Proceedings of the 2013 25th International Teletraffic Congress (ITC)

Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks

David Hock, Matthias Hartmann, Steffen Gebert, Michael Jarschel, Thomas Zinner, Phuoc Tran-Gia
University of Würzburg, Institute of Computer Science, Würzburg, Germany
Email: {david.hock,hartmann,steffen.gebert,michael.jarschel,zinner,trangia}@informatik.uni-wuerzburg.de

***Abstract*—With the introduction of Software Defined Networking (SDN), the concept of an external and optionally centralized network control plane, i.e. controller, is drawing the attention of researchers and industry. A particularly important task in the SDN context is the placement of such external resources in the network. In this paper, we discuss important aspects of the controller placement problem with a focus on SDN-based core networks , including different types of resilience and failure tolerance. When several performance and resilience metrics are considered, there is usually no single best controller placement solution, but a trade-off between these metrics. We introduce our framework for resilient Pareto-based Optimal COntroller-placement (POCO) that provides the operator of a network with all Pareto-optimal placements. The ideas and mechanisms are illustrated using the Internet2 OS3E topology and further evaluated on more than 140 topologies of the Topology Zoo. In particular, our findings reveal that for most of the topologies more than 20% of all nodes need to be controllers to assure a continuous connection of all nodes to one of the controllers in any arbitrary double link or node failure scenario.**

***Keywords*—SDN, Software Defined Networks, OpenFlow, Controller Placement, Latency, Resilience, Failure Tolerance**

I. Introduction

The introduction of Software Defined Networking (SDN) has caused a paradigm shift in communication networks. The SDN concept allows separating control and data plane, namely moving complex functions from devices in a network to sophisticated dedicated controller instances. The most popular example of SDN is OpenFlow [1], where a central OpenFlow *controller* defines rules for switches how to handle packets, thus enabling a centralized routing approach. With the HyperFlow [2] concept, OpenFlow networks can be separated into several domains, each with their own controller. This facilitates load balancing and resilience in the SDN infrastructure. Recently, the term Network Functions Virtualization (NFV) [3] has been introduced to describe an approach that externalizes not only the control plane but also functions of the data plane, e.g. load balancing or firewalling, to virtual appliances running on commodity hardware. In this work, we analyze two crucial issues for external control architectures: i) how many controllers are necessary in a network and ii) where to place them for an adequate trade-off between latency and resilience?

This work has been performed in the framework of the CELTIC EUREKA project SASER-SIEGFRIED (Project ID CPP2011/2-5), and it is partly funded by the BMBF (Project ID 16BP12308). The authors alone are responsible for the content of the paper.

Heller et al. [4] indicated that the topic of general controller placement is well explored and no new *theoretical* insights are expected. In particular, the very basic version of controller placement according to the latency of nodes to their controller, also known as *facility* or *warehouse location problem*, is a typical example for a Mixed Integer Linear Program (MILP) provided e.g. with the *IBM ILOG CPLEX* [5] software. Most work on the topic of controller placement in literature concentrates on the fact that the problem is NP-hard and depending on the complexity of the particular considered objective often provides only approximations to solve it. Heller et al. showed that finding optimal solutions is computationally feasible for realistic network instances and failure-free scenarios, by analyzing the entire solution space using "weeks of computations" on today's CPUs. Thus, they could address and optimally answer the question posed before but without considering failure tolerance. They revealed that in most topologies one single controller is enough to fulfill "existing reaction-time requirements". We extend their controller placement analysis to include different resilience aspects that are important in the context of SDN and NFV. In particular, we show that in most topologies, where a single controller would be enough from a latency point-of-view, many more controllers are necessary to meet resilience requirements. We also take into account inter-controller latency, load balancing between controllers, and trade-off considerations between latency and failure resilience. We implemented a Matlab-based framework to compute resilient Pareto-based Optimal COntroller-placements called POCO. It makes an efficient combined use of CPU and RAM, so it can evaluate the entire solution space even when resilience is considered. The advantage of this approach compared to any particular MILP or heuristic is that evaluating the entire solution space naturally provides information for all considered objectives for all placements. Regarding multi criteria/multi objective optimization that means that no decision has to be taken before invoking the optimization by defining some constraints or weighted objective functions. In contrast, offering all possible solutions evaluated by all objectives, offers the possibility to take the decision afterwards. In literature, there also exist different approaches for multi criteria facility location for a given combination of certain objectives. However, the approach of evaluating the entire solution space offers – for realistic network sizes – the most freedom to consider different objectives. With POCO we do *not* offer a recipe to solve all instances of the problem of any size, but we can find optimal solutions for realistic network sizes. Our POCO-toolset *[http://www3.informatik.uni-wuerzburg.de/poco]* is able to solve the problems within acceptable computation time.

978-0-9836283-7-8

The remainder of this work is organized as follows. In Section II, we give a brief introduction to SDN as well as different SDN scenarios, and illustrate different challenges with central controllers and how good placements can alleviate or avoid these problems. Section III investigates the resilience against different possible network failures and their impact on the controller placement. Section IV discusses how to balance the load among different controllers and how to account for inter-controller latencies. In Section V, we give an overview on related work and conclude the paper in Section VI. The appendix contains an overview of the abbreviations and symbols used in this work in Table I.

II. Scenarios and Problem Description

In this section, we briefly explain SDN and different SDN scenarios. Then, we show several aspects that have to be considered for a resilient controller placement in SDN networks.

A. SDN and SDN Scenarios

One of the key ideas of Software Defined Networking is to externalize the network control plane. Functionality, such as routing, is taken out of the data plane and moved towards external controllers. Depending on the particular use case, these controllers can either be realized in hardware or as pure software components. There can be a single controller or a set of controllers communicating with each other. Controller communication architectures can be either flat, with each controller having the same role, hierarchical with normal controllers and coordinating "master controllers", or follow a variety of different approaches. The connection between the nodes and their controllers can be realized inband or outband, i.e. using the same physical connections or dedicated lines.

The concept of SDN can be used for a variety of use-cases, some of which we cannot even imagine, yet. Use cases that go beyond network topology planning are for example monitoring, load balancing, or traffic engineering. The idea of Network Functions Virtualization (NFV) is to virtualize different network functionalities, such as monitoring, firewalling, or security aspects. SDN can serve as an enabler for this approach, e.g. by providing certain functionalities on central controllers. This paper focuses on the use case of SDN in core networks as well as a possible extension towards NFV. We assume that usually in core networks primary paths are pre-installed for traffic aggregated between different nodes in the network. SDN controllers are thus not contacted for each single flow, but only in case of traffic engineering actions or in case of outages to find adequate backup paths. Therefore, depending on the network size a single controller might be able to control the entire network without being overloaded. However, for resilience issues, more than one controller is necessary. If NFV functionalities are based on an SDN control platform, the number of necessary controllers is significantly higher due to the heavy load on the control plane. We assume that in SDN core networks, controllers are co-located with regular network nodes. The signaling between nodes and controllers is done in-band, in the sense of "in the same physical network". That means that if the network is physically disconnected in several parts, nodes and controllers in different parts of the network cannot contact each other anymore.

