# On Controllers' Utilization in Software-defined Networking by Switch Migration

Faroq AL-Tam(✉), Mohammad Ashrafi, and Noélia Correia

Center for Electronic, Optoelectronic and Telecommunications (CEOT), Faculty of Science and Technology - University of Algarve, 8005-139 Faro, Portugal
{ftam,a41301,ncorreia}@ualg.pt

**Abstract.** This work presents a model to solve the switch migration problem in software-defined networking. This model is formulated as a mixed-integer linear programming, and compared against the static mapping approach. Two scenarios of homogeneous and heterogeneous controllers are evaluated. The experimental results show that the dynamic mapping enabled by the proposed model can enhance the controllers' utilization by ≈63% for homogeneous scenario and ≈47% for heterogeneous scenario, while maintaining a low control plane overhead.

**Keywords:** Software-defined networking · Multi-controller SDN
Switch migration · Load balancing

## 1 Introduction

Software-defined Networking (SDN) is an emergent technology that offers a promising software oriented design to manage IP networks [11], Internet of Things (IoT) [3] and 5G networks [5]. In SDN, the data and control planes are decoupled to make the forwarding devices programmable and to promote network scalability and evolution [4]. Network policies are defined in the management plane, materialized by software modules in controllers in the control plane, and carried out by switches in the data plane [11,13]. From a layer perspective, the communication between the data and control planes is made available via standardized **southbound** interfaces (e.g., OpenFlow [7]), while the communication between the management and control plane is usually done through non-standardized **northbound** interfaces [9].

Control plane can be centralized or distributed. In the former, a single controller is responsible for managing all flow requests from the switches, while in the latter, multiple controllers are used. A single controller ensures a unified

This work was developed within the Centre for Electronic, Optoelectronic and Telecommunications (CEOT), and supported by the UID/MULTI/00631/2013 project of the Portuguese Science and Technology Foundation (FCT).


V. Sucasas et al. (Eds.): BROADNETS 2018, LNICST 263, pp. 52–61, 2019.
https://doi.org/10.1007/978-3-030-05195-2_6

knowledge about the network, however, it represents a possible single point of failure [11,21]. Additionally, a centralized control plane does not scale well [21], is susceptible to overloading [19], and can hinder the Quality of Service (QoS) [10]. On the other hand, in multi-controller SDN [7] each controller is responsible for a set of switches (domain). A controller can be master, equal, or slave, where the first two types can process the flow requests from the switches and install the forwarding rules in the switches. A slave controller can only read the switch flow table, but can not update it. Each switch can have multiple equal and slave controllers, but only one master controller. Furthermore, a master controller for a specific switch can be slave controller for another one, and whenever a master controller fails, a slave or local controller can request (via OpenFlow `role-request` message) to become the new master of the affected switches.

The multi-controller paradigm is shown to improve many aspects of SDN [22], but it presents many challenges, especially for controllers' utilization when switch-controller assignments are static. The load of a controller is mainly caused by the processing of the `packet-in` messages sent from the switches [19], and due to network dynamics, the number of these messages vary both regionally and temporally [2]. As a result of these variations, some controllers will be overcommitted (hot spot), while some others will be underutilized (cold spot). This leads to domain failure (and multi-domain failure [18]), or network underutilization. Therefore, switch migration (SM) [6] can be used as a key-enabler for dynamic switch-controller mappings in order to adaptively shift the load of controllers. Whenever overloading is detected in a master controller, it selects a switch from its domain and asks a slave controller to become the new master of this switch.

This work presents a simple and concise modeling of the SM problem as a mixed integer linear programming (MILP), which considers the load balancing of the controllers, and the overhead created by the migrated switches. Two scenarios of homogeneous and heterogeneous controllers are considered.

This work is organized as follows: Related work is described in Sect. 2, and the proposed model is presented in Sect. 3, the experimental results and the conclusions are presented in Sects. 4 and 5, respectively.

## 2 Related Work

SM problem is clearly addressed for the first time in ElastiCon [6]. Since then some approaches have been proposed [22]. SM is not only used for balancing load [12], but also for improving resource utilization [20] and security [17]. SM is usually treated and modeled within linear [1,8] or nonlinear [14,16] models of dynamic switch-controller assignments. In these models, the mappings are calculated for all switches and controllers. However, in SM only a subset of controllers and a subset of switches require reassignments. Therefor, it is more appropriate to model the SM separately. The most recent works that model SM separately are presented in [15,23], but aspects like heterogeneous controllers' utilization are not investigated. In [15], the overloading is detected based on the load diversity matrix of the controllers. A user-defined threshold is used to decide which controller is overloaded. A switch is migrated based on a probabilistic measure. In [23], SM is modeled within the earth mover distance (EMD), which is a histogram matcher that calculates the cost of "morphing" one histogram into another.

# 3 The Proposed Model

The proposed model considers various factors in SM, some of them are not already considered in the literature, like switch importance in its original domain and the ratio between the remaining capacity in the controller and the number of flow requests in the migrated switches. These are two important factors in order to reduce the control plane overhead when exchanging information of the migrated switches.

## 3.1 System Model

The network scenario for switch migration problem is depicted in Fig. 1, and the notation used is described in Table 1. A set $\mathcal{C}$ of controllers is managing a set $\mathcal{S}$ of switches. Each switch is connected to exactly one master controller (thin solid lines) and one or more slave controllers (thin dashed lines). Therefore, each controller is managing a subset (domain) $\mathcal{S}_i \subset \mathcal{S}$ of switches. The latency between controller $c_i$ and switch $s_j$ is known and denoted by $d_{ij}$, and the latency between controllers $c_i$ and $c_{i'}$ is known and denoted by $v_{ii'}$.

At each switch $s_j$, a number of `packet-in` messages per second $r_j$ is generated (usually following Poisson distribution). Each controller has a limited processing capability $\gamma_i$ to process a certain number of `packet-in` messages per second. In addition, each switch has a relative-importance $\theta_j$, which is related to the size of its flow table and the number of neighbor switches in the same domain.

Given a scenario where loads at the controllers are imbalanced, the objective is to find a set of switches to be migrated between controllers, in order to reestablish the load balance, while minimizing the control plane overhead.

## 3.2 Controller Load

The load $\omega_i$ of controller $c_i$ is the aggregation of the flow requests from the switches in its domain $\mathcal{S}_i$:

$$\omega_i = \sum_{s_j \in \mathcal{S}_i} r_j, \quad \forall c_i \in \mathcal{C}, \tag{1}$$

and after switch migration, the load of controller $c_i$, denoted by $\tilde{\omega}_i$, can be calculated as [23]:

$$\tilde{\omega}_i = \omega_i - \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} r_j x_j^{ii'} + \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} r_j x_j^{i'i}, \quad \forall c_i \in \mathcal{C} \tag{2}$$

where $x_j^{ii'}$ is the migration decision variable to be calculated.

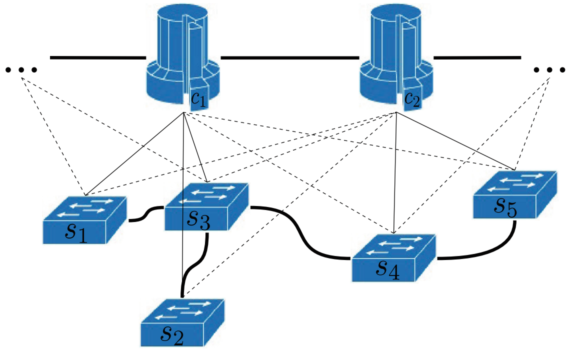



**Fig. 1.** The SDN model. Thick solid lines are the physical inter-plane links, thin solid lines are the switch-controller master mappings, and thin dashed lines are the switch-controller slave mappings.

**Table 1.** Notation used through out this paper.

| Term | Description |
|---|---|
| $\mathcal{C}$ | Set of controllers |
| $\mathcal{S}$ | Set of switches |
| $r_j$ | Number of packet-in messages generated by switch $s_j$ |
| $\theta_j$ | Switch importance (weight) in its domain |
| $\gamma_i$ | Capacity of controller $c_i$ |
| $d_{ij}$ | Latency between controller $c_i$ and switch $s_j$ |
| $v_{ii'}$ | Latency between controller $c_i$ and controller $c_{i'}$ |
| $\omega_i$ | Load at controller $c_i$ |
| $l$ | Minimum load-to-capacity ratio |
| $u$ | Maximum load-to-capacity ratio |
| $x_j^{ii'}$ | Migration decision variable, $x_j^{ii'} = 1$ if switch $s_j$ is to be migrated from controller $c_i$ to $c_{i'}$, zero otherwise |

### 3.3 Cost Function

The cost function is a linear combination of the load balancing and the control plane overhead costs. The load balancing cost is defined as the difference between the maximum and minimum load-to-capacity ratios, denoted by the real-valued variables $u$ and $l$, respectively. The control plane overhead cost, caused by migrating switch $s_j$ from controller $c_i$ to controller $c_{i'}$, is denoted by $\vartheta_j^{ii'}$. Therefore, for two user-defined weights $\alpha_1$ and $\alpha_2 = 1 - \alpha_1$, the cost function is defined as:

$$f = \alpha_1(u - l) + \alpha_2 \sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} \vartheta_j^{ii'} x_j^{ii'} \quad (3)$$

To calculate $\vartheta_j^{ii'}$, let the remaining capacity in controller $c_i$ be denoted by $\hat{\gamma}_i = \gamma_i - \omega_i$. Therefore, the control plane overhead cost is the composition of the following terms:

1. Number of switch migrations: $\sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} x_j^{ii'}$
2. Overall importance of switches: $\sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} \theta_j x_j^{ii'}$
3. Migrated flow requests to remaining capacity ratio: $\sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} \left( \frac{r_j x_j^{ii'}}{\hat{\gamma}_{i'}} \right)$
4. Inter-plane delay and control plane delay: $\sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} r_j d_{i'j} x_j^{ii'} + \sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} \sum_{s_j \in \mathcal{S}} \theta_j v_{ii'} x_j^{ii'}$

Combing these sub-costs, $\vartheta_j^{ii'}$ can be calculated as:

$$\vartheta_j^{ii'} = \left\{ \left( \frac{r_j}{\hat{\gamma}_{i'}} (d_{i'j} \hat{\gamma}_{i'} + 1) + \theta_j (v_{ii'} + 1) + 1 \right) \right\} \tag{4}$$

### 3.4 Overall Formalization

The switch migration problem that handles the load balancing and minimizes the control plane overhead, can be now formulated as a mixed-integer linear programming as:

$$\text{Minimize } f \tag{5}$$

subject to:

$$0 \leq \sum_{c_i \in \mathcal{C}} \sum_{c_{i'} \in \mathcal{C}: c_i \neq c_{i'}} x_j^{ii'} \leq 1, \qquad \forall s_j \in \mathcal{S} \tag{6}$$

$$0 \leq l \leq \frac{\tilde{\omega}_i}{\gamma_i} \leq u < 1 \qquad \forall i \in \mathcal{C} \tag{7}$$

$$x_j^{ii'} \in \{0, 1\}, \qquad \forall c_i, c_{i'} \in \mathcal{C}, s_j \in \mathcal{S} \text{ and } l, u \in [0, 1) \tag{8}$$

The first constraint (6) ensures that when a switch is chosen for migration it can be only migrated to one controller. The controller capacity limitation and the upper and lower bounds, $l$ and $u$, are determined by the second constraint (7). The last constraint (8) defines the domains of the binary and real-valued variables.

## 4 Experimental Results

To evaluate the proposed model, a comparison against the static mapping model is performed. A random topology of 4 controllers and 16 switches is created (Fig. 2). The number of flow requests for each switch is generated randomly

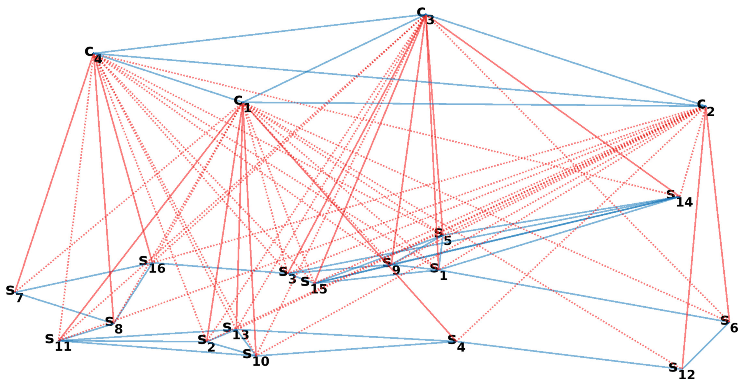



**Fig. 2.** A random SDN topology with 4 controllers and 16 switches used for simulation. Blue lines are intra-plane connections; red solid lines are the master assignments; and red dashed lines are the slave assignments. (Color figure online)

in the range [100, 400] `packet-in` messages per second, the latency is chosen from the range [0.1 1] milliseconds, and the simulation time is set to 1000 s. The weights in (3) are empirically set to $\alpha_1 = 0.7$ and $\alpha_2 = 0.3$. For the controllers capacity, two scenarios are considered: homogeneous and heterogeneous.

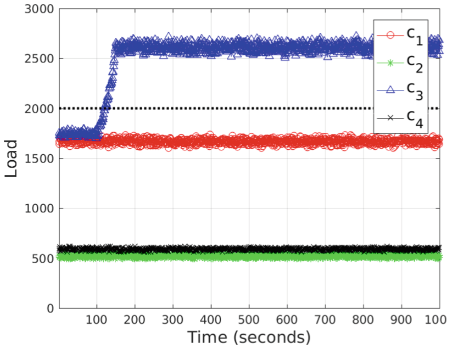
### 4.1 Homogeneous Controllers

In this scenario, the capacity $\gamma$ of each controller is set to 2000 `packet-in` messages per second. In order to create a realistic network fluctuation, a hot-spot is generated by stressing controller $c_3$, in Fig. 2, from time 100 to 150 s. The simulation results of this scenario are shown in Fig. 3.

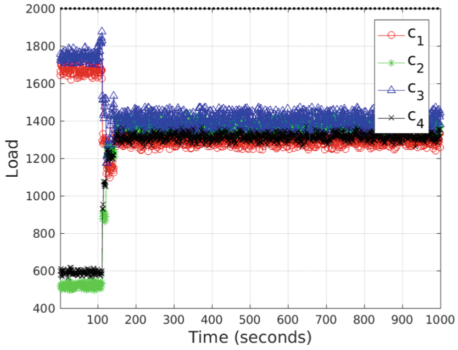
In Fig. 3a, the load of this controller increases until it eventually exceeds its limited capacity, which will cause in a real case scenario a failure in the domain managed by $c_3$. However, when applying the proposed switch migration model, in Fig. 3b, the loads of all controllers are shifted under the limited capacity, with remaining capacity $\hat{\gamma} \approx 500$ `packet-in` messages per second in all time slots after the stressing period (i.e., >150 s).

In order to assess the controllers' utilization, the min-max ratio is used. The range of this ratio is between 0 and 1. When it is 1, it means all controllers have the same load, i.e., a perfect utilization. On the other hand, values close to 0 mean very low utilizations. Figure 3c shows this measure for the static and proposed models. In average, the static model produced ≈21%, while the proposed model produced ≈84%, i.e., with enhancement of 63%. However, when calculating this measure after the stressing period, the static model produced ≈20%, while the proposed model produced ≈92%, with improvement of ≈72%.

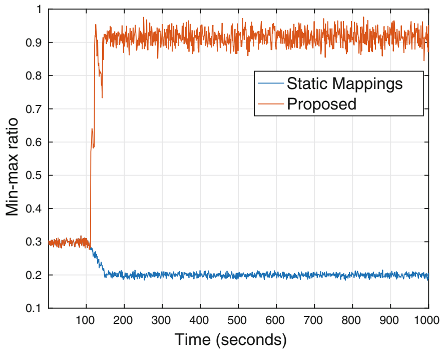
When considