A Resilient Distributed Controller for Software Defined Networking

Diogo M. F. Mattos*†, Otto Carlos M. B. Duarte*, and Guy Pujolle†

*Grupo de Teleinformática e Automação - Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Rio de Janeiro, Brazil - Email:{diogo,otto}@gta.ufrj.br

†Laboratoire d'Informatique de Paris 6 - Sorbonne Universities, UPMC Univ Paris 06
Paris, France - Email: {Diogo.Mattos,Guy.Pujolle}@lip6.fr

***Abstract*—Control plane distribution on Software Defined Networking enhances security, performance and scalability of the network. In this paper, we propose an efficient architecture for distribution of controllers. The main contributions of the proposed architecture are: i) A controller distributed areas to ensure security, performance and scalability of the network; ii) A single database maintained by a designated controller to provide consistency to the control plane; iii) An optimized heuristic for locating controllers to reduce latency in the control plane; iv) A resilient mechanism of choosing the designated controller to ensure the proper functioning of the network, even when there are failures. A prototype of the proposal was implemented and the placement heuristic was analyzed in real topologies. The results show that connectivity is maintained even in failure scenarios. Finally, we show that the placement optimization reduces the average latency of controllers. Our proposed heuristic achieves a fair distribution of controllers and outperforms the network resilience of other heuristics up to two times better.**

I. Introduction

Software Defined Networking (SDN) advocates a logically centralized network control plane [1]. The control plane centralization and the consistent global network view allow operators to innovate and agilely implement new services in the network core. However, the physical centralization of control plane in Software Defined Networks implies challenges to security, performance and scalability of the network [1]. Scalability is affected by both the network diameter; as distant switches suffer from great latency to access controller; and also the number of switches in the network, since the number of controllers may not be sufficient to respond in a timely manner to requests from all switches [2].

The control distribution and controller replication are the main techniques to increase the resilience of Software Defined Networks [3]. The adoption of a controller model that acts as a distributed system [4], [5], or the simple replication of controllers are not enough to assure the security, the performance and the scalability of the network. It still depends on the optimized mapping between switches and controllers, which creates control domains. Another emerging challenge is maintaining consistency of the unified global network view [6], [7]. Inconsistent global network view brings performance loss and execution errors to the network-control applications [7]. Thus, SDN control distribution can be understood from two standpoints. The first one deals with the provision of a distributed controller with a consistent global network view [5], [4]. The second optimizes the location and the number of required controllers over a topology [8], [9].

In this paper, we propose an efficient architecture for physical distribution of controllers in a Software Defined Network. Thus, we discuss the control distribution from the perspective of both approaches. We propose and develop a distributed controller for SDN which creates independent areas. Each controller takes over an area. The controllers of the areas report to a designated controller, who is responsible for maintaining the consistency of the global view known by each controller. Then, we propose optimization heuristics for the controller placement problem. Our heuristics consider the minimization of latency between the controller and switches, and also the maximization the connectivity of controllers and the network, in failure scenarios. A prototype of the proposed controller was implemented. The controller placement optimization was analyzed against real network topologies.

The control distribution between different physical nodes, while maintaining a centralized global view of the network, is a major challenge in SDN [5], [4]. Earlier proposals point out drawbacks of Software Defined Networks, and define initial solutions for the control distribution and for enhancing resilience of the network [8], [10]. Other proposals seek to create new distributed SDN controller implementations, from the point of view of a distributed operating system. Nevertheless, these proposals do not optimize the controller placement [11], [5], [4], [12]. There are also studies that focus on optimizing the placement of the controllers according to different heuristics, and objectives [3], [9]. These proposals do not aim to create a single architecture that considers the problem from the controller design up to the optimization of the controller placement. In our turn, in this paper, we propose a distributed control architecture with optimized controller placement, and assurances of network resilience.

The remainder of the paper is organized as follows. Related works are discussed in Section II. Section III presents the problem of distribution and placement of controllers. The distributed controller and the placement heuristics are proposed in Section IV. Section V evaluates our proposed architecture. Section VI concludes the paper.