B. Resilient Controller Placement for SDN

A main objective for a good controller placement is to minimize the latencies between nodes and controllers in the network. However, looking only at delays is not sufficient. A controller placement should also fulfill certain resilience constraints. To illustrate this, we look at the best controller placement with $k = 5$ controllers in the Internet2 OS3E topology according to maximum latency as shown by Heller et al. [4]. Figure 1 shows four different illustrations of the same placement to depict issues to be considered when judging the resilience of a placement. In the following, we briefly explain these issues and what is necessary to be resilient against them.

1) Controller Failures: As illustrated by Heller et al. [4], a larger number of well-distributed controllers in a network can help to lower the maximum latency between the nodes and their controllers. It also increases the failure tolerance if some of the controllers stop working. Zhang et al. [6] assume in their work that a node is not able to route anymore if it loses its connection to the controller. However, we suppose that in case of a controller outage, it is possible to reassign all nodes previously attached to that controller to their second closest controllers in the network using a backup assignment or signaling based on normal shortest path routing. Thus, as long as at least one of the controllers is still reachable, all nodes keep being functional. However, the latencies of the reassigned nodes to their new controller can be significantly higher than the latencies to the primary controller. Figure 1(a) illustrates the latencies of all nodes to the last remaining controller in case of an outage of all other four controllers using a traffic light color scheme. The color changes from pure green indicating a latency of zero to yellow indicating 50% of the network's diameter to pure red indicating 100% of the diameter. The displayed controller failure scenario corresponds to the worst case because the remaining controller is the one located the furthest from the center of the network. Thus, the requests of some of the nodes need to pass almost through the entire network to reach the controller. To increase resilience against this phenomenon, the controller placement optimization should not only consider the latencies during failure-free routing, but also worst case latencies during controller failures. This problem is addressed in Section III-A.

2) Network Disruption: In contrast to controller failures, the outage of network components, such as links and nodes, often has a much higher impact on the network stability, as it alters the topology itself. The shortest paths between some of the nodes change, leading to different latencies and possibly to the reassignment of nodes to other controllers. Even more severe is that entire parts of the network are in danger of being cut off by link or node outages. In the worst case, some nodes can no longer be connected to a controller as they are cut off from all controllers. These nodes are still working and able to forward traffic, but cannot request instructions anymore. Figure 1(b) illustrates the worst possible scenario for double node failures. All nodes depicted with a white question mark icon ⍰ are controller-less, i.e. still working but cannot reach any controller. Hence, the entire subnetwork consisting of these controller-less nodes is no longer able to address any functionality realized by the controller despite the fact that the nodes are still working. Rerouting flows to working paths is no longer possible, even though some of the nodes

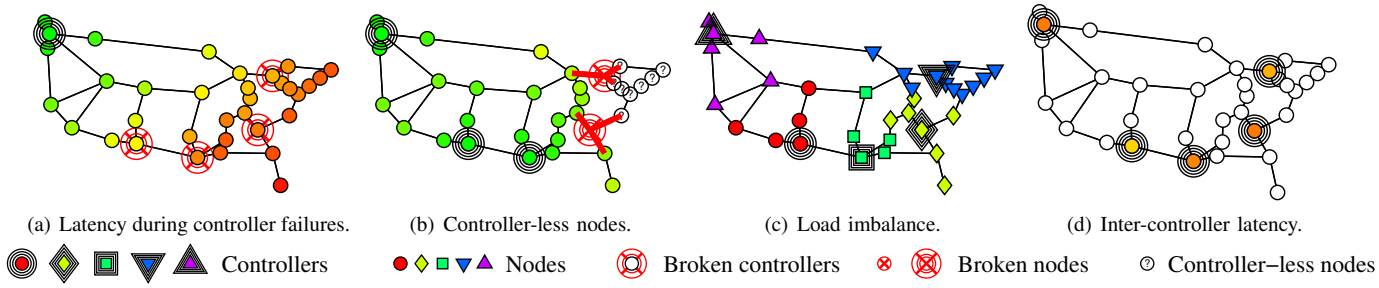


Fig. 1. Illustration of different issues to be considered when judging the resilience of a controller placement.

are still physically connected. In Section III-B, we address the problem of controller-less nodes and network component failure tolerance.

3) Load Imbalance: Analog to [4], we assume that nodes are always assigned to their nearest controller using latency as metric, i.e. the shortest path $d(n, c)$ between the node n and controller c. Figure 1(c) uses different markers and colors to illustrate the node-to-controller assignment. The number of nodes per controller is imbalanced and ranges from 4 (,) to 10 ( ). The more nodes a controller has to control, the higher is the load on that controller. This is especially relevant in scenarios where nodes communicate often with their controler, e.g. when considering NFV. If the number of node-to-controller requests in the network increases, so does the chance of additional delays due to queuing at the controller system.

To be resilient against controller overload, the assignment of nodes to the different controllers should be well-balanced. In [7], we demonstrated that controller performance can vary among connected switches. This stresses the importance of an intelligent controller placement that also takes load balancing aspects into account. We address this problem in Section IV-A.

4) Inter-Controller Latency: It is clear that a single controller is not enough to reach any kind of resilience in a network. However, when several controllers are placed in the network, another issue arises that we briefly address in Section IV-B. If the control logic of the network is distributed over several controllers, these controllers need to synchronize to maintain a consistent global state. Depending on the frequency of the inter-controller synchronization, the latency between the individual controllers plays an important role. Figure 1(d) illustrates the maximum controller-to-controller latencies using the same traffic light color scheme relative to the network diameter as in Figure 1(a). The color of each controller indicates the maximum distance of this controller to all others. For the depicted placement, the messages between the controllers have to travel relatively long distances in the network, which might not be acceptable.

III. Failure Tolerant Controller Placement

In this section, we include resilience against different failures into the controller placement and calculate optimal results using our POCO framework. We first discuss the inclusion of resilience against controller failures, then the inclusion of resilience against network element failures.

