Optimal and Dynamic SDN Controller Placement

Nadia Mouawad*†‡, Rola Naja*†§ and Samir Tohme†¶
*Doctoral School of Science and Technology, Lebanese University, Lebanon
†Li-Parad Laboratory, University of Versailles Saint Quentin, France
‡mouawad.nadia@gmail.com §rola.naja@ul.edu.lb ¶samir.tohme@uvsq.fr

***Abstract*—Software Defined Networking (SDN) is an emerging paradigm that separates control and data plane. This technology will be a key component in designing 5G networks that involve essentially higher capacity and lower latency. With SDN, traffic and network devices are controlled using SDN centralized controllers. Several studies denoted the limitation of using one controller in a large-scale network and advocated the use of multiple controllers. The main problem appears in the placement of these controllers in the network, while considering several criterion such as latency, network load and connectivity. To this end, our work solves the controller placement problem and deals with network load using dynamic switch migration algorithm. The validation of the algorithm and the performance analysis showed that the proposed topology is stable and presents a reduced number of load balancing occurrences.**

***Keywords*—Software Defined Networking, Controller placement, Load balancing and 5G Networks**

I. Introduction

5G networks are expected to have an explosive growth of data traffic and to provide billions users a high capacity and low latency [1]. These networks are more flexible than the traditional networks and guarantee enhanced communication across heterogeneous technologies. Nevertheless, these features bring a lot of challenges, especially in terms of providing low latency, high reliability and flexibility.

SDN [2] is gaining a lot of attention since it enables centralized network provisioning and lower operating costs. Therefore, 5G networks will take advantages of using this technology to face the above mentioned challenges.

SDN is a new paradigm that separates the control plane (SDN controllers) from the data plane (switches). Openflow protocol [3] defines an interface between the data and control plane to describe the interaction between controllers and the switches.

SDN controller receives openflow requests (Packet _In) that should be processed in order to organize the data plane. Therefore, using a single controller brings several drawbacks such as controller overload in large-scale network and the inability of providing network decisions in a tight time window. Therefore, the use of multiple controllers is primordial. To this end, several studies ([4], [5]) designed a hierarchical SDN topology using multiple controllers as shown in figure 1. The proposed architecture presents a global controller having an abstract network view in addition to local controllers each one managing a domain of switches.

The SDN architecture incurs the following challenging issues:

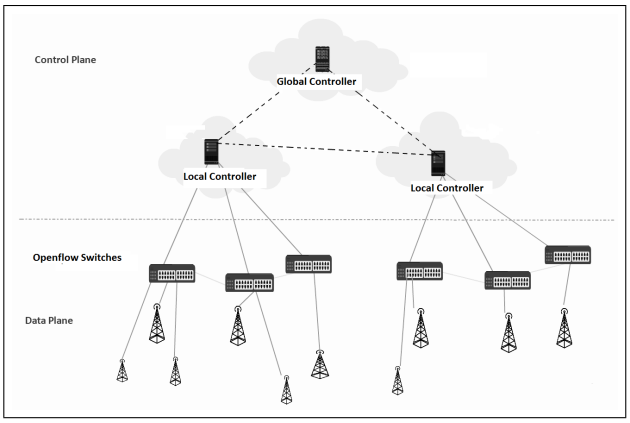

Fig. 1. 5G SDN based architechture

1) The optimization of the controller placement has a paramount importance in terms of network latency.
2) The determination of the switches embedded in one controller domain should be carefully studied since it impacts network load.
3) A load balancing algorithm should be derived in case of controller overload.

The controller placement problem has attracted many researchers ([6],[7],[8],[9]). In [10], after proposing an algorithm for placing controllers, a flow based dynamic switch migration algorithm is presented to adjust controllers load. This algorithm consists of migrating the boundary switches to neighbor controllers. However with this method, the controller load is not accurately adjusted, since all boundary switches may be sending a low average of packets to the controller. As a result, a large number of switches should be migrated to balance the load.

In [11], the controller placement problem was tackled while taking into consideration three criterion: connectivity, capacity and link failure recovery. This work aimed at maximizing the number of disjoint paths between devices and controllers respecting the controller capacity constraints. While authors took into account connectivity metrics, they neglected the controller-switch latency.