II. Related Works

Levin *et al.* argue that the distribution of physical control, while the logic control remains centralized, affects network performance, when the centralized control applications are

This work was sponsored by CNPq, CAPES, and FAPERJ.

agnostic about the state of the distribution [6]. The authors identify the existence of two trade-offs. The first one is between the performance of applications and overhead of distributing control states between controllers. The second trade-off is between the complexity of the logic of control applications and the robustness against inconsistency. In their turn, Schmid and Suomela argue that there are applications that can be performed by local controllers without requiring a consolidated global view [7].

ONIX controller acts as a middleware for distributed operating systems, spreading the information to the physical controllers, and provides an API (Application Programming Interface) for applications to communicate between the different replicas [5]. ONOS controller is proposed based on the same precepts of ONIX [4]. ONOS controller is based on the Floodlight controller and distributes the network states according to the distributed coordinator Zookeeper. These proposals implement the eventually-consistent model, which enhances controller availability and performance at the cost of inconsistencies on the global network view [13]. Another distributed controller is the HyperFlow [11], in which events are disseminated through a Publisher/Subscriber channel.

The Kandoo proposal is a hierarchical SDN controller, where the scope of applications is divided into local or global [12]. Local and global applications coexist in the network control. The local applications are those that operate only on the local state of each controller and, therefore, are implanted in controllers next to the data plane. In their turn, the global applications are on top-hierarchy controllers.

Müller *et al.* propose a controller placement strategy, called *Survivor*, which considers the diversity of paths from switches to the controller, the resources of controllers and disaster-recovery mechanisms [3]. Bari *et al.* also propose an optimization scheme of controller placement according to the network load [9]. Moreover, Ros and Ruiz propose the Fault Tolerant Controller Placement (FTCP) problem formulation [13]. Ros and Ruiz claim that a reliable SDN installs controllers according to a network reliability-estimator constrain. Unlikely these previous works, which do not handle the controller placement problem for both approaches of physically distributing the control, and optimizing the controller placement, we propose a distributed control architecture that considers controller resilience while performing the controller placement.

III. The Controller Distribution and Placement

Heller *et al.* defines controller distribution and placement problem defines as two SDN control features: i) the number of required controllers; and ii) the location and placement of these controllers [8]. However, these two features do not exhaust the network control optimization possibilities. These two features just guide the search for optimized control solutions. Moreover, we consider an objective function, which may maximize the controller distribution to tolerate failures, whereas it can also minimize the latency to increase performance and to reduce the flow set up time.

Besides the optimization of the placement and the best number of controllers to fit the network needs, there is also the challenge of distributing the consistent global network view among the controllers. For this end, we consider the network

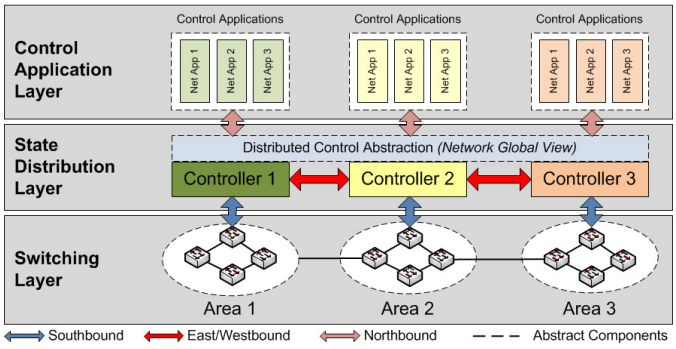


Figure 1. A Software Defined Network with its distributed control plane. The network model is composed of three layers: switching, state distribution, and application layers. The arrows show the control dataflow. Each point of data exchange has its own API (Southbound, Northbound, and East/Westbound).

architecture model shown in Figure 1. Figure 1 shows the points of network-states data exchange on a SDN. The network is classified into three logical layers [7]. The bottom layer, the switching layer, is composed of packet-forwarding elements. These elements store the forwarding tables, and in the case of OpenFlow [1], they also store information regarding flows and flow counters.

