Proactive Controller Assignment Schemes in SDN For Fast Recovery

Selcan Güner
Dept. of Computer Engineering
Bogazici University
Istanbul 34342, Turkey
selcan.guner@boun.edu.tr

Gürkan Gür
Inst. of Applied Information Technology
Zurich University of Applied Sciences (ZHAW)
Winterthur 8401, Switzerland
gueu@zhaw.ch

Fatih Alagöz
Dept. of Computer Engineering
Bogazici University
Istanbul 34342, Turkey
fatih.alagoz@boun.edu.tr

***Abstract*—A sizeable software defined network with a single controller responsible for all forwarding elements is potentially failure-prone and inadequate for dynamic network loads. To this end, having multiple controllers improves resilience and distributes network control load. However, when there is a disruption in the control plane, a rapid and performant controller-switch assignment is critical, which is a challenging technical question. In this work, we propose a proactive switch assignment approach in case of controller failures using a genetic algorithm based heuristic that considers controller load distribution, reassignment cost and probability of failure. Moreover, we compare the performance of our scheme with random and greedy algorithms. Experiment results show that our proposed PREF-CP framework has better performance in terms of probability of failure and controller load distribution.**

I. Introduction

Conventional computer networks consist of network elements such as switches and routers with many complicated protocols, policies and communication interfaces. Their structure is restrictive for network operators to change the network according to the needs of changing traffic demands and service requirements, which are becoming more taxing with the increasing number of mobile devices and impact of big data flows [1], [2]. Furthermore, the proliferation of advanced wireless systems such as Beyond 5G networks and massive connectivity of IoT impose stringent performance requirements on the wired networks such as backhaul and core networks [3]. The idea of Software Defined Networking (SDN) was proposed to facilitate network evolution while addressing these challenges to realize the Future Internet concept [4]. With SDN, control decisions and network intelligence is moved out of individual network nodes, which transforms the network to simpler forwarding hardware augmented with decision making network controllers.

For various software-defined systems, a single controller might be sufficient since a single controller may meet the service level requirements under specific conditions [5]. However, resilience is an important aspect when designing a network architecture. As the system is performing, failures can occur in both control and forwarding planes. A switch, a controller or links between them may fail. For instance, when a switch is disconnected from its controllers, it can not forward new flows and becomes unresponsive except residual flows in time. A failure may eventually cause loss of data which reduces the reliability of the system. Therefore, it is of great importance to improve resilience of software-defined networks [2]. In a network with a single controller, when that controller fails, the network will be left without a control framework, i.e. become "headless". To overcome this single point of failure problem, control plane is distributed over multiple controllers to increase resilience and simultaneously provide adaptation to dynamic network loads [6]. The distribution of switches to controllers influences multiple aspects of network system such as controller-switch latency, load balancing and network reliability. Therefore, with use of multiple controllers, the dynamic and responsive controller-switch assignment becomes crucial.

In this work, we focus on this problem and investigate the efficient controller-to-switch assignment problem in SDN to overcome effects of failures. We propose a proactive switch assignment scheme against controller failures via genetic algorithm considering controller load distribution, reassignment cost and probability of failure. The main premise is to have a pre-calculated mapping for the network calculated at run-time and applied rapidly when failure incidents happen.

II. Related Work

When calculating controller-switch assignment for multiple controller environment in SDN, multiple parameters such as controller load, probability of failure between network components or reassignment cost are considered [7]. Controller failure recovery mechanisms in the literature, are considering controller reliability of the network, or switch-controller delay, or controller load. For considering controller load, [8] proposed to reassignment algorithm which chooses a controller with highest controller capacity. For considering switch-controller delay, [6] proposed a failure recovery mechanism that, for switches under control by a failed controller, the nearest controller takes over these switches. As studied in [8], [9], controller capacity and controller overload are critical issues on switch assignment planning.