A. Controller Failure Tolerance

When a controller fails, some nodes are reassigned to other controllers and experience increased latencies. To quantify the latencies in a network, we take the maximum over all node-to-controller-latencies and denote it with $\pi^{\text{max latency}}$. Similar as presented in [4], based on a matrix $d_{v,w}$ containing the shortest path distances between all nodes v and w of the set of all nodes $\mathcal{V}$, the maximum node-to-controller latency for a placement of controllers $\mathcal{P} \in 2^{\mathcal{V}}$ can be defined as

$$\pi^{\text{max latency}}(\mathcal{P}) = \max_{(v \in \mathcal{V})} \min_{(p \in \mathcal{P})} d_{v,p} \tag{1}$$

We do not consider the average but the maximum latency, because an average hides the worst case values that are important when resilience should be improved. All latency values are calculated relative to the diameter of the network in the failure free case (% of diameter). In the failure free case, the maximum latency is denoted as $\pi_{\emptyset}^{\text{max latency}}$. For a placement of k controllers, we construct a set of scenarios $\mathcal{C}$ that includes all possible combinations of up to $k-1$ controller failures (including the failure free case) and denote the resulting maximum latency with $\pi_{\mathcal{C}}^{\text{max latency}}$.

$$\pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) = \max_{(s \in \mathcal{C})} \pi_{s}^{\text{max latency}}(\mathcal{P}) \tag{2}$$

The node-to-controller assignments change in case of controller outages. Therefore, for the metric $\pi_{\mathcal{C}}^{\text{max latency}}$, not only the distance to the (primary) controller in the failure free case but also the distance to the other (backup) controllers in case of failures is included in the metric. Let the placements $\mathcal{P}_1$ be all these subsets of working controllers of a placement $\mathcal{P}$ that can appear for the considered controller failure scenarios $\mathcal{C}$. Then, $\pi_{\mathcal{C}}^{\text{max latency}}$ can also be obtained as

$$\pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) = \max_{(v \in \mathcal{V})} \max_{(\emptyset \subset \mathcal{P}_1 \subseteq \mathcal{P})} \min_{(p \in \mathcal{P}_1)} d_{v,p} \tag{3}$$

When controllers are placed in the network so that $\pi_{\emptyset}^{\text{max latency}}$ is minimized, the intuitive result is a placement where controllers are equally spread in the network. On the other hand, when placing several controllers to reach the best possible $\pi_{\mathcal{C}}^{\text{max latency}}$ even in the worst failure cases, all controllers tend to be in the center of the network. Thus, even if all except for one controller fail, the latencies are still satisfying. Figure 2 shows the optimal placement $\mathcal{P}_{\mathcal{C}}^{\text{max latency}}$ for $k = 5$ controllers. The corresponding minimal maximum

latency is denoted as $\underline{\pi}_{\mathcal{C}}^{\text{max latency}}$. For a given number of k controllers, it can be obtained as:

$$\underline{\pi}_{\mathcal{C}}^{\text{max latency}} = \min_{\mathcal{P}\in\{\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_{\binom{|\mathcal{V}|}{k}}\}} \pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) \qquad (4)$$

As expected, the controllers are all situated close to the center of the network. Even in the worst case situation shown in Figure 2, where all but one controller fail, the maximum latencies are still relatively low. This particular placement with $k = 5$ controllers actually corresponds to a combination of the five best placements of single controllers. The worst case is reached when 4 out of 5 controllers fail and only the fifth best single controller placement is still active. In general, when considering up to $k-1$ simultaneous controller failures, i.e., in particular, all subsets $\mathcal{P}_1$ of $\mathcal{P}$ containing only a single element, the value of $\pi_{\mathcal{C}}^{\text{max latency}}$ corresponds to the distance to the k-closest controller in the failure free case. Therefore, another shorter (and easier to calculate) definition of $\pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P})$ can be given for this case:

$$\pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) = \max_{(v\in\mathcal{V})} \max_{(p\in\mathcal{P})} d_{v,p} \qquad (5)$$

Even though the maximum latencies are still relatively low even in the case of $k-1$ simultaneous controller outages, this centralized placement with low $\pi_{\mathcal{C}}^{\text{max latency}}$ leads to an increase of $\pi_{\emptyset}^{\text{max latency}}$ compared to the best placement $\mathcal{P}_{\emptyset}^{\text{max latency}}$ optimized for the failure free scenario with an equal number of controllers.

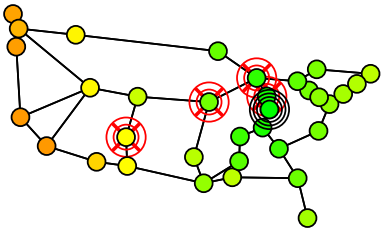


Fig. 2. Impact of worst-case failure scenario (4 out of 5 controllers fail) on node-to-controller latency.

There is a clear trade-off between the placements optimized for the failure free case and those including controller failure resilience. To look at that trade-off in more detail, we show all possible placements and their quality according to the failure free and the resilient case for the metric $\pi^{\text{max latency}}$ in Figure 3. Each point in the graph indicates one placement. The x-value of a point indicates the $\pi_{\emptyset}^{\text{max latency}}$ value of the corresponding placement, the y-value the $\pi_{\mathcal{C}}^{\text{max latency}}$ value. As mentioned before, we regard the outage of up to all but one controller. The dashed lines indicate the mean values for the corresponding metric over all placements.

For better visibility, the axes limits of the graph have been adapted to display only the most important range of the placement solution space. To give an impression, the worst placements have values of $\pi_{\emptyset}^{\text{max latency}} > 85\%$ of the diameter.

Out of all possible placements, our POCO-framework returns only the Pareto-optimal placements, which are indicated in Figure 3 with black points connected with a black line. For

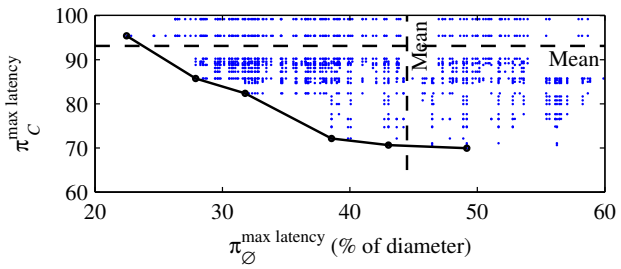


Fig. 3. Trade-off between failure free and controller failure values.

a number of n metrics $m_1,\ldots,m_n$, a value $(x_1,\ldots,x_n)$ is Pareto-optimal if and only if there is no value $(y_1,\ldots,y_n)$ with y_i better x_i for all metrics m_i. In Figure 3, there is no single optimal placement with best possible values for both the failure-free case and the controller failure case. In contrast, Pareto-optimal placements that perform better in the resilient case perform worse in the failure free case and vice versa. Thus, POCO usually gives no recommendation for a particular placement, but returns the set of Pareto-optimal placements, which enables the network operators to choose the placement that fits their needs best. In particular, they can also decide up to how many controller failures should be covered by a resilient placement.