In [12], the goal is to find the optimal controller placement while minimizing latency and considering the resilience (capacity to recover rapidly after a failure). Authors in [12] did not take into consideration the load constraint. When applying the proposed algorithm, the number of switches in

each controller domain is unbalanced, and this will induce the risk of flooding controllers.

The before mentioned papers considered one or two important parameters for the controller placement while neglecting important constraints. Our work differs from these literature works since it brings a global solution by taking into account all the following parameters: the connectivity, the controller-switch latency, inter-controllers latency and controllers load.

To this end, we design an algorithm to solve the controller placement problem using a quadratic program that provides as an output the locations of the controllers and number of switches in each domain.

After calculating the optimal controllers placement, we provide an enhancement for the proposed topology. In fact, the network may suffer in certain cases from fluctuation due to large number of openflow requests rate. Therefore striving to prevent congestions, we developed an algorithm for dynamically migrating switches in case of controller overload.

The rest of this paper is structured as follows: section II derives our proposed controller placement and load balancing algorithms. Section III presents our simulation results and performance analyses. Finally, we conclude our work in section IV.

II. Controller Placement and Load Balancing Algorithms

In this paper, we present our work that aims at finding an optimal dynamic SDN controller placement that reduce controllers-switches latency, inter-controllers latency and controllers load. The proposal is split into three parts as follows:

1) **Optimal Controller Placement** that takes into account the following inputs: network topology, inter-nodes distance, connectivity between nodes, average request rate, controller maximal capacity, minimal distance between two controllers. The algorithm is designed as an offline application that derives an optimal strategy for controller placement.
2) **Load balancing algorithm** that aims at enhancing the derived topology in order to balance the network load. It takes into account the controllers load and their maximal capacity and switches request rate. The algorithm is implemented as SDN application in each local controller.
3) **Switch migration algorithm** that presents a heuristic to choose the corresponding migrated switches and their target controllers. It takes into account the derived optimal topology, controllers load, maximal controller capacity and the minimum allowed distance between migrated switches and target controllers. The heuristic is designed as SDN application running in the global controller.

Figure 2 depicts the integration of the proposed algorithms in SDN planes. Next, we explain these algorithms in details.

A. Optimal controller placement

The study objective is to find the optimal placement of C controllers while minimizing the distance (i.e the latency) and maximizing connectivity between adjacent nodes. The proposed algorithm will take the following inputs:

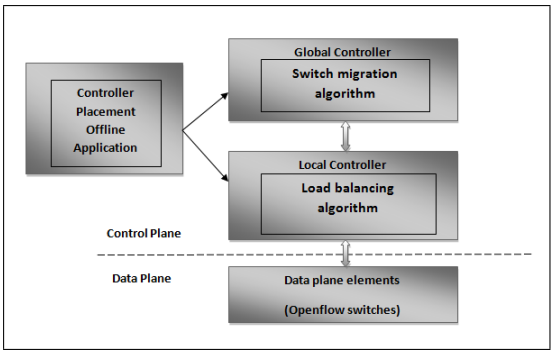

Fig. 2. Integration of the proposed algorithms in SDN planes

- *G(N,E)*: a graph that denotes the network topology, where $n \in N$ represents forwarding devices and edges $(m,n) \in E$ represent bi-directional links.
- *Dis*: a (NxN) matrix where Dis_{ij} represents the distance between nodes i and j. We represent the distance by the number of hops.
- *Cn*: a (NxN) matrix where $Cn_{ij} = 1$ if there exists a link between nodes i and j.
- λ_n denotes the average requests rate from each device $n \in N$.
- μ_{max}: the maximum capacity of the controllers.
- α: percentage of the maximum capacity at which we consider that the controller is overloaded.
- M: maximal threshold latency between two controllers.

It is noteworthy that taking into account the connectivity leads to reduce the number of requests received by the controller. Consider a node connected to several switches, this node is preferred to be a controller rather than a normal switch.

As a result, we obtain the following output:

- X: a (NxN) matrix where $x_{nc} = 1$ if a device n is mapped to a controller c and 0 otherwise.
- Y: a (1xN) matrix where $y_n = 1$ if a controller is placed on the node n and 0 otherwise.

The following Quadratic Problem (QP) solves the matrices X and Y:

$$\text{minimize} \quad \sum_i \sum_j (Dis_{ij} - Cn_{ij}) * x_{ij} * y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in C} y_i = C \qquad \forall i \in C. \tag{2}$$

$$\sum_{i \in C, j \in N} x_{ij} = N \qquad \forall j \in N \; \forall i \in C. \tag{3}$$

$$\sum_{i \in C} x_{ij} = 1 \qquad \forall j \in N; \forall i \in C. \tag{4}$$

$$x_{ij} \leq y_i \qquad \forall i \in N \; \forall i \in C. \tag{5}$$

$$x_{ii} = y_i \qquad \forall i \in N \; \forall i \in C. \tag{6}$$

$$\sum_{j \in N, j \neq i} \lambda_j * x_{ij} \leq \mu_{max} * \alpha \qquad \forall i \in C. \tag{7}$$

$$\sum_{j \in N} Dis_{ij} * y_i * y_j \leq M \qquad \forall i \in N. \tag{8}$$

$$x_{ij} \in \{0,1\}\ \forall j \in N; \qquad \forall i \in C. \tag{9}$$

$$y_i \in \{0,1\} \qquad \forall i \in C. \tag{10}$$

The goal in (1) is to minimize the controller-switch latency (the distance in our case) and maximize connectivity between the assigned switches and their controller. Equations (2) and (3) assess that the number of assigned controllers should be equal to the number of used controllers, and the number of assigned switches is equal to the number of considered nodes. Equation (4) guarantees that each device $j \in N$ will be controlled by one controller. Equation (5) makes sure that the assigned controller must be active. By (6), we denote that each node designed to be a controller belongs to the set of the assigned switches. Constraint (7) guarantees the controller load: the average number of requests from all the assigned switches is in the acceptable range. Finally, in (8), we include into the problem the inter-controller latency and make sure that the distance that separates each two controllers should not exceed the maximal predefined distance M.

B. Load balancing algorithm

In the previous section, we formulated the controller placement problem by considering several metrics. The problem solution computes the controllers/switches connected graph. Once controllers are placed in the calculated position, the network topology will be static.

Nevertheless, real networks are subject to a dynamic flow variation. In fact, switches may flood controllers with a high packet_in request load. Consequently, the static network topology will not be able to cope with the load change.

Therefore, we elaborated Algorithm 1 that aims at migrating switches to adjust controller load. The main goal is to calibrate load of the overloaded controller by migrating the minimum number of switches to a target controller in order to keep the topology as close as possible to stability.

Algorithm 1 Load balancing algorithm

A traffic monitor is present in each controller in order to collect packets statistics.
if (controller load $\geq \alpha.\mu_{max}$) **then**
  Solve the quadratic program (QP1) as presented in the following section
  Migrate the calculated switch to the target controller
  Calculate the remaining controller load after switch migration
  if (remaining controller load $\geq \mu_{max} * \alpha * \beta$) **then**
    Resolve QP1
  else
    Derive the migrated switches, their number, and the target controller
  end if
end if

C. Switch migration algorithm

The present subsection details the switch migration application that is executed in the global controller in case of overload. To this end, we solve a quadratic program in order to determine the migrated switches and target controllers. Migrated switches are chosen according the following constraints:

1) The target controller should be chosen in a way that its capacity could accept the migrated switches load.
2) A minimal number of switches should be migrated to the target controller.
3) The migrated switch should reduce the overloaded controller load.

To this purpose, we formulated the following quadratic program (QP1) that takes as inputs:

- o: the overloaded controller number.
- D_2: a ((C-1)x N) matrix composed by the controllers rows of matrix *Dis* except the overloaded one. For reader clarity, D_2 elements are filled as follows: D_2[k][j]=*Dis*[i][j] where $y_i = 1$ and $i \neq o\ \forall k \in (C-1), \forall i \in C, \forall j \in N$.
- λ_{ij} is the packet requests rate from switch j to controller i
- β: the percentage of the maximum capacity that should be respected in each controller in order to avoid the overload.
- μ_{max}: the maximum capacity of the controller.
- D: the maximum allowed distance (number of hops) between the migrated switch and its new controller.
- α: percentage of the maximum capacity at which the controller is overloaded.