The middle layer, the network-state distribution layer, is the communication layer between controllers, propagating the event and local states of each controller. In this paper, we propose the realization of state distribution layer through a designated controller who is responsible for maintaining an updated copy of the global network view. The state distribution layer is typically referred to as Network Operating System [6], as it mediates communication between control applications and the switching layer, abstracting the switching layer for the above applications. Finally, the top logical layer, control application layer, hosts control applications. Each application exchanges information with others via the state distribution layer. Applications can run on multiple physical nodes.

Figure 1 highlights three sensitive points of data exchange. The first one is between switches and controllers (Network Operating Systems). The most common implementation of SDN standardizes this interface through the OpenFlow API [1]. This interface is called Southbound API. The second is among controllers in the state distribution layer. In this case, it is horizontal communication interface between controllers that is called East/Westbound API. Finally, we highlight the data exchanging between the controllers and the applications, which is called Northbound API [10].

IV. The Proposed Control Architecture

The key idea of the proposed controller is to divide the Software Defined Networking control plane into areas, and to abstract the single global network view for all area controllers. Therefore, our proposal envisages the creation and the maintenance of a consistency layer among controllers. This layer distributes the global network view among all area controllers.

Figure 2 shows the architecture of the proposed controller. One of the area controllers is chosen as the Designated Controller. The designated controller is a controller like the others, but, since it is chosen, it becomes responsible for

maintaining the global network view and for propagating it to others. The global network view is any information of the control applications which are of interest over an area controller and the area controller provides to others.

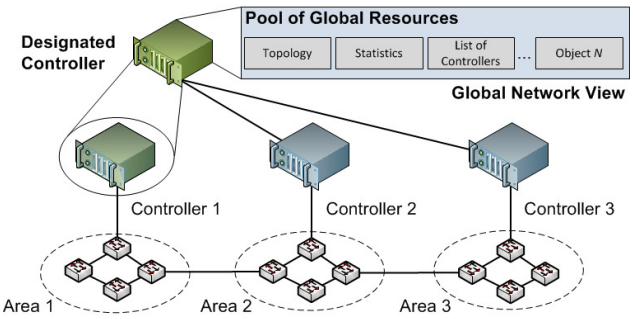


Figure 2. The proposed distributed controller. The Controller 1 is the Designated Controller and, thus, it keeps the Pool of Global Resources. The pool is accessible to all controllers and distributes of the global network view.

The pool of global resources is important to achieve the abstraction of distributed control. To access an object that is registered in the designated controller, the area controller can operate it as if it was local. The control of the object's consistency is performed by the designated controller. Thus, each registered object relies on a lock and can only be updated through a single access by a time. The applications running in area controllers can perform local control operations without the need for reporting to the designated controller, as local operations do not update the global network view.

For applications that update global network view, such as topology discovery application, each area controller accesses the object of the application data and updates it. In the case of topology discovery, the shared object is a collection of the known links between the known nodes of the network. Besides, other global application is the calculation of the spanning tree. It is worth noting that, given the consistent and single global network view, all nodes calculate the spanning tree locally. Each area controller is only responsible for activating and deactivating flood property of switch ports on its own area, without taking action over than its own domain [7].

Our proposal enhances network resilience as follows. Each area controller is able to assume the role of designated controller. Thus, each area controller has a sorted list of which controllers must assume the role of designated controller in case of failure. This sorted list is distributed as an object registered on the pool of global resources in the active designated controller. From time to time, area controllers access the sorted list and keep their local copy of the global state, since, if the designated controller fails, the overall state of the network is maintained. The list is sorted according to the increasing value of $L(C_k)$, the average latency from a controller to all others. Each controller C_k is associated with a value:

$$L(C_k) = \frac{\sum_{i \in N} Latency(C_k, C_i)}{|N|}, \qquad (1)$$

where i is the index of each network node, N is the set of all network nodes, and C_i is the controller of node i. The $Latency(C_k, C_i)$ in Equation 1 is calculated *a priopri*, according to the network topology, at the time of controller placement. The designated controller just keeps this updated value according to network monitoring[1].

The placement of the controllers is defined as an optimization problem with multiple objectives. To achieve the maximum network resiliency is important that the placement of the controllers considers that, in a failure scenario, the network remains connected and that, if there is partition of the network, each partition has a controller inside. At the same time, it is important for the proper operation of the network, that the setup time of a new flow is minimized [8], [9]. Thus, the communication time between the controller and the set of controlled switches is minimized. The first proposal for an objective function of our optimization problem is given by:

$$min \prod_{i \in N} (NetworkRatio(C_i) * Prob(C_i))^{y_i}, \qquad (2)$$

where y_i identifies whereas a node i is a controller ($y_i = 1$), or not ($y_i = 0$). $NetworkRatio(C_i)$ identifies the network ratio that goes out of control if C_i fails, and N represents the set of all nodes in the network.

Moreover, as a second approach, we also define another objective function for maximizing resilience, while minimizing latency. This approach exponentially weights the number of switches per controller, and is given by:

$$min \sum_{i \in N} y_i * (e^{Count(i)} + Prob(i) * \ e^{OutNodes(i)}), \qquad (3)$$

where we introduce two function $Count(i)$ and $OutNodes(i)$. The function $OutNodes(i)$ counts how many nodes are disconnected and out of control in a case of failure of node i. The function $Count(i)$ represents how many switches are controlled by the controller indexed i. This function introduces itself a minimization, which maps a switch to the controller that introduces the lower latency, as a direct application of the following expression:

$$min \sum_{i \in N} \frac{\sum_{k \in N} y_i * y_{i,k} * Latency(C_k, C_i)}{|N|}, \qquad (4)$$

which minimizes latency between controllers and the switches, where $y_{i,k}$ represents that controller indexed i controls a node k, and y_i represents if a node indexed i is a controller, or not. The expressions have $i, k \in N$, where N is the set of all nodes in the network.

Our first proposal, the Expression 2, minimizes the ratio of the network, represented by the function $0 \leq NetworkRatio(C_i) \leq 1$, which is out of control in case of failure of controller C_i. This approach is called Partition Minimization. The function $Prob(C_i)$ models the probability of node failure of C_i. The failure of a node C_i, however, only is detrimental to the control of the network if C_i is a controller. Thus, each term of the product operator is raised to y_i that determines whether the node indexed i is a controller or not. In the case of the node is a controller, the term is considered.

[1]The monitoring interval may vary according to the network infrastructure and topology

Otherwise, the term is raised to 0, resulting in the value 1, that do not interfere with the overall result. Thus, we minimize Expression 2, looking for a solution that maintains the network connected and controlled, even in failure scenarios.

In the second approach, we propose to maximize fairness of the number of switches that is in the area of each controller, and to maximize resilience, Expression 3. This approach is called Fairness Maximization. It is a sum that only considers the terms in which the node is a controller ($y_i = 1$). The cost of selecting a controller on this approach is exponentially weighted by the number of switches that a controller takes over. We also weight the number of nodes that goes out of control with the probability of a controller failure. Gathering Expression 3 and Expression 4, we seek for a solution where the number of switches controlled by each controller is fairly distributed between controllers and the probability of networking partition is reduces according to the probability of controller failures.

