typically provide more than five nines reliability. While the total number of controllers varies greatly and is more related to the network topology than to the network size, 10 controllers or less cover 75% of the most interesting cases. Therefore, fault tolerant SDNs are achievable by carefully determining the placement of controllers.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Performance of Systems**]: Design studies, Fault tolerance

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. While traditional network nodes (switches, routers, middleboxes) integrate both planes within the same device, the SDN 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 [8] while others have been designed with clustering as an integral feature from the beginning [1]. 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 [10]). However, the random placement of an arbitrary number of controllers does not provide satisfactory performance [6], 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* they are deployed, and *what 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 and synchronization overhead among controllers. In general, we should also strive to optimize the overall number of controllers in the network, since each one has an inherent cost in terms of capex, opex, and state synchronization with the rest of controllers.

HotSDN'14, August 22, 2014, Chicago, IL, USA.

ACM 978-1-4503-2989-7/14/08 ...$15.00.
http://dx.doi.org/10.1145/2620728.2620752.

In this paper, we focus on building reliable software-defined networks and introduce the *fault tolerant controller placement* (*FTCP*) problem (§2). In our formulation, each node must meet a reliability constraint so 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 (§3) 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. We exploit such algorithm to explore placements that lead to at least **five nines** southbound reliability in 124 publicly available network topologies (§4). In the evaluated scenarios, we found that each node is required to connect to just 2 or 3 controllers, which typically provide more than five nines reliability. As expected, the total number of controllers varies greatly and is more related to the network topology than to the network size. Nonetheless, 10 controllers or less are enough for 75% of the most interesting cases. Therefore, fault tolerant SDNs are achievable by carefully choosing controller placements within the target network topology. We conclude this paper by reviewing related works (§5) and discussing the main results of our analysis (§6).

2. THE FTCP PROBLEM

2.1 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}.

For every node j, it 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 [2], 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** 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* problem [12], which belongs to the NP complexity class. In addition, we have introduced a non-linear constraint on the network reliability R. Computing R is an NP-hard problem[1] by itself, so for the remainder of this paper we consider a lower bound $\hat{R}$ on network reliability that can be obtained in polynomial time.

2.2 Reliability lower bound

G can be applied a series of reliability-preserving transformations [2] 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 [3].

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*[2] between j_0 and t in G'). 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:

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

In order to find a set $\Pi_{j_0,t}$, we make G' a *flow network* by decorating edges with a capacity attribute (see Figure 1). In our case, j_0 is the source and t is the sink of the flow. The source is allowed to issue as much flow as possible, while facilities can only receive a large constant amount of flow $\Gamma \gg 1$. The remaining edges either have capacity 1 or ∞, depending on whether they are derived from an original edge in G or are a virtual construct defined by G', respectively. By solving the *max-flow* problem on the capacitated network G', we find a subset of paths that contribute to network reliability. Among them, $\Pi_{j_0,t}$ consists of those without edges in common.

[1]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 least) a path from j to I_j, and computing the probability of each state. Refer to [2] and the references therein for a more elaborated discussion.

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

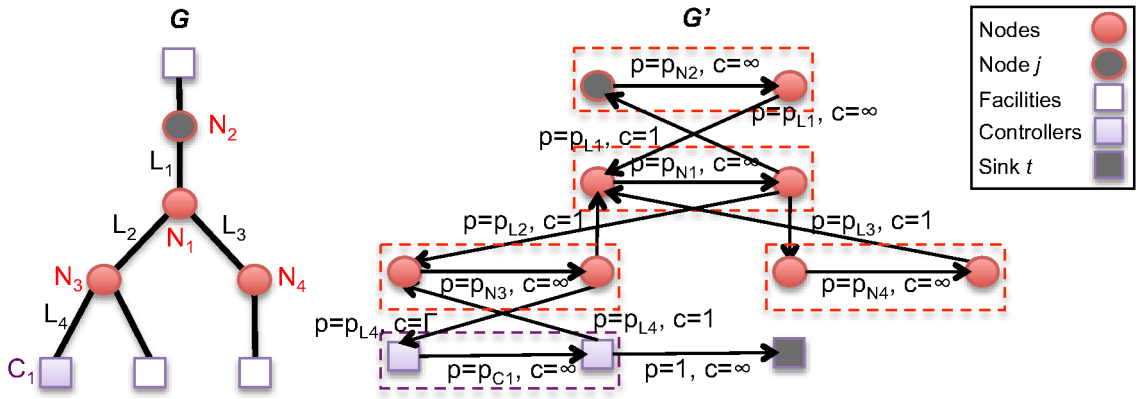

Figure 1: Topology transformation from original graph G to flow network G' with reliable nodes.

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

3. 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 can be seen as a worst-case analysis of 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 (line 2). After initializing some variables (line 3), 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 7-11). 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 12-16). The scores of all facilities in I_j are multiplied by τ to try to reuse the same controllers for subsequent nodes (line 17).

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 $\quad I_j \leftarrow \emptyset$; $\hat{R}_{prev} \leftarrow 0$;
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 $\quad\quad remove\_edge(G', u, v), \forall (u, v) : f(u, v) = 0$;
10 $\quad\quad \Pi_{j_0,t} \leftarrow \lambda$ disjoint paths in G';
11 $\quad\quad \hat{R} \leftarrow 1 - \prod_{\forall \pi \in \Pi_{j_0,t}} \left(1 - \prod_{\forall (u,v) \in \pi} p_{u,v}\right)$;
12 $\quad\quad$ **if** $\hat{R} > \hat{R}_{prev}$ **then**
13 $\quad\quad\quad$ **if** $\hat{R} \geq \beta$ **then**
14 $\quad\quad\quad\quad I_j \leftarrow \{k \in I : (\exists \pi \in \Pi_{j_0,t} : k \in \pi)\}$;
15 $\quad\quad\quad\quad$ break;
16 $\quad\quad\quad \hat{R}_{prev} \leftarrow \hat{R}$
17 $\quad S_i \leftarrow \tau S_i, \forall i \in I_j$;

4. ANALYSIS OF 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 net-

	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.

work 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 five nines reliability, we want to answer the following questions: *How 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.

4.1 Evaluation setup

The network topologies under consideration have been obtained from a publicly available repository, namely the Internet Topology Zoo [7]. From the set of networks reported in [7] 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 up to k controllers. Therefore, every node links with k facilities in the topology graph G. For the remainder of this paper we fix $k = 2$, although experiments with different values yielded similar results.

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 [13], higher reliability for nodes [5], and lower reliability for higher layer devices [5].

Regarding the incentive parameter τ employed by Algorithm 1, we experimented with values 2 and 5. In this paper we report the results obtained with the best of these configurations with respect to the total number of required controllers. An exhaustive parameter sensitivity analysis is out of the scope of this paper, but we note that (as expected) higher τs provide solutions with less overall controllers.

Since we look for five nines reliability, we fix $\beta = 0.99999$. For each configuration we run twenty independent experiments with different seeds.

4.2 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. A non-reliable network, i.e. a network without redundant paths, requires many more controllers than a redundant network of similar size. 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 (Figure 2), and those that require more than half the number of nodes (not shown due to space constraints). Topologies in the X axis are ordered from lowest to highest size (number of nodes). For each topology, the overall number of controllers is shown as a boxplot[3]. This figure contains the most interesting scenarios, given that it is expensive (in terms of controllers) to achieve southbound reliability in an inherently unreliable network.

Mode	Median	75-percentile	Mean	Std. Dev.
5	6	10	8.2	7

Table 2: Statistics for the total number of controllers.

As an example of a small redundant network, let us consider the `Sprint` topology that features 11 nodes. In most cases, three controllers suffice to provide five nines reliability to every node. One of the controllers must be placed on a central PoP with high connectivity, such as Washington or Stockton. An additional controller should go to a PoP that has different paths to as many nodes as possible. Kansas and Cheyenne are common candidates for this. Finally, a controller devoted to provide the only node without redundant paths to the rest of the network (Boulder) with high reliability is required (it might be located in Boulder itself or Stockton, depending on the other placements).

Bigger networks like `BtNorthAmerica` (36 nodes) can also meet the reliability constraint with just 3 controllers in some cases (up to 6 depending on the individual operational probabilities of links, nodes and controllers). On its hand, the largest evaluated network (`Cogentco`, 197 nodes) requires 51.7 controllers on average. This suggests that parts of the network are not protected enough against failures through redundant paths. This aligns with the data reported in [7], where this network is shown to have an average node degree (2.5) close to the average degree of a tree (2) of the same size. Hence, localized redundancy allows for a reduction on the number of controllers, although the overall amount remains high because of the different rings that form the network. This might be an issue for some applications if strong consistency of network state is required among controllers. Although such concern requires further investigation to be clarified, we note that an intercontinental network like this would likely require multiple controller domains for management and operational reasons. In this way, strong consistency between domains might be traded off for better liveness [9].

Overall, in network topologies with certain degree of redundancy, the number of controllers required for highly reliable southbound connections remains below a reasonable level (Table 2). In fact, 75% of all the evaluated scenarios require 10 controllers or less. On average, 8.2 controllers are needed, despite the dispersion of the data is high (due to `Cogentco`'s corner case).

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) [7], so that 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 `Itnet` network has a star topology and it usually needs as many controllers as nodes for five nines southbound reliability.

[3]Whiskers extend to 1.5 times the interquartile range. Outliers are shown individually as points.

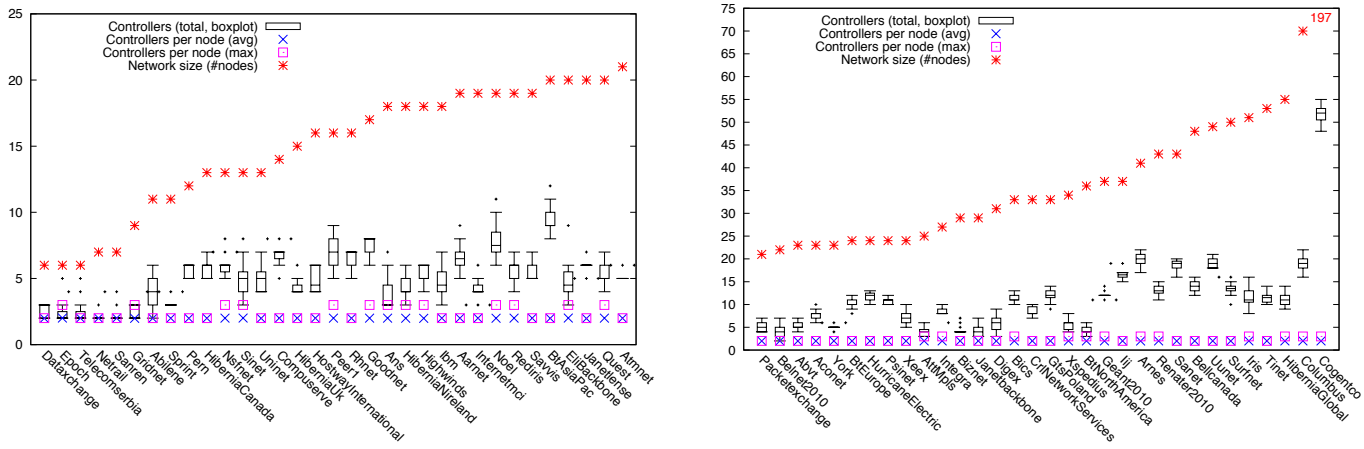

Figure 2: Overall number of controllers (boxplot) and controllers per node (average, maximum) for topologies requiring on average less or as many controllers as half the number of nodes.

$\hat{R}$	5 nines	6 nines	7 nines	$\geq$ 8 nines
Nodes (%)	2.26%	44.38%	47.15%	6.21%

Table 3: Percentage of nodes that can find an operational path towards any of their connected controllers with different reliabilities.

4.3 Southbound connections

Regardless the topology, each node only needs to establish a connection with two controllers on average (Figure 2). At most, one node connects to three controllers. Note that the same is true for the remainder of the evaluated topologies not shown in the figure.

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.

5. 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 [9]. Onix [8] provides applications with flexibility to partition the network state and apply different storage mechanisms. To reduce synchronization overhead among controllers, some works propose to organize them into hierarchical layers [8, 14]. In addition, controller load can be dynamically shifted across controllers [4].

The closest work to the one performed in this paper is due to Heller *et al.* [6]. They introduce the *controller placement* problem and explore the trade-offs when optimizing for minimum latency. 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.

To the best of our knowledge, our paper is the first work that addresses the controller placement problem from a network reliability perspective. The literature on network reliability is extensive [2], although not related to SDN.

6. DISCUSSION AND CONCLUSION

When nodes get disconnected from controllers, SDN applications lose visibility and control 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.

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 five nines reliability. In addition, such number is a worst case scenario since our proposed algorithm works on a lower bound on southbound reliability. In practice, such bound is at least one order of magnitude higher for most nodes, given the placements found by our algorithm. Such placements put a very light load on network nodes, requiring three connections at most.

In our evaluation we assumed in-band control. However, the proposed algorithm can also be applied for out-of-band control networks. In any case, we note that naive out-of-band control may actually provide lower resilience to failures [11]. The work in this paper can help prospective SDN operators mitigate their concerns about having a logically centralized control plane in their networks.

7. ACKNOWLEDGMENTS

This work has been partially funded by the "Technologies for Cyber-Physical Systems" Strategic Action for Preferential Research of the University of Murcia.

8. REFERENCES

[1] Opendaylight controller: Architectural framework. on-line.

[2] M. O. Ball, C. J. Colbourn, and J. S. Provan. Network reliability. Technical Research Report TR 92-74, June 1992.

[3] T. B. Brecht. Lower bounds for two-terminal network reliability. *Discrete Applied Mathematics*, 21:185–198, 1985.

[4] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella. Towards an elastic distributed sdn controller. In *Proc. of HotSDN'13*, pages 7–12, 2013.

[5] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proc. of the ACM SIGCOMM 2011*, pages 350–361, 2011.

[6] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *Proc. of HotSDN'12*, pages 7–12, 2012.

[7] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9), October 2011.

[8] 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 OSDI'10*, pages 1–6, 2010.

[9] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically centralized? state distribution trade-offs in software defined networks. In *Proc. of HotSDN'12*, pages 1–6, 2012.

[10] Open Networking Foundation. Openflow switch specification (version 1.2.0), December 2011.

[11] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker. Cap for networks. In *Proc. of HotSDN'13*, pages 91–96, 2013.

[12] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. *ACM Transactions on Algorithms*, 4(4):51:1–51:27, August 2008.

[13] 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*, pages 315–326, 2010.

[14] S. H. Yeganeh and Y. Ganjali. Kandoo: A framework for efficient and scalable offloading of control applications. In *Proc. of HotSDN'12*, pages 19–24, 2012.