As a result, we obtain the following output:

- S: a (1xN) matrix that determines the position of the migrating switch; $s_i = 1$ indicates the switch i should be migrated.
- T: a (1x(C-1)) matrix where $t_i = 1$ indicates that controller i is the target controller.

At this stage, we solve the following quadratic program(QP1):

$$\max \qquad \sum_i \sum_j \lambda_{oj} * s_j * t_i \tag{11}$$

$$\text{s.t.} \sum_i \sum_j D_{2ij} \leq D \qquad \forall i \in (C-1) \forall j \in N. \tag{12}$$

$$\sum_i t_i = 1 \qquad \forall i \in (C-1). \tag{13}$$

$$\sum_i \sum_{j,j \neq i} \lambda_{ij} * t_j + \sum_j \lambda_{oj} * s_j \leq \mu_{max} * \alpha \tag{14}$$

$$\sum_j s_j = 1 \qquad \forall j \in N, \lambda_{oj} \neq 0. \tag{15}$$

$$s_j \in \{0,1\} \qquad \forall j \in N. \tag{16}$$

$$t_i \in \{0,1\} \qquad \forall i \in (C-1). \tag{17}$$

The main goal in (11) is choosing a high load switch in order to minimize the number of migrated switches while taking into account the latency constraint in equation (12). This is achieved under the following constraints: (13) and (17) show

TABLE I
SWITCH-CONTROLLER DISTANCE COMPARISON

	Zoo topology	Internet2	NSFNET
ODCP	3	4	2
CPFD	4	5	3

TABLE II
INTER-CONTROLLER DISTANCE COMPARISON

	Zoo topology	Internet2	NSFNET
ODCP	3	3	3
CPFD	5	5	3

that one target controller is chosen. (15) and (16) ensure that one switch in the overloaded controller domain is migrated. Equation (14) guarantees that the load of the migrated switch added to the total load of the target controller should satisfy the maximum acceptable load.

III. Performance Analysis

In this section, we evaluate the robustness of the proposed dynamic optimal SDN topology and provide a performance analysis. We proceed first by evaluating the controller placement by considering the latency metric. Second, we study the network performance by evaluating the topology robustness.

A. Controller placement evaluation

In order to validate the proposed controller placement algorithm and to show its capability of reducing the switch-controller distance and inter-controller distance, we considered several networks topologies proposed in the literature: Zoo Toplogy(50 nodes) [13], Internet2 (34 nodes) [14] and NSFNET topology (14 nodes) [10]. We studied the average distance in terms of number of hops between switches and controllers and inter-distance controllers and compared our results with the ones given by the algorithm proposed in [10]. In order to simply mentioning reference [10], we give it the name CPFD based on its title (Controller Placement and Flow based Dynamic Management Problem towards SDN). In addition, we call our proposal ODCP referring to Optimal and Dynamic Controller Placement. Tables I and II show that our proposal results in minimizing the distance between the switches and the controller and the inter-controller distance.

In the following, we adopt the NSFNET topology [10]. NSFNET (figure 3) nodes should be clustered into two separated domains; each controlled by a controller and remaining nodes are switches. Figures 4 and 5 illustrate the derived topology with our proposal (calculated for $M = 3$) and with CPFD respectively.

B. Performance evaluation

The robustness of a topology is achieved when it manages load variation while inducing the least number of switch migrations, and the least number of overloads. In fact, the optimal controller placement strategy conducts to a stable topology that presents a reduced number of controller overloads. We proceed in two steps in order to evaluate our work performance:

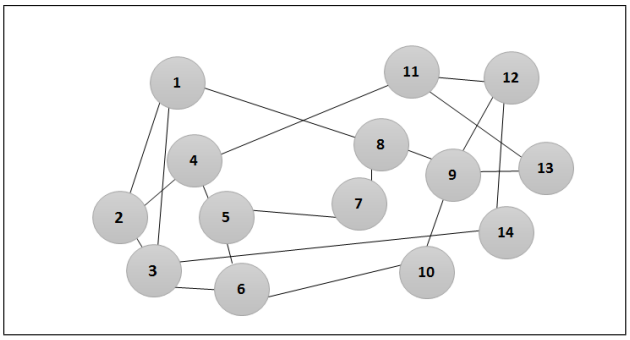

Fig. 3. NSFNET nodes

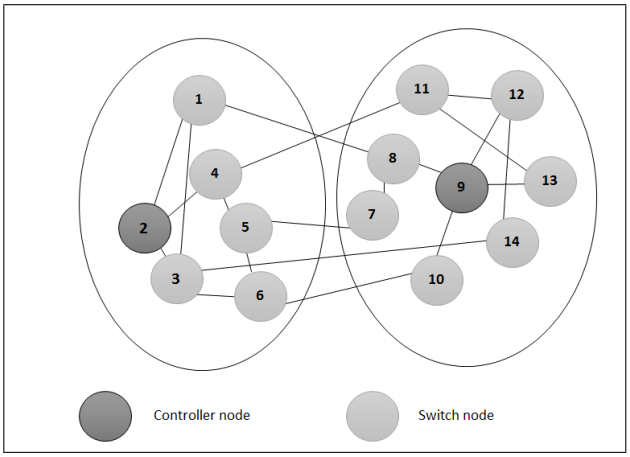

Fig. 4. NSFNET topology studied in ODCP

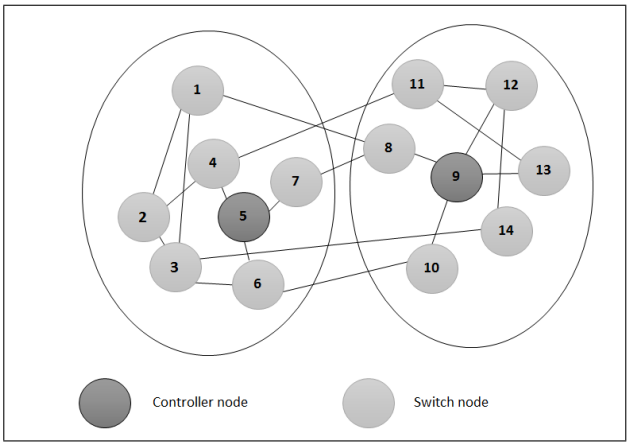

Fig. 5. NSFNET topology studied in CPFD

First, we evaluate the switch migration in the context of 3 scenarios:

1) **Scenario 1**: Controller load before and after switch migration in normal network state.
2) **Scenario 2**: Controller load before and after switch migration in case of heavy Packet_In request load in all switches.
3) **Scenario 3**: Controller load before and after switch migration in case of low Packet_In request arrival from boundary switches.

Next, in order to assess the robustness of our algorithm and to compare it with that of CPFD, we derive for each scenario the following performance parameters:

1) **Number of overloads**: number of overload occurrences.
2) **Number of migrated switches**: number of migrated switches from overloaded controller domain.

In the following, we study each scenario separately and compare its results with the work presented in CPFD.

Scenario 1: Controller load before and after switch migration in normal network state: Figure 6 depicts the controllers load achieved in our proposal and in CPFD. This figure shows that in ODCP controllers load is less than the overload threshold ($\mu_{max} * \alpha = 90\%$); this will prevent the network from a second overload incident. However, with CPFD controller load after switch migration is 89% . This value makes a next overload likely to occur.

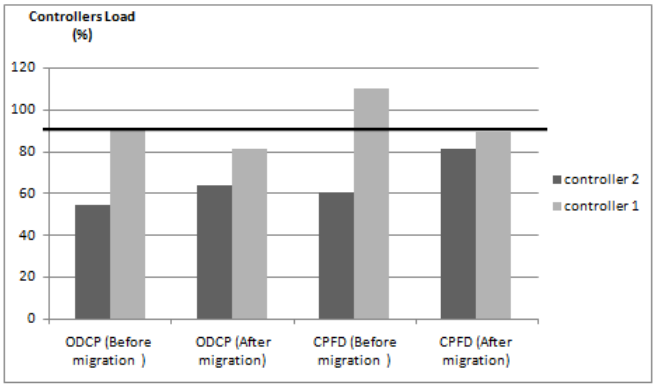

Fig. 6. Controllers load before and after migration

Scenario 2: Controller load before and after switch migration in case of heavy load in all switches: The second scenario tackles the controller load in a saturated network.
We consider the following inputs: $\mu_{max} * \alpha = 90\%$ and $\beta = 0.9$. Figure 7 depicts controllers load in this case. With the algorithm adopted in CPFD, we can not find a solution for the switch migration algorithm. Contrarily to our work that provides load balancing. Indeed, figure 7 shows that in ODCP controller 2 is overloaded and controller 1 load is far below the threshold. As a result, we can easily migrate switches from domain 2 to domain 1 and achieve the network load balancing. These results validate the derived topologies in ODCP and CPFD.

Scenario 3: Controller load before and after switch migration in case of low request rate from boundary switches: In this scenario, we consider a low packet rate from boundary switches, and study the controllers load before and after triggering the switches migration algorithm.

After applying the method proposed in CPFD, three switches are migrated from domain 2 to domain 1 while the load balancing problem is not solved as shown in figure 8. This is due to the fact that in CPFD, the boundary switches are migrated and in this case all of them send low packets requests to the controller. On the other hand, our proposal succeeds to achieve load balancing.

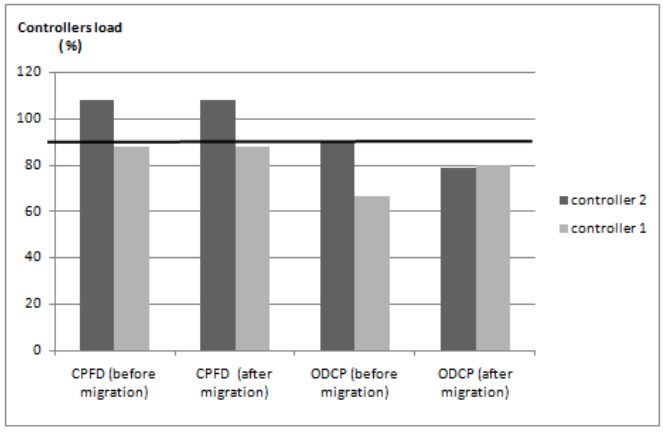

Fig. 7. Controllers load in case of heavy packets arrival from all switches

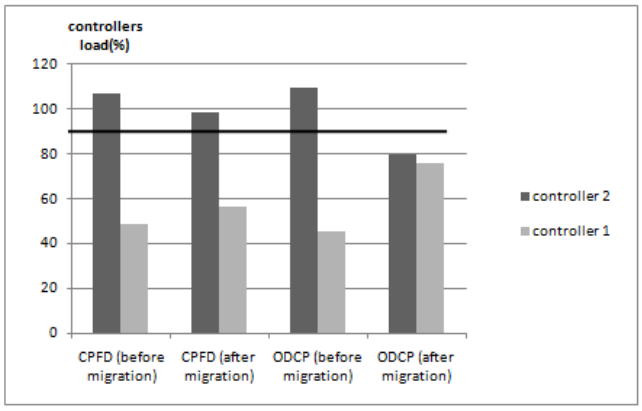

Fig. 8. Controllers load in case of low packets arrival from boundary switches

Number of overloads : Figure 9 shows that, despite the heavy requests arrival from all the switches, ODCP results in a reduced overload rate (75% less than CPFD).
This is explained in figures 4 and 5 where the proposed topology is not equally distributed as in CPFD, which leads to reduce the number of overloads in domain 1.

Number of migrated switches: Figure 10 illustrates the number of migrated switches from one domain to another in several network states:

- Normal requests rate
- Heavy requests rate from boundary switches
- Heavy requests rate from non boundary switches
- Low requests rate at boundary switches

In figure 10, we can notice that ODCP reduces the number of migrated switches except in case of heavy requests rate from boundary switches. In fact, CPFD migrates boundary switches which helps adjusting controllers load in this case.

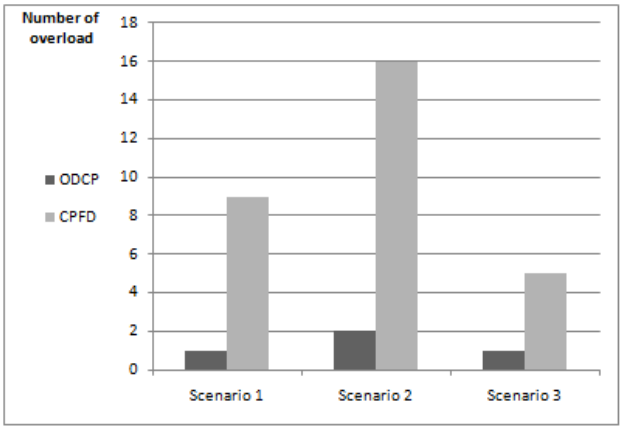

Fig. 9. Number of overload

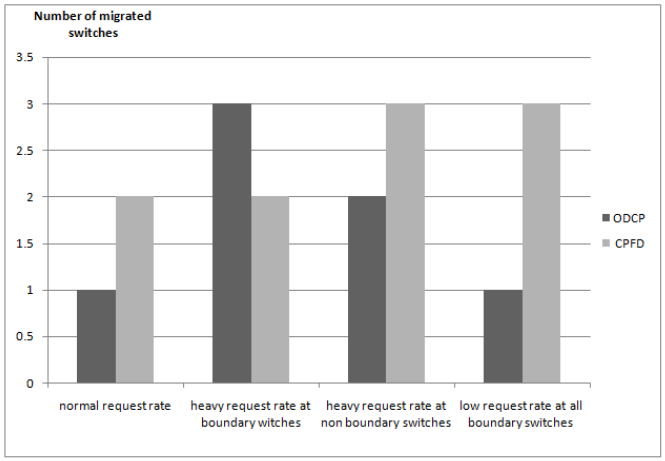

Fig. 10. Number of migrated switches

IV. Conclusion

This research paper addresses the optimal placement of SDN controllers. In fact, SDN topology should be designed carefully to guarantee performance parameters. To this end, we presented a novel work that aims at designing an optimal and dynamic topology for SDN networks. We considered the hierarchical SDN topology proposed in the literature where the SDN control plane consists of several local controllers controlled by a global controller. In this work, we designed an offline application that derives the optimal controller placement, we implemented a local controller application for load balancing and a switch migration application global controller.

We demonstrated the stability of our topology by showing that within a sufficient duration of time we achieve a minimum number of overloads and a minimal number of switch migrations from one domain to another. Performance analysis showed that our proposal outperforms other literature algorithms in terms of network stability, minimizing the number of controllers overloads and minimizing the probability of overloads occurrences.

Acknowledgments The work was funded by the Lebanese University and the AUF ' Projet de cooperation scientifique interuniversitaire' (PCSI).

References

[1] G. P. A. W. Group *et al.*, "View on 5g architecture," *White Paper, July*, 2016.
[2] "Sdn architecture overview," *Open Networking Foundation*, 2013.
[3] "Openflow switch specification," *Version1.5.0*, vol. 1, no. 0, 2014.
[4] L. S. P. Ameigeiras, J.J. Ramos-Munoz, "Link-level access cloud architecture design based on sdn for 5g networks," *IEEE Network*, vol. 29, no. 2, pp. 24–31, 2015.
[5] D. Z. MA Zheng, ZHANG ZhengQuan, "Key techniques for 5g wireless communications: network architecture, physical layer, and mac layer perspectives," Science China Information Science, Tech. Rep., 2015.
[6] E. Borcoci, T. Ambarus, and M. Vochin, "Distributed control plane optimization in sdn-fog vanet," *ICN 2017*, p. 135, 2017.
[7] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.
[8] M. F. Raouf Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and Boutaba, "Dynamic controller provisioning in software defined networks," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM)*, 2013.
[9] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.
[10] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards sdn," in *Proceedings of the International Conference on Communication Workshop (ICCW)*. IEEE, 2015, pp. 363–368.
[11] L. F. Marinho P Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and Barcellos, "Survivor: an enhanced controller placement strategy for improving sdn survivability," in *Proceedings Global Communications Conference (GLOBECOM)*. IEEE, 2014, pp. 1909–1915.
[12] Q. L. Y. J. Sheng Guo, Shu Yang, "Towards controller placement for robust software-defined networks," in *Proceedings the 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.
[13] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.
[14] R. Dantu, T. A. Anderson, R. Gopal, and L. L. Yang, "Internet2 open science, scholarship and services exchange," 2004.