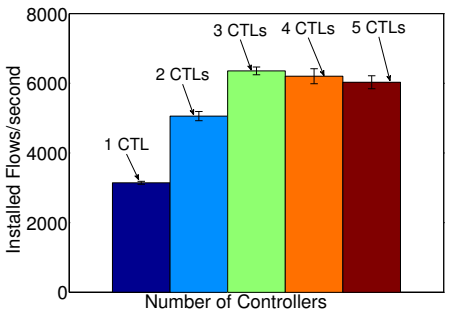
The optimization problem of placing controllers in a SDN to enhance resilience is NP-Hard. The problem formulation recalls to a *fault tolerant facility location* (FTFL) problem [13]. Expression 2 and 3 define the central idea of the proposed placement strategies. Hence, to apply the proposed strategies, we develop a heuristic calculation of the optimum placement of controllers. The heuristic is based on the Simulated Annealing method[2] [14].

The placement problem is modeled as a Simulated Annealing in the following manner. The solution is a Y-length vector, where Y is the number of controllers that we allocate to the network. In each iteration, we generate a new candidate solution that is accepted, if it has a lower cost than the previous. If the solution is not better than the current one, it associated with an acceptance probability. The meta-heuristic gets a random number (between 0 and 1), and checks if the number is less than the acceptance probability of the solution. If so, the solution is accepted, even though the higher cost than the cost of the current solution. This behavior is necessary to prevent that optimization converges to a local minimum.

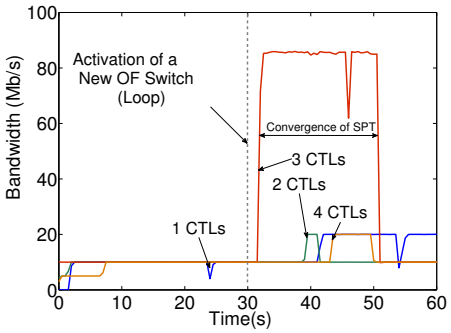
V. The Evaluation of the Proposed Architecture

The proposals were evaluated in two steps. The first step was the implementation of a prototype of the distributed controller. The second step is to evaluate the proposed placement heuristics against real topologies. We evaluate for average latency between controllers and switches, and for network resilience. The proposed optimization heuristics are compared to heuristics with different objective functions: lowest average latency between controller and switches [8], network centroids [15], fewest hops between controller and switches, a greedy algorithm for finding greatest centrality nodes [2].

We consider an OpenFlow network and we implement the distributed controller over the POX controller[3]. The Pool of



(a) Running of `cbench` for 1, 2, 3, 4, and 5 controllers (CTLs on figure). For 4 and 5 controllers, the ration of installed flows/s is lower than for 3 controllers, because of CPU limitations on the testing machine.



(b) A new switch is activated on 30 s, looping the network. While the spanning tree algorithm converges, packets are duplicated on the network.

Figure 3. Evaluation of distributed controller prototype. a) Evaluation of the aggregate number of installed flows on the network. b) Reaction of the distributed controllers to the entry of a new switch in the network.

Global Resources was implemented using an adapted version of DOPY[4], a Python module for distributed programming. It is worth noting that to perform the experiments, the POX applications `forwarding.l2_learning` and `openflow.spanning_tree` were adapted for registering and querying the pool of global resources rather than using the local view of each controller. We run the distributed controller as several processes on an Intel i7-2600 @ 3.40 GHz, 16 GB RAM, running Debian Linux. The OpenFlow switches are four computers equipped with a Core 2 Duo @ 2.40 GHz, 3 GB RAM, with Debian and Open vSwitch[5].

The first experiment assesses the rate of installed flows/second, using to `cbench`[6]. `Cbench` emulates several switches connected to a controller and generates `packet_in` events to measure the controller ability to react to them. Figure 3(a) shows that, by adding more controllers at the control plane, the rate of flows/second increases. With three controllers, the served rate almost doubles in comparison to the configuration with only one controller. However, with 4 or 5 controllers, the rate of served flows is smaller than with three controllers, because there is competition between the controller processes and flow generation process. Thus, there is a processing bottleneck, as 4 out of 8 cores are used by

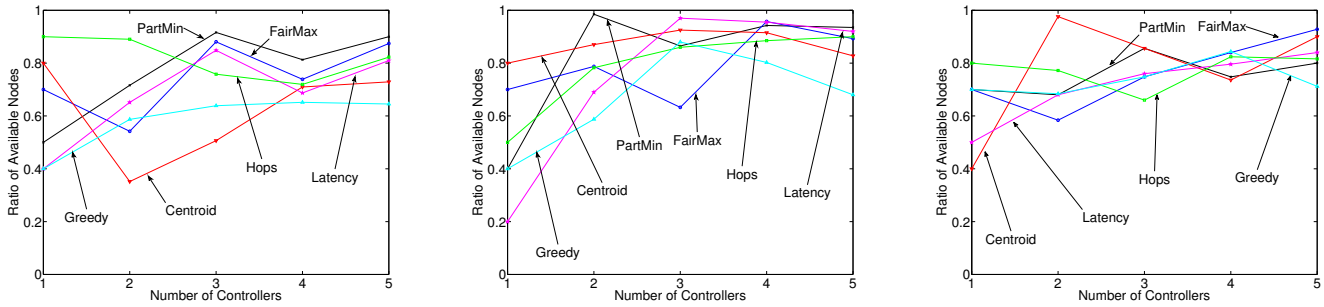
[2]The Simulated Annealing optimization method was chosen as meta-heuristic, because this method is proved to converge to the global minimum on indefinitely time, even in a geometrically decreasing temperature scenario. The disturbance applied on current solution for generating a new solution follows Cauchy distribution.

[3]http://www.noxrepo.org/pox.

[4]http://www.mindhog.net/∼mmuller/projects/dopy/.

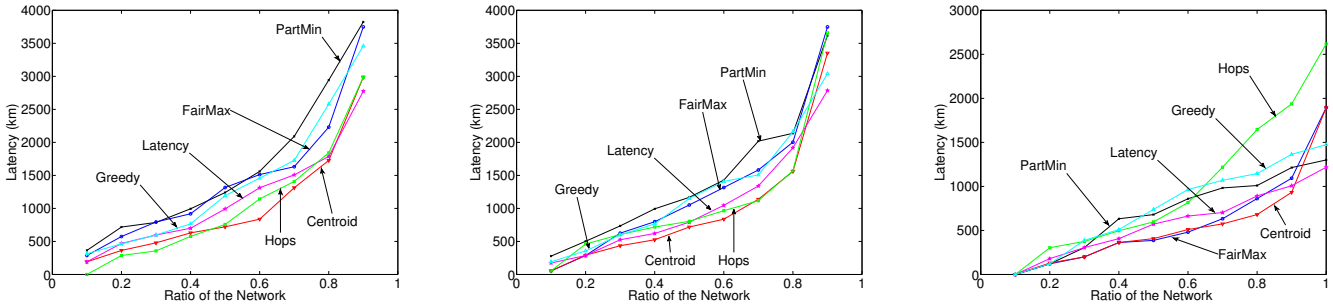
[5]http://openvswitch.org/.

[6]http://www.openflowhub.org/display/floodlightcontroller/Cbench.



(a) RNP Topology in Brazil. (b) GEANT Topology in EU. (c) AT&T Topology in USA.

Figure 4. Median of the network ratio that remains connected and controllable as a function of the number of controllers. We considered a failure probability of 0.5 for network nodes. Comparison between Partition Minimization (`PartMin`), Fairness Maximization (`FairMax`), lowest latency (`Latency`), network centroids (`Centroid`), lowest hop count (`Hops`), and the centrality-based greedy algorithm (`Greedy`).



(a) RNP Topology in Brazil with 3 controllers. (b) GEANT Topology in EU with 4 controllers. (c) AT&T Topology in USA with 5 controllers.

Figure 5. Latency introduced in the network by each placement heuristics. We consider the latency as proportional to the propagation delay. Thus, results are presented in function of the average distance between nodes and controller.

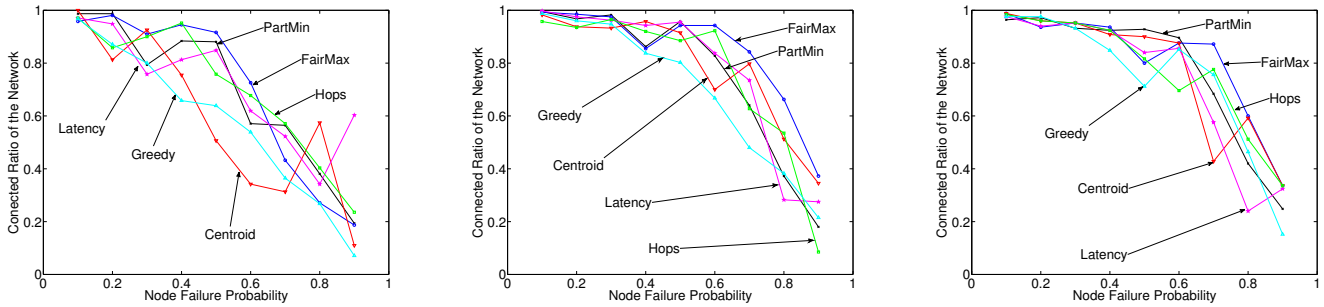
controllers while the other 4 cores are generating flows. The second experiment checks the control plane reaction to the activation of a new switch, creating a loop topology with 4 switches. The switch is activated at 30 s. we observe that after the activation; there is a duplication of packets due to inconsistency in the spanning tree. In the worst case, with three controllers, the convergence of the spanning tree reaches 20 s, and converges to all switches have access to the global network view.

The second step of the proposal evaluation asserts the optimization heuristics. Our experiments evaluate the placement heuristics against three real network topologies[7]: the RNP (Nation Research Network) in Brazil, which counts with 30 nodes; the GEANT network in Europe with 40 nodes; and the AT&T MPLS network in the U.S. with 25 nodes. Figure 4 shows the resilience achieved in each network topology with a variable number of controllers. This experiment runs random nodes failures on the network and verifies the ration of the network that remains connected and controllable even after failures. We compare our two proposals, Partition Minimization (`PartMin`), and Fairness Maximization (`FairMax`), against four other proposals, lowest latency (`Latency`), network centroids (`Centroid`), lowest hop count (`Hops`), and the centrality-based greedy algorithm (`Greedy`). Taking into consideration these results, we defined the best number of controllers for each topology as the number of controllers that provides the greatest resilience. It is noteworthy that the choice of the best number of controllers depends on the network topology, as even increasing the number of controllers, the achieved resilience may not increase. Taking into consideration the Figure 4(b), we conclude that the maximum of resilience for the GEANT network is achieved with 2 controllers. For the RNP network, Figure 4(a), the best number of controllers are 3, and for AT&T, Figure 4(c), 5 controllers.

After deciding the best number of controllers for each topology, we evaluate the latency that each placement heuristic introduces on the controller placement, Figure 5. We verify that the Partition Minimization heuristic (`PartMin`) is the one that introduces the highest latency between controllers and switches. It comes as trade-off of avoiding partitions on the network through the choice of less centralized nodes as controllers. On the other hand, the lowest latency is introduced by the centroid heuristics (`Centroid`), which choses the most central nodes as the controllers. Nevertheless, centroid heuristics implies partition-prone controller placement as it does not consider the connectivity of clusters to network.

The resilience evaluation, shown in Figure 6, reveals that the Fairness Maximization (`FairMax`) heuristics is the one that assures more resilience to the network, even for scenarios where the node failure probability is up to 0.5. Figure 6(a) reveals that our proposed `FairMax` heuristic achieves the double of connected network ratio than when compared to the `Centroid` heuristic with a node failure probability of 0.6. When considering the GEANT topology, Figure 6(b) shows that the `FairMax` is an upper bound to the resilience

[7]The topology graphs are available at the The Internet Topology Zoo (http://www.topology-zoo.org/).



(a) RNP Topology in Brazil with 3 controllers. (b) GEANT Topology in EU with 4 controllers. (c) AT&T Topology in USA with 5 controllers.

Figure 6. Resilience introduced in the network by each placement heuristics. We consider the network ratio that remains connected and controllable as a function of the node failure probability.

of the network. The same result is achieved on the AT&T network topology, Figure 6(c), where our proposed `FairMax` heuristic outperforms all others. It is worth noting that the proposed `PartMin` heuristic also performed well, following the behavior of `FairMax`, but always keeping a lower ratio of network connected nodes. Moreover, `FairMax` is the only heuristic that considers the load balancing between controllers as a metric of the heuristic. The load balancing is done by the exponential weighted count of switches above a controller as a part of the metric.

VI. Conclusion

In this paper, we proposed a controller architecture for Software Defined Networking, which maintains the global network view, while establishing control areas with distributed control. The global network view is achieved by applying a designated controller, which is an area controller that assumes the role of maintaining consistency of the global network view. The designated controller keeps a repository of global objects that is accessible to all controllers. Besides, we also proposed two heuristics for optimizing the placement of controllers: one based on minimizing the probability of partitioning the network (`PartMin`) in case of node failure, and another one based on fairly distributing switch between controllers (`FairMax`), while also minimizing the area of the network that runs out of control in a case of node failure. A prototype of the proposed controller has been implemented and evaluated. The results showed that the global network view is kept, even in a scenario when the control is physically distributed, and all controllers access and converge in the calculation of a common spanning tree. The optimization heuristics were evaluated and the results showed that `FairMax` heuristic enhances the resilience of the network, especially when the failure rate is up to 0.6, when the proposed heuristics outperforms up to two times the centroid-based heuristic.

References

[1] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Van Reijendam, P. Weissmann, and N. Mckeown, "Maturing of OpenFlow and software-defined networking through deployments," *Comput. Netw.*, vol. 61, pp. 151–175, Mar. 2014.

[2] M. E. A. Lopez and O. C. M. B. Duarte, "Providing elasticity to intrusion detection systems in virtualized software defined networks," in *IEEE ICC 2015 - Communication and Information Systems Security Symposium (ICC'15 (11) CISS)*, London, United Kingdom, Jun. 2015.

[3] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving SDN survivability," in *2014 IEEE Global Communications Conference (GLOBECOM)*, Austin, Texas, USA, Dec. 2014.

[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN'14. Chicago, Illinois, USA: ACM, 2014, pp. 1–6.

[5] 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 *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. Vancouver, BC, Canada: USENIX Association, 2010, pp. 1–6.

[6] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized? State distribution trade-offs in Software Defined Networks," in *Proceedings of the First workshop on Hot topics in software defined networks*, ser. HotSDN'12. Helsinki, Finland: ACM, 2012.

[7] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN'13. Hong Kong, China: ACM, 2013, pp. 121–126.

[8] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN'12. Helsinki, Finland: ACM, 2012, pp. 7–12.

[9] M. Bari, A. Roy, S. Chowdhury, Q. Zhang, M. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *2013 9th International Conference on Network and Service Management (CNSM)*, Oct. 2013, pp. 18–25.

[10] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, p. 63, 2015.

[11] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, ser. INM/WREN'10. San Jose, CA: USENIX Association, 2010.

[12] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN'12. New York, NY, USA: ACM, 2012, pp. 19–24.

[13] F. J. Ros and P. M. Ruiz, "On reliable controller placements in software-defined networks," *Computer Communications*, pp. –, 2015, to be published.

[14] L. P. Cardoso, D. M. F. Mattos, L. H. G. Ferraz, O. C. M. B. Duarte, and G. Pujolle, "An efficient energy-aware mechanism for virtual machine migration," in *Global Information Infrastructure and Networking Symposium 2015 (GIIS'15)*, Oct. 2015, pp. 1–6, to appear.

[15] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *2011 IEEE Global Telecommunications Conference (GLOBECOM)*, Dec. 2011, pp. 1–6.