To assure the reliability of the control plane, the most important aspect is keeping switches connected to controllers which are up and running. For making it happen, failures of the controllers must be prevented that they can work

properly [10]. In [11], probability of failure is defined as possibility of communications failure from node a to node b. Another important requirement is related to recovery speed after failures: to minimize disruptions in case of controller failure, quick reactions are needed with failover mechanism ensuring that recover connectivity with remaining components. [6].To prevent long restoration and back up calculation time, [12], [6] and [9] generates a back up map proactively. There are also other papers which calculates reassignment on the fly when a failure occurs [8]. Depending on the back-up calculation time both of the approaches have pros and cons. Proactive approaches may be faster to recover from a failure. On the other hand, calculation on the fly may use live network data while proactive calculations will use the last state of the system when the calculation algorithm is run.

III. Proactive SDN Controller-Switch Assignment

There are different assignment algorithms that can be used in case of controller failure at runtime. For instance, the switches of failed controller(s) can be randomly assigned to other controllers. However, such a strategy does not provide satisfactory performance [7]. Thus, in this work, we develop an assignment scheme using genetic algorithm to increase the reliability of a running system. The proposed scheme calculates a back-up switch assignment map considering each controller might fail and taking load-balancing into account to adapt to new conditions if a failure occurs despite all the efforts. It operates in an online manner and proactively determines assignment maps for different failure cases. It determines a backup controller for each forwarding node according to load of controllers, switch reassignment cost and maximum probability of failure for the shortest paths from a controller to its switches. In the case of controller failure, a switch will be assigned to its proactively calculated backup controller.

Software or hardware malfunction on the machine hosting the controller can cause failures on control plane. Since controllers are the most essential devices in the system, the probability of their failures is typically lower than the others [13]. However, a broken controller can no longer get flow request from its switches, and messages from other controllers. Basically, when a controller fails, its switches must be assigned to other controllers.

A. Assignment Parameters

For an assignment to increase connectivity between network components and improve resilience, it should utilize existing network connectivity among the switches. Moreover, the load distribution should be considered to minimize load-induced catastrophic incidents. The cost of assignment is also another factor since reassignment requires on-the-fly configuration of controller framework as well as switches. Therefore, to develop efficient reassignment algorithms with resilience objectives, probability of failure, controller load distribution and reassignment costs are important elements to be considered.

1) Probability of Connectivity Failure: If the connection between controller and the forwarding planes is severed, some switches will be left without any controller and become unresponsive for new traffic flows. Therefore, network connectivity should be ensured to increase reliability of SDN. To formalize this factor, it is defined as the probability of failure 1 of a path from a switch to its controller(s). To consider this issue for an efficacious switch assignment, our algorithm tries to minimize probability of failure P defined as:

$$1 - \prod_{e_i, v_j \in R_{ab}} (1 - P_{e_i})(1 - P_{v_j}) \tag{1}$$

2) Controller Load Distribution: Optimal load distribution directly affects network performance, thus improving load balancing is important for the network resilience [14]. In case of a controller failure, when the switches are assigned to other controllers, an ignorant assignment may lead some controllers to overload and even cause cascaded failures. Thus, reassignment in case of failures should be done taking prospective controller loads into account. For controller load L_c,

$$ControllerLoad(L_c) = \sum_{i \in n} x_{C,n} R_n \tag{2}$$

a relevant assignment criterion is the load variance defined as:

$$LoadVariance(\sigma^2) = \frac{1}{c} \sum_{i \in c} (\overline{L} - L_c) \tag{3}$$

3) Reassignment Cost: When a controller fails, its controllers will be distributed among other controllers. Reassignment cost is considered when calculating reassignment map to to measure the cost of applying reassignment. It is calculated as in (4), the total number of switches whose controllers are changed during reassignment. Number of reassigned switches will be at least as much as switch number of failed controller. Beside that, when network size increases, it is possible that reassignment cost will also increase. Thus, when are setting reassignment cost as a constraint, we scale it to the number of switches S with a scale factor γ.

$$\sum i \in s, j \in C x_{ij} z_{ij} \leq \gamma * S \tag{4}$$

B. PREF-CP Implementation

PREF-CP works proactively to enable system continue working in case of a controller failure in an SDN architecture. A controller backup approach to ensure that network can be prevented from failures; and even if failure occurs, network can recover from failures quickly. PREF-CP evaluates the current switch-to-controller assignment and calculates a backup map considering each controller might fail. For k number of controllers, k backup maps are calculated. If a reassignment is performed, the PREF-CP recalculates the switch-to-controller assignment considering possible future failures. PREF-CP contains two modules as depicted in Figure 1 and explained below:

TABLE I: Model Parameters.