To verify that our observation also holds for other topologies, we evaluated the trade-off between the optimal placements $\mathcal{P}_{\emptyset}^{\text{max latency}}$ and $\mathcal{P}_{\mathcal{C}}^{\text{max latency}}$ for all 146 topologies[1] in the Topology Zoo collection [8] with a size of (at least 5 and) up to 50 nodes for $k = 5$ controllers. Figure 4 shows the distribution of the worst values $\pi^{\text{max latency}}$ over all topologies for 0, 2, and 4 failures respectively (from left to right). The results confirm our findings. Without failures (green lines), $\mathcal{P}_{\emptyset}^{\text{max latency}}$ achieve the lowest delays, as they are optimized for this case. But when four controllers break down simultaneously (red lines), such placements lead to high latencies in a very large fraction of the examined topologies. As expected, the placements $\mathcal{P}_{\mathcal{C}}^{\text{max latency}}$ that are optimized for up to four controller failures provide far lower latencies. But also when only two out of five controllers fail (blue lines), the controller placements that are optimized for up to four failures yield lower latencies than the placements optimized for the failure free case.

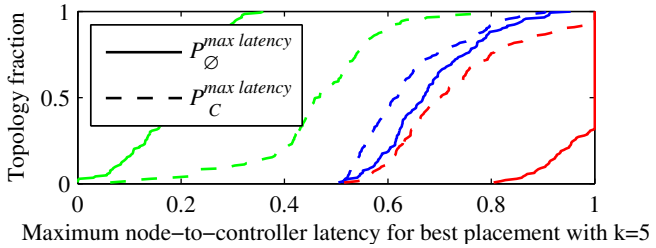


Fig. 4. Worst values $\pi^{\text{max latency}}$ for different numbers of controller failure (0:green, 2:blue, and 4:red) for the Topology Zoo topologies considering the optimal placements with $k = 5$ controllers.

[1]If several versions of the same topology exist, only the most recent one was considered.

B. Network Disruption Tolerance

In the following, we address the issue of link and node failures in a network and the resulting risk of a network disruption. As shown in Section II-B, a simultaneous outage of two nodes in the Internet2 OS3E topology can lead to up to eight controller-less nodes when the placement of $k = 5$ controllers is done only according to $\pi_{\emptyset}^{\text{max latency}}$. As explained before, a node is considered controller-less if it is still working and part of a working subtopology (consisting of at least one more node), but cannot reach any controller. Nodes that are still working, but cut off without any working neighbors, are not considered to be controller-less. This seems a valid assumption, as during this failure situation these nodes have nobody to communicate with. Furthermore, without this relaxation, each node having at most two neighbors would have to contain a controller to not be controller-less when both neighbors fail.

Formally, we define $\pi_{\mathcal{X}}^{\text{controller-less}}$ as the maximum number of controller-less nodes appearing for a certain placement when considering all failure scenarios of $\mathcal{X}$. Let matrices $e_{v,w}^{s}$ be disconnection matrices with $e_{v,w}^{s_i} = 1$ if and only if in failure scenario s_i, node v cannot reach node w. All other entries of matrix $e_{v,w}^{s}$ are 0. Using this, we define $\pi_{\mathcal{N}}^{\text{controller-less}}$ as

$$\pi_{\mathcal{N}}^{\text{controller-less}}(\mathcal{P}) = \max_{(s \in \mathcal{S})} \sum_{(v \in \mathcal{V})} \min_{(p \in \mathcal{P})} e_{v,p}^{s} \tag{6}$$

Given these assumptions, the lowest possible value $\pi_{\mathcal{N}}^{\text{controller-less}}$ that can be reached by a placement of $k = 5$ controllers in the Internet2 OS3E topology when considering up to two node failures[2] is two. A placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$ leading to this value is shown in Figure 5.

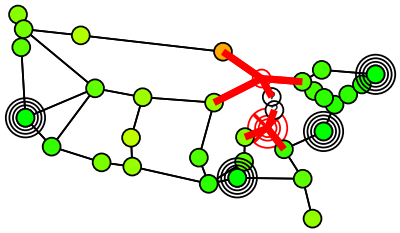
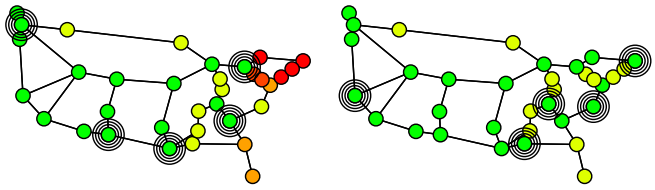


Fig. 5. Illustration of the optimal placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$ minimizing the number of controller-less nodes in at most two node-failures.

To illustrate $\pi_{\mathcal{N}}^{\text{controller-less}}$ for different controller placements, we color the topology to indicate how often a certain node is controller-less when considering all different failure scenarios. We use a traffic-light scheme, where green depicts that a node is never controller-less in any scenario and red that it is controller-less in more than three failure scenarios. In between the values for green and red, a logarithmic scale is applied. Figure 6 shows the described color scheme for two placements $\mathcal{P}_{\emptyset}^{\text{max latency}}$ and the placement from Figure 5, $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$.

In previous considerations, each node is considered to be of equal importance and $\pi^{\text{controller-less}}$ just indicated the maximum

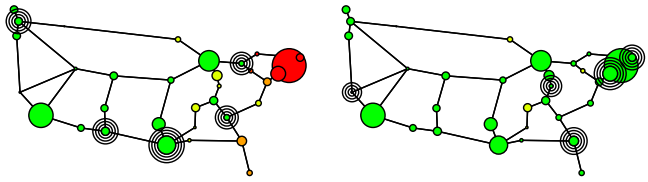
[2]We limit the question to two simultaneous failures for two reasons: First, more simultaneous failures are unlikely to happen. Second, if more than two arbitrary failures happen in the same time, the topology can be totally disrupted so that basically no controller placement would help here anymore.



(a) Placement $\mathcal{P}_{\emptyset}^{\text{max latency}}$ (b) Placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$.

Fig. 6. Illustration for different placements of how often a certain node is controller-less when considering all failure scenarios.

number of simultaneously controller-less nodes. However, depending on the concrete application of SDN and centralized controllers, different nodes may account for a different amount of signaling with the controller or are of different importance. To include this, we assign each node of the Internet2 OS3E example topology with a weight according to the population of the city where the node is located. The population values are obtained using *Wolfram Alpha* [9]. Figure 7 shows the same illustrations as Figure 6 but with population sizes included in the optimization as importance of certain nodes. The resulting best placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$ differs from the one obtained when counting all nodes with uniform weight 1.



(a) Placement $\mathcal{P}_{\emptyset}^{\text{max latency}}$ (b) Placement optimized.

Fig. 7. Illustration for different placements of how often a certain node is controller-less considering all failure scenarios including population sizes.

Obviously, $\pi^{\text{controller-less}}$ can be reduced with increasing number of controllers in a network. Figure 8 shows for the consideration of one and two failures how $\pi_{\mathcal{N}}^{\text{controller-less}}$ decreases with increasing controller number k for the best placements $\mathcal{P}$ according to $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\mathcal{N}}^{\text{controller-less}}$. It can be seen that in the Internet2 OS3E topology with a number of $k = 7$ controllers it is possible to eliminate all controller-less nodes in all one and two failure scenarios.

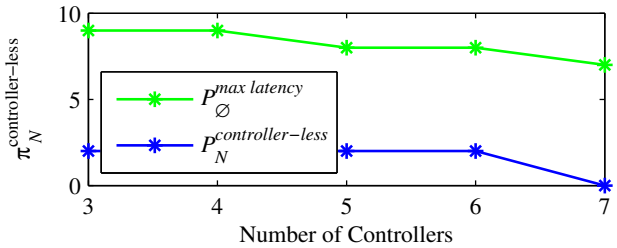


Fig. 8. Controller-less nodes decrease with higher number of controllers k.

This raises the general question what the minimum number of controllers k and their placement is to eliminate the occurrence of controller-less nodes for up to two link and node failures. Each subtopology consisting of at least two nodes that can be cut off from the entire network by at most two link- or node-failures has to be covered by at least one controller. Placing a controller in a subtopology automatically

also covers all larger subtopologies that include the smaller one. This allows to develop a procedure to calculate k for a topology without evaluating $\pi_{\mathcal{N}}^{\text{controller-less}}$ for all possible placements. Without this trick, the computational effort to find an optimal k could not be handled. The minimum number of controllers can be found as follows. First, find the set of all possible vulnerable subtopologies of at least two nodes that are not supertopology of any other topology in the set. In Figure 9, there are eight such subtopologies, so the *maximum* necessary number is $k = 8$. Second, find the minimum number of controllers necessary to cover all subtopologies. In this case, in the right hand side of the topology, two controllers are enough to cover the three overlapping subtopologies, thus $k = 7$ is enough.

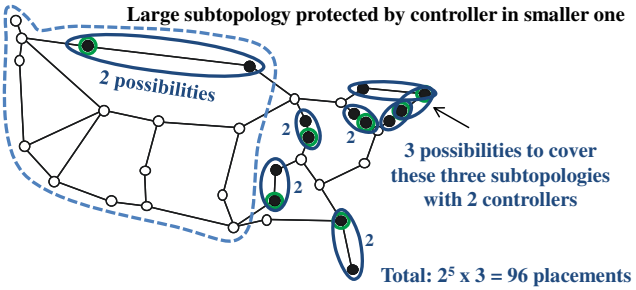
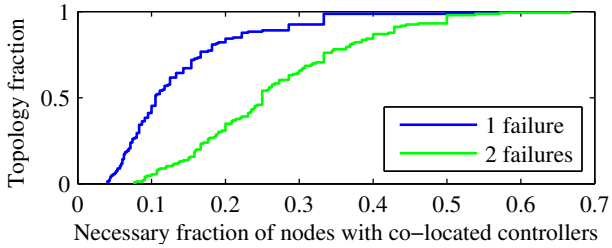


Fig. 9. Subtopologies needing a controller to eliminate controller-less nodes.

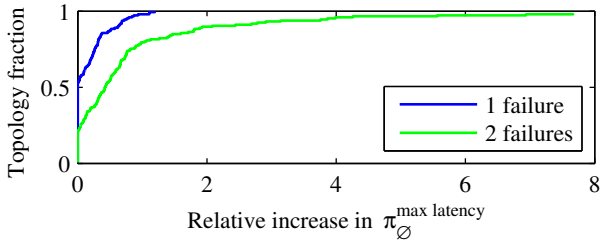
In this topology with $k = 7$, there are only 96 out of $\binom{34}{7} = 5.4$ million placements that are resilient, i.e. covering all subtopologies. This shows that the protection against network disruption significantly reduces the fraction of possible controller placements, in this case to 0.002%. The nodes in Figure 9 with a green border illustrate the best placement out of the 96 placements according to $\pi_{\emptyset}^{\text{max latency}}$. However, with $\pi_{\emptyset}^{\text{max latency}} = 44.9\%$, this placement is by far worse than the best possible placement with an equal number of nodes not fulfilling the resilience criterion ($\pi_{\emptyset}^{\text{max latency}} = 22.5\%$). That means that depending on how important $\pi_{\emptyset}^{\text{max latency}}$ is for an operator, much better results of this value could be obtained by paying the price of being not or not entirely resilient against controller-less nodes.

Before concluding this section, we extend the investigations made for the Internet2 OS3E topology to the Topology Zoo. In Figure 10, we illustrate the minimum number of controllers (measured in percentage of nodes that require a controller) to eliminate controller-less nodes and the increase in $\pi^{\text{max latency}}$ to be resilient for all 146 Topology Zoo topologies.

Obviously, the necessary fraction of nodes with co-located controllers is higher when resilience against two failures is required, compared to the one-failure-resilience. In 85% (125 out of 146) of the topologies it is enough to place $k = 2$ controllers to prevent controller-less subtopologies in any single node failure scenario. These are evidently all topologies that can not be split into subtopologies (larger than a single node) by a single node failure, i.e. at least two-connected (excluding potential single "appendix nodes") with a vertex connectivity larger than one. Depending on the network size, $k = 2$ results in a different fraction of nodes of a topology. When two arbitrary node failures should be covered, 50 out of 146 topologies need only $k = 3$ controllers, i.e. they are at least three-connected (excluding potential single "appendix nodes") with a vertex connectivity larger than two . All other topologies can be split into several subgraphs with two failures and need more controllers. The topologies that require a fraction of around 50% to be covered have a ring structure. To protect all subtopologies in a ring, every other node has to host a controller, leading to 50% fraction for an even node count or even more than 50% for an odd node count.



(a) Minimal necessary fraction of nodes with co-located controllers to fulfill the resilience property.



(b) Relative increase in $\pi_{\emptyset}^{\text{max latency}}$ to be resilient.

Fig. 10. Evaluation of resilience against network disruption and controller-less nodes for different numbers of node failures and different topologies.

Finally, we look at the price in terms of increased latencies that has to be paid to achieve resilience against controller-less nodes. In Figure 10(b), we show the relative increase in the optimal $\pi_{\emptyset}^{\text{max latency}}$ between the best resilient placements with minimal controller number k and the best placements with an equal number of controllers k that are not fulfilling the resilience criterion. The topologies where $\pi_{\emptyset}^{\text{max latency}}$ does not increase, i.e. is equally good as without resilience, mostly correspond to topologies, where the number k of necessary controllers is low. In this case, the fraction of resilient placements is high and in particular, often contains the best possible placements with regard to $\pi_{\emptyset}^{\text{max latency}}$. There are however some topologies where the best values obtainable with a resilient placement are far worse than those obtainable without regarding the resilience requirements. An option to obtain both, a resilient placements and competitive $\pi^{\text{max latency}}$ values, could be, to place more than the minimum number of controllers. When first, a resilient placement is accomplished, the remaining controllers can be used subsequently to decrease the maximum latency. However, this requires additional controllers and thereby increases the complexity.

IV. Further Aspects of Resilient Placements

In this section, we discuss further aspects of resilient controller placements. First, we focus on node-to-controller load balancing. Then, we address inter-controller latency in the placement process.

A. Balancing Controller Load

Depending on the use case, it can be desirable to have roughly equal load on all controllers, so that no controller is overloaded while others have only little work to do. In the following, we address a good balance of the node-to-controller distribution. As formal metric, we introduce the balance of a placement or rather the imbalance, $\pi^{\text{imbalance}}$, i.e. the offset to a totally balanced distribution, as the difference between the number of nodes assigned to the controller with the most nodes and the number of nodes assigned to the controller with the fewest nodes.

As mentioned before, it is assumed that each node is assigned to its closest controller according to the distance matrix $d_{v,w}$. These assignments allow to define assignment matrixes n_p^s containing for each failure scenario s and controller p the number of nodes assigned to this controller. $\pi_{\emptyset}^{\text{imbalance}}$ and $\pi_{\mathcal{X}}^{\text{imbalance}}$ are defined subsequently as follows:

$$\pi_{\emptyset}^{\text{imbalance}}(\mathcal{P}) = \max_{(p\in\mathcal{P})} n_p^{\emptyset} - \min_{(p\in\mathcal{P})} n_p^{\emptyset} \tag{7}$$

$$\pi_{\mathcal{X}}^{\text{imbalance}}(\mathcal{P}) = \max_{(s\in\mathcal{X})} \left(\max_{(p\in\mathcal{P})} n_p^s - \min_{(p\in\mathcal{P})} n_p^s \right) \tag{8}$$

As shown in Figure 1(c), the node-to-controller distribution can be really imbalanced if not considered while choosing the controller placement. If taken into account, the imbalance can be drastically reduced. There are actually many placements with $k = 5$, leading to each controller having either 5 or 6 nodes assigned and thus $\pi_{\emptyset}^{\text{imbalance}} = 1$. However, this leads to an increase of the corresponding $\pi^{\text{max latency}}$ values. Figure 11 illustrates the trade-off between the metrics $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$ by displaying the entire solution space. Analogously to the previous illustration, the figure also shows the mean values, as well as the Pareto-optimal values. For better visibility, the axes limits of the graph have been adapted to display only the most important range of the placement solution space. To give an impression, the worst placements have values of $\pi_{\emptyset}^{\text{max latency}} > 85\%$ of the diameter and $\pi_{\emptyset}^{\text{imbalance}} \geq 25$.

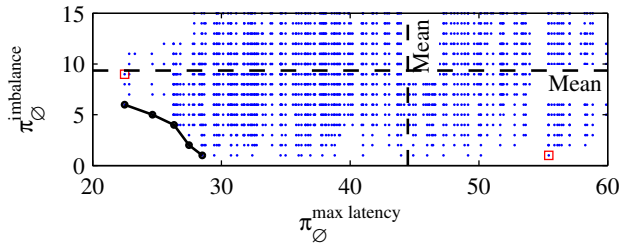
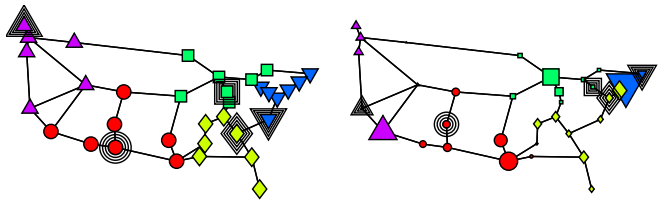


Fig. 11. Trade-off between $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$.

The graph shows that there exist really well-balanced placements with optimal balance: $\underline{\pi}_{\emptyset}^{\text{imbalance}}$ that can have extremely bad $\pi_{\emptyset}^{\text{max latency}}$. The worst of these placements is illustrated by a red square in Figure 11. In this case, $\pi_{\emptyset}^{\text{max latency}}$ is even worse than the mean value of all possible placements. Similarly, placements with lowest latency: $\underline{\pi}_{\emptyset}^{\text{max latency}}$ can have bad $\pi_{\emptyset}^{\text{imbalance}}$ values. The worst value is again marked with a red square. However, a good trade-off between both metrics is possible. The set of all Pareto-optimal values returned by POCO allows the network operator to choose one of the placements which seems to be the most adequate for their particular needs, e.g. a rather balanced one or one with lower latencies.

To verify whether our observation also holds for other topologies, we evaluated the trade-off between the optimal values for both parameters $\underline{\pi}_{\emptyset}^{\text{max latency}}$ and $\underline{\pi}_{\emptyset}^{\text{imbalance}}$ for the topologies in the Topology Zoo. In about 20% of all topologies, there is one placement which is best according to both metrics. In the other 80% of topologies, the choice of the best placement according to one metric can significantly worsen the other. In these cases, our POCO-framework helps the operator to choose the most adequate and Pareto-optimal controller placement according to the network's policies.

In the context of controller-less nodes in Subsection III-B we have already used node weights based on city populations to illustrate different importance of different nodes. For load balancing, and especially for some NFV use cases like firewalling or monitoring, different nodes impose different load on the controllers. To illustrate this effect, Figure 12 depicts the best placements $\mathcal{P}_{\emptyset}^{\text{imbalance}}$ for two different node weights: uniform and based on city populations. This consideration can also be extended from static weights to dynamic weights. As an example, we considered four topologies together with dynamic traffic matrices offered in the SNDlib library [10]. Due to space constraints no further details can be shown here, but we revealed that node weight dynamics can be quite effectively included in the controller placement process by considering a set of dynamic matrices as reference.



(a) Uniform node weights. (b) City populations as node weights.

Fig. 12. Placements $\mathcal{P}_{\emptyset}^{\text{imbalance}}$ for different node weights.

To complete the discussion on load imbalance, we describe how to combine both, the resilience requirements from this Subsection and Subsection III-A. In other words, we look for a placement that offers a trade-off between load balancing and low maximum latency - not only in the failure free case but also in case of controller failures. Altogether, this leads to four optimization objectives to be considered at the same time: $\pi_{\emptyset}^{\text{max latency}}$, $\pi_{\mathcal{C}}^{\text{max latency}}$, $\pi_{\emptyset}^{\text{imbalance}}$, and $\pi_{\mathcal{C}}^{\text{imbalance}}$.

An intuitive approach to address this multi-criteria optimization would be introducing constraints to some of the metrics, e.g. $\pi_{\mathcal{C}}^{\text{max latency}} < 80\%$ and $\pi_{\mathcal{C}}^{\text{imbalance}} < 15$, and reducing the set of all placements to a smaller set of candidates fulfilling these constraints. Then, in a second step, the best placement out of this candidate set is chosen as adequate placement according to the remaining metrics $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$. However, this procedure can be unfavorable for two reasons. First, it demands a lot of knowledge about the network to choose limits for the constraints. Second, the risk is high to miss a much better result with respect to some metrics, if the result only slightly extends the limits given by the constraints.

Therefore, we use another approach. We compute the set of all Pareto-optimal values according to the four considered criteria and show them in Figure 13.

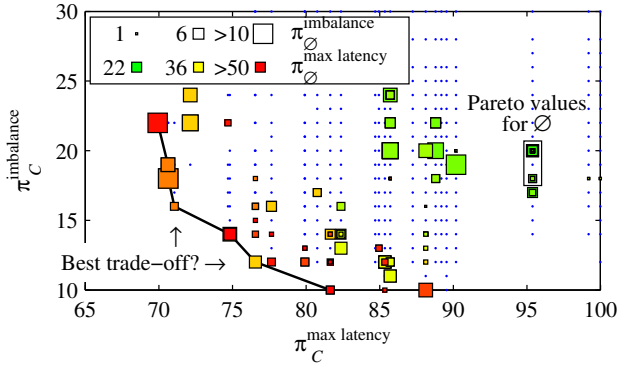


Fig. 13. Trade-off between $\pi_{\emptyset}^{\text{max latency}}$, $\pi_{\mathcal{C}}^{\text{max latency}}$, $\pi_{\emptyset}^{\text{imbalance}}$, and $\pi_{\mathcal{C}}^{\text{imbalance}}$ displayed as 4-dimensional Pareto-optimal values.

To display the 4-dimensional solution space, the following illustration is chosen. Two out of the four dimensions, $\pi_{\mathcal{C}}^{\text{max latency}}$ and $\pi_{\mathcal{C}}^{\text{imbalance}}$, are chosen as x- and y-axis of the graph. The other two dimensions, $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$, are illustrated by different marker sizes and colors. For $\pi_{\emptyset}^{\text{max latency}}$, we use the traffic-light color scheme as described before, and for $\pi_{\emptyset}^{\text{imbalance}}$ we use squares of different sizes, where a larger size indicates larger $\pi^{\text{imbalance}}$ values, i.e. worse balance. The points connected by black lines show the two-dimension Pareto-set according to $\pi_{\mathcal{C}}^{\text{max latency}}$ and $\pi_{\mathcal{C}}^{\text{imbalance}}$. The points in the background illustrate all the remaining solution space that is not part of the Pareto-optimal set. The area surrounded by a rectangle on the right side of the graph illustrates the location of the two-dimensional Pareto set according to $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$. Looking at this location shows that in particular, choosing one of the placements being Pareto-optimal for the failure free case can result in arbitrarily bad performance when considering also the controller failure case. The constraint approach as described above could be illustrated in the figure, by looking only at these Pareto-values which are in the "lower left corner" of the graph and selecting the best, i.e. "greenest" or "smallest" square, out of those. Maybe the best trade-off, as indicated in the figure, is reached, if the "smallest" or "lightest" square on the Pareto-optimal curve according to $\pi_{\mathcal{C}}^{\text{max latency}}$ and $\pi_{\mathcal{C}}^{\text{imbalance}}$ is chosen. In this case, the resilience is included in the trade-off as well as possible, and still the failure free case is on an acceptable level. All this information provided by our POCO-framework allows the network operators to choose the most adequate placement for their particular requirements. This can be either one of those described before or a totally different one.

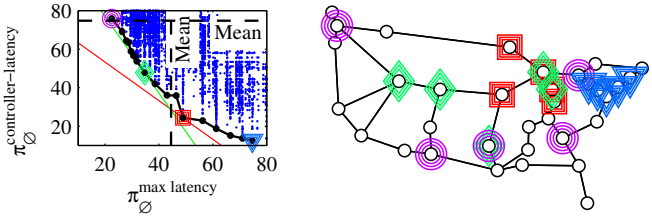
B. Inter-Controller Latency

As last aspect of resilient controller placement, we investigate how the inter-controller latency can be respected in the choice of a controller placement and what influence it has on the normal controller-to-node latency. Formally, the inter-controller latency $\pi^{\text{controller-latency}}$ is defined as the largest latency between any two controllers p_1, p_2 of a placement $\mathcal{P}$:

$$\pi^{\text{controller-latency}}(\mathcal{P}) = \max_{(p_1,p_2\in\mathcal{P})} d_{p_1,p_2} \tag{9}$$

In [11], Levin et al. investigate the impact of a "logically centralized" but physically distributed network control plane on the operational performance of a network. However, they do not consider placement issues or the impact of failures on such an approach.

In general, without illustrating the placements here, all placements considering inter-controller latency tend to place all controllers much closer together. This increases the maximum latency from nodes to controllers.



(a) Solution space. (b) Example placements.

Fig. 14. Trade-off between node-to-controller and inter-controller latencies.

In Figure 14(a), we show the trade-off between both latency metrics for the failure free case $\emptyset$. The Pareto-optimal values show that these two optimization goals cannot be achieved at the same time. To illustrate how the Pareto-optimal placements look like, Figure 14(b) shows four example placements out of the Pareto-optimal set: the best according to inter-controller latency, the best according to node-to-controller latency, the best when taking the average of inter-controller and node-to-controller latency, as well as the best when taking the weighted average with node-to-controller latency counting twice as much as inter-controller latency. It can be observed that when higher influence is given to node-to-controller latency, the controllers are more distributed in the network. Otherwise, when inter-controller-latency is given higher priority, the controllers are placed closer together.

V. Related Work

Before concluding this paper, we provide an overview on related work on the general controller placement problem and its variants. Related work to resilience in general can be found in our previous publications, e.g., [12], [13].

The general problem without any additional constraints is also known as *plant, warehouse, or facility location problem*. If the objective is to minimize $\pi^{\text{max latency}}$, the problem is called *k-centers problem*, if the objective is $\pi^{\text{avg latency}}$, it is called *k-median* or *k-mean problem*. Further references to this general problem are provided in Heller's work [4]. A variant of the problem similar to our node-to-controller balancing has been introduced by Archer et al. [14] as *load-balanced facility problem*. The objective is similar to our $\pi^{\text{imbalance}}$. However, the authors address this problem in another context and provide only approximations to the solution. In the context of load balancing, also the term *capacitated and uncapacitated facility problem* can be found, see e.g. [15] and contained references. The *capacitated* version assumes that the maximum number of nodes that can be assigned to a single controller is limited.

Different authors, among others Khuller et al. [16] and Chaudhuri et al. [17], look at variants called *fault tolerant* or *p-neighbor k-center problems*. These variants are similar to

what we call "controller failure resilient placements". Again, only approximations to the problem are provided. Zhang et al. [6] address a resilient optimization of the controller placement problem considering the outage of controllers or connections between nodes and controllers, i.e. network disruptions. They do not reassign nodes to new controllers if the original controller fails, but assume these nodes to be controller-less. They propose a placement heuristic and simulation with the objective to minimize the number of such controller-less nodes. The very recent work of Hu et al. [18] goes in a similar direction. It introduces and compares different heuristic approaches to increase the resilience of a software defined network against connection failures between nodes and controllers. Both papers [6], [18] focus only on resilience against network failures and do not consider any additional metrics such as $\pi^{\text{imbalance}}$ or $\pi^{\text{max latency}}$ or trade-offs between resilience and failure free case.

VI. Conclusion and Outlook

When designing a centralized network control architecture, it is crucial to determine how many controllers are required and where they should be located in the network. In this paper, we addressed these questions including different important aspects: quality in terms of maximum latencies between nodes and controllers as well as resilience in terms of failure tolerance and load balancing. We discussed different resilience aspects and showed that the optimal values for the metrics quality and resilience are often impossible to achieve at the same time and adequate trade-offs have to be found. In particular, we revealed that in most of the topologies more than 20% of all nodes need to be controllers to assure a continous connection of all nodes to one of the controllers in any arbitrary double link or node failure scenario. The existence of more than a single controller, however raises new questions such as the inter-controller latency. We have also briefly addressed this issue in our work. Introducing our framework for resilient Pareto-based Optimal COntroller-placement (POCO), we offer network operators a range of options to select the placement that is most adequate for their particular needs, e.g., certain constraints on maximum latency, failure tolerance, or maximum number of nodes per controller. An evaluation on a large set of 146 topologies from the Topology Zoo underlines the validity of our findings. The POCO-toolset used to produce the results presented in this paper is available online *[http://www3.informatik.uni-wuerzburg.de/poco]*.

In the future, we plan to extend our work in two directions. Based on our Omnet++ implementation [19] of OpenFlow, we want to evaluate further aspects of resilient controller placement on a level of detail that could not be covered by this paper. This is especially interesting for packet-based protocol simulations considering, e.g., restoration times for a node to switch between different controllers or a detailed trade off between node-to-controller and inter-controller latency. In addition, we will look at controller architectures and placement metrics for other SDN scenarios, such as SDN in business networks, data centers, or access networks, and integrate them into POCO.

Acknowledgments

The authors want to thank Rastin Pries for the fruitful discussions and helpful insights.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM CCR*, vol. 38, no. 2, 2008.

[2] A. Tootoonchian and Y. Ganjali, "HyperFlow: a Distributed Control Plane for OpenFlow," in *INM/WREN'10*, Berkeley, CA, USA, 2010.

[3] Network Functions Virtualisation - Introductory White Paper. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *HotSDN '12*, New York, NY, USA, 2012.

[5] *CPLEX*, ILOG, Inc., http://www.cplex.com/.

[6] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in *GLOBECOM 2011*, 2011.

[7] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in *European Workshop on Software Defined Networks (EWSDN)*, Darmstadt, Germany, Oct. 2012.

[8] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE JSAC*, vol. 29, no. 9, 2011.

[9] Wolfram Alpha. [Online]. Available: http://www.wolframalpha.com

[10] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0 - Survivable Network Design Library," *Networks*, vol. 55, no. 3, 2009.

[11] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized? State Distribution Trade-offs in Software Defined Networks," in *HotSDN '12*, 2012, pp. 1–6.

[12] M. Menth, M. Duelli, R. Martin, and J. Milbrandt, "Resilience Analysis for Packet-Switched Communication Networks," *IEEE/ACM Transactions on Networking*, vol. 17, Dec. 2009.

[13] M. Menth, R. Martin, and J. Charzinski, "Capacity Overprovisioning for Networks with Resilience Requirements," *SIGCOMM CCR*, vol. 36(4), Oct. 2006.

[14] A. Archer and S. Krishnan, "Importance Sampling via Load-Balanced Facility Location," in *IPCO'08*, Bertinoro, Italy, 2008.

[15] F. J. F. Silva and D. S. de la Figuera, "A Capacitated Facility Location Problem with Constrained Backlogging Probabilities," *IJPR*, vol. 45, no. 21, 2007.

[16] S. Khuller, R. Pless, and Y. Sussmann, "Fault Tolerant K-center Problems," *Theoretical Computer Science*, vol. 1203, 1997.

[17] S. Chaudhuri, N. Garg, and R. Ravi, "The p-Neighbor k-Center Problem," *IPL*, vol. 65, no. 3, 1998.

[18] Y. nan Hu, W. dong Wang, X. yang Gong, X. rong Que, and S. duan Cheng, "On the Placement of Controllers in Software-Defined Networks," *JCUPT*, vol. 19 Supplement 2, 2012.

[19] D. Klein and M. Jarschel, "An OpenFlow Extension for the OMNeT++ INET Framework," in *6th International Workshop on OMNeT++*, Cannes, France, Mar. 2013.

Appendix

Table I summarizes the symbols used in this work.

TABLE I. Summary of abbreviations and symbols.

Name	Description
$\emptyset$	Failure free case
$\mathcal{C}$	Controller failure case - if not specified differently, we consider the simultaneous outage of up to all except for one controller.
$\mathcal{N}$	Node failure case - if not specified differently, we consider the simultaneous outage of up to two nodes.
$\mathcal{X}$	Considered failure case. Shown results focus on $\mathcal{X} \in \{\emptyset, \mathcal{C}, \mathcal{N}\}$.
$\pi_{\mathcal{X}}^{\text{max latency}}$	Maximum latency. Corresponds to metric L_{wc} in [4].
$\pi_{\mathcal{X}}^{\text{avg latency}}$	Average latency. Corresponds to metric L_{avg} in [4].
$\pi_{\mathcal{X}}^{\text{imbalance}}$	Node-to-controller distribution imbalance.
$\pi_{\mathcal{X}}^{\text{controller-less}}$	Maximum number of controller-less switches ($\mathcal{X} \in \{\mathcal{L}, \mathcal{N}\}$)
$\pi_{\mathcal{X}}^{\text{controller-latency}}$	Maximum inter-controller latency.
$\underline{\pi}_{\mathcal{X}}^{\text{metric}}$	Best value π^{metric} for given metric and controller number k.
$\mathcal{P}_{\mathcal{X}}^{\text{metric}}$	Any placement leading to $\underline{\pi}_{\mathcal{X}}^{\text{metric}}$.