Symbol	Definition
P_v	Probability of switch failure
P_e	Probability of link failure
$\tilde{P}$	Probability of failure
$\tilde{R}$	Reassignment cost
α	Objective function weight multiplier
γ	Reassignment cost multiplier
σ	Controller load distribution of PACKET_IN messages
U_c	Maximum number of requests that controller C can handle
R_n	Number of requests of each device n

- **Monitoring Module - MM** tracks controllers' health and pulls relevant statistics. The statistics contain x_{cn}:current assignment, L_{cn} controller loads, Probability of Failures P_n. Monitoring module also detects failures occurred in control plane, and in case of controller malfunction informs reassignment module.
- **Reassignment Module - RM** periodically checks the collected statistics from the monitoring module and calculates reassignment map accordingly. In case of controller failure, reassignment module performs new assignment based on pro-actively calculated backup map.

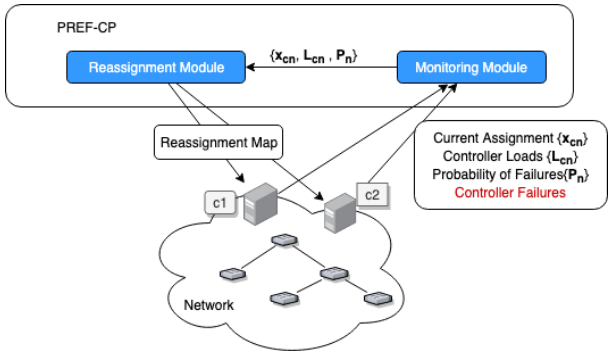

Fig. 1: PREF-CP Architecture

IV. Problem Formulation

A. System Model

In our system model, the SDN physical network is presented as a graph which is denoted as $G(V, E)$, where V is the set of nodes and E is the set of links. The set of controllers is denoted as $V_c \in V$.

Objective. The goal of the proposed strategy is to keep in balance with minimizing controller load distribution, minimizing probability of failure. when a failure occurs, new placement of the controllers or the reassignment of switches should be calculated as probability of failure and load distribution are minimized.

$$Min(\sigma + \tilde{P}) \tag{5}$$

Constraints.

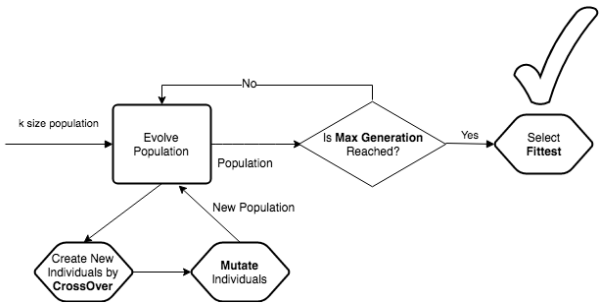

Fig. 2: Genetic Algorithm

Device will be controlled by exactly one controller

$$\sum_{c \in C} x_{n,c} = 1, \quad \forall_n \in N \tag{6}$$

Controller Capacity Cannot be Exceeded

$$\sum_{n \in N} x_{c,n} R_n \leq (1 - \alpha_c) U_c \tag{7}$$

α values fall in range [0,1]

$$0 \leq \alpha \leq 1 \tag{8}$$

Reassignment Cost γ can not exceed 0,8 of S in the network

$$\sum i \in s, j \in C x_{ij} z_{ij} \leq \gamma * S \tag{9}$$

V. Assignment Algorithms

A. Random Assignment

Although this is usually not a practical algorithm, it is typically used as a baseline case for performance evaluation. In random assignment algorithm, each candidate controller have a uniform probability of hosting a switch. In that case, assignment algorithm randomly chooses a controller among all potential ones.

B. Genetic Algorithm

A genetic algorithm is a search heuristic that reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

1) Fitness Function: The performance of a genetic algorithm relies on how well a fitness function is derived. Fitness function in our approach considers minimizing maximum probability of failure of the shortest path of a switch of the controller (P) and minimize load distribution among controllers (σ).

Equation 10 is used during crossover for choosing which controller a switch will be assigned to, chrome evaluation. That is, when the crossover operation has to choose a better gene from two parent genes during the crossover, equation 10 is used to evaluate these two parent genes.

$$Eval = \alpha * \sigma + (1 - \alpha) * P \tag{10}$$

Algorithm 1 Genetic Algorithm

Require: Input: Topology, Load, CrossoverFunction, PopulationSize

Ensure: Generate solution x^c initialize R_{max} and S_{best}

```
Population ← InitializePopulation(PopulationSize)
Evaluate population
S_best ← BestSolution(population)
while population has not converged do
    Selection : Parents ← SelectParent(Population, PopulationSize)
    Children ← 0
    Crossover
    Compute fitness
    if Better fit then
        Pick Parent
    end if
end while
```

C. Inter-controller Greedy Algorithm (ICA)

This algorithm generates a list of the potential assignments, which is ranked increasingly based on failure probabilities of the shortest path from switch to controller and load of switches. Then, it chooses one switch at a time for assigning to controllers. The algorithm iterates until all switches are assigned [15].

Algorithm 2 ICA.

Require: Input: Topology G, Switch Load S, Number of Controllers N, Probability of Failures P

```
for i in Devices Size do weight[i] = CalculateDeviceWeight (L, P)
end for
Sort switches s in ascending order of their weights considering failure properties P_f and loads L_n as set S'
while Devices left to add to Assignment Map do
    for j in Controllers Size do
        Among all devices select the device lowest weight from S'
        Add switch s_i to Controller c_j's backup map
    end for
end while
return AssignmentMap
```

VI. Performance Evaluation

We used ONOS Nightingale 1.13.1 as SDN controller to implement the controller plane and Mininet 2.2.1 for creating network topologies to implement data plane. All simulations are ran on a physical machine installed operations system Ubuntu 14.1 with 8GB RAM and Intel Core i7 6700HQ CPU. In our system, controllers are running inside Docker containers. We ran Mininet on that physical machine and five ONOS instances in Docker 17.09 containers for simulating the distributed controller plane. The switches run in Open vSwitch mode to deliver OpenFlow functionality.

For network traffic, we generated the traffic flows of VoIP, video and two game traffic with Counter Strike characteristics related to the active phase of the game or an idle player using D-ITG traffic generator 2.8.1 [16]. We run our experiments tree topologies T1:40 switches, T2:85, T3:63, T4:121 and internet2 OS3E topology T5:34 switches. T2 and T4 are used for testing algorithms runtime. Simulations are run on topologies T1, T3 and T5.

TABLE II: Simulation Parameters

Parameter	Symbol	Value
Number of switches	S	{34, 40 , 63, 85, 121}
Weight parameter	α	{0.0, 0.5, 0.7, 1.0}
Maximum Population Size	θ	50
Number of controllers	V	5
Reassignment Cost Multiplier	γ	0.8

A. Simulation Results

1) Load Distribution: Load distribution characteristics can be affected by several parameters of the network; specifically reassignment algorithms, α values and number of switches in the network. To see the effects of these parameters, we compared load variance of number of PACKET_IN messages that are distributed among controllers. Increasing α, increases weight of the load distribution as defined in Equation 5, thus load distribution σ decreases as α increase. The test results are showing that in Figure 3, PREF-CP distributes controller load by 8.38 %, 11.5 % ,16.6% better from ICA for α values 0, 0.5, 1, respectively. Random assignment is plainly for reference and it is not affected by the α since it assigns switches to random controllers without considering load distribution. For random algorithm, load distribution value equals to 9284.9 messages, which is higher than both PREF-CP:3933.9 messages and ICA: 4438.9 messages. Although ICA algorithm outperforms random assignment, it is not as good as PREF-CP algorithm. The load distribution σ per algorithms can be seen in Table III.

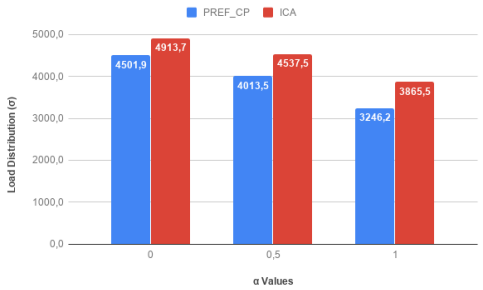

Fig. 3: Number of PACKET_IN Messages Distribution for Algorithms

The results of normalized values for load distribution of PREF-CP on different sized topologies: T1-40 switches, T3-63 switches and T5-34 switches is displayed incTable IV. For

TABLE III: Load Distribution For Single Controller Failure

Algorithm	**Load Distribution σ**
PREF-CP	3933.9
ICA	4438.9
Random	9284.9

$\alpha = 0$, topology sizes affect load distribution performance more than for $\alpha \geq 0.5$. Increasing number of switches does not negatively impact the performance of PREF-CP algorithm. For greater α values, the network size is less important at load distribution performance of PREF-CP algorithm.

TABLE IV: PREF-CP Load Distribution vs S

Number of Switches (S)	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
34	0,373	0,360	0,356
40	0,369	0,363	0,356
63	0,364	0,359	0,350

2) Impact of α on PREF-CP performance: As we change α to prioritize load distribution or probability of failure in our optimization objective, these components behave as expected. When α is 0, PREF-CP tries to find an assignment algorithm with lower PoF. As stated in constraint-8, α values range in [0,1]. We expected that as α increase, PREF-CP will choose calculate assignment map considering load distribution σ over PoF in Table V.

TABLE V: Load Distribution For Two Controller Failure Scenario - PREF-CP on Internet 2 Topology

Objective Function	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
Load Distribution σ	0.371	0.359	0.356
PoF	0.389	0.433	0.465

3) PoF Characteristics: We compared the reliability parameter, probability of connectivity failure, for different algorithms to see the performance of algorithms.

There are basically two aspect in network that effects PoF. First one is α, lower α values result in better PoF. As seen in Figure 4, PREF-CP outperforms ICA and random assignment algorithms. ICA algorithm is slightly higher than random. However, the random algorithm is for reference since it assigns switches randomly regardless of load or PoF.

Second one is, as seen in Figure 5, when number of hops in topology increase probability of failure is also increasing. This outcome is expected because PoF directly depending on the number of the switches and links between from node a to b. If we compare two tree topologies T1 and T3; T1 has depth of 4 and T3 has depth 6; average PoF values are 0,373 and 0,509 respectively for average of α values.

4) Computational Time: For the complexity aspect, computational time of the proposed PREF-CP algorithm was compared with ICA, for different network sizes, i.e. number of switches(S) to see the effect of number of switches.

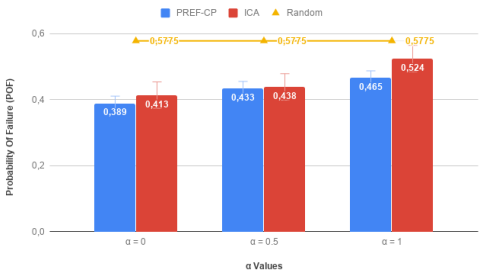

Fig. 4: PoF calculated by different algorithms

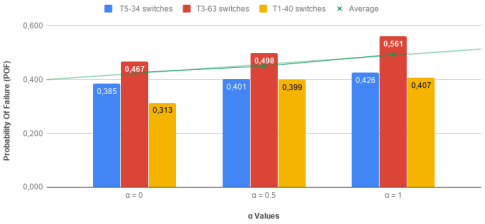

Fig. 5: PoF values for different topologies

As seen in Table VI, when S increases in the network, calculation time for reassignment map also increases. For PREF-CP, it is substantially higher compared to ICA algorithm as expected. This result renders the complexity-assignment quality trade-off since PREF-CP is most time-consuming albeit being better in optimization.

B. Cascaded Failure Characteristics

When two controllers fail in tandem, the residual load is distributed among the remaining $V - 2$ (three in our case) controllers. In Figure 6, load distributions are shown with respect to different topologies. Tree topologies are effected by multiple controller failures more than Internet2 topology.

Figure 7 shows load distribution for all algorithms in case of one controller failure and two controller failures. PREF-CP algorithm outperforms both random and ICA algorithms. The presented results shown in Figure 6 are the average of load distribution for three different topologies.

To see more in details, how control load is distributed on each individual controllers on Internet2 topology with 34-switches for α : 0.5. In this case, Controller-1* and then Controller-3* fails one after another and we compare PREF-CP performance in handling two controller failures. Load distributions on average for *before-failure*, *after-one-failure* and *after-two-failures* are shown in Table VII.

TABLE VI: Run-times for different assignment algorithms (in msec).

Assignment Algorithms	Number of switches		
	40	85	121
GA	2480	7540	12100
ICA	12	27	80

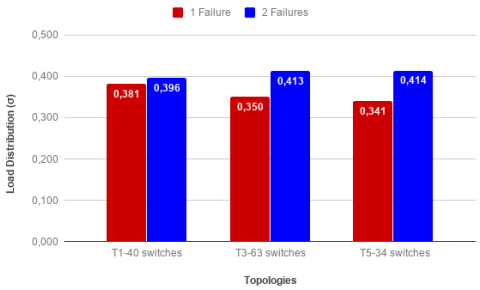

Fig. 6: Load distribution under controller failures for different topologies.

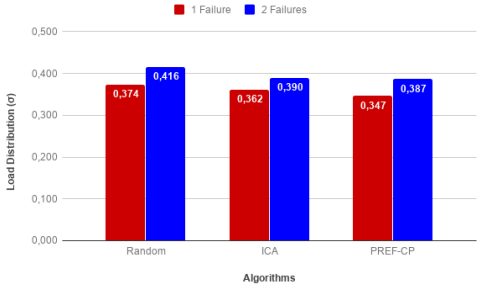

Fig. 7: Load distribution under two controller failures for different algorithms.

TABLE VII: Load Distribution For Two Controller Failure Scenario - PREF-CP on Internet 2 Topology

Time	**Load Distribution σ**
Before Failure	4260.1
After C1 Fails	3773.9
After C3 Fails	5182.2

VII. CONCLUSION

In this paper, we consider control plane failures and how to recover from them with an efficient proactive reassignment approach. To address this challenge, we propose a proactive switch assignment PREF-CP in case of controller failures using genetic algorithm considering controller load distribution, reassignment cost and probability of failure. Moreover, we compare the performance of our scheme with random assignment and greedy algorithm ICA. Our test results show that when controllers' load distribution and probability of connectivity failure are considered PREF-CP performs better than ICA algorithm. Secondly, When reassignment map calculation time is considered, ICA is a faster algorithm as expected. However, since are making our calculation before a failure happens the impact of that calculation time may not have that much of importance.

For future work, these algorithms can be run on different topology types and larger network sizes. For a more valid comparison, latency and throughput results can also be considered. The loads of controllers may be distributed efficiently, however if latency between controller-switch is higher, network performance might be affected negatively. One more heuristic algorithm can be implemented to have a more wider scope comparison.

ACKNOWLEDGMENT

This work was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant number 117E165.

REFERENCES

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[2] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 36–43, July 2013.

[3] M. zelik, N. Chalabianloo, and G. Gr, "Software-defined edge defense against iot-based ddos," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*, Aug 2017, pp. 308–313.

[4] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, Apr. 1999.

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

[6] M. Obadia, M. Bouet, J. Leguay, K. Phemius, and L. Iannone, "Failover mechanisms for distributed sdn controllers," in *2014 International Conference and Workshop on the Network of the Future (NOF)*, vol. Workshop, Dec 2014, pp. 1–6.

[7] 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. New York, NY, USA: ACM, 2012, pp. 7–12.

[8] L. F. Mller, 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*, Dec 2014, pp. 1909–1915.

[9] K. Fang, K. Wang, and J. Wang, "A fast and load-aware controller failover mechanism for software-defined networks," in *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, July 2016, pp. 1–6.

[10] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient wans," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013, pp. 1–5.

[11] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 2665–2670.

[12] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient sdn control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Sep. 2016, pp. 253–259.

[13] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. M. Machuca, "Characterization of failure dynamics in sdn controllers," in *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Sep. 2017, pp. 1–7.

[14] Y. Zhou, Y. Wang, J. Yu, J. Ba, and S. Zhang, "Load balancing for multiple controllers in sdn based on switches group," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sept 2017, pp. 227–230.

[15] S. Guner, H. Selvi, G. Gur, and F. Alagoz, "Controller placement in software-defined mobile networks," in *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, May 2015, pp. 2619–2622.

[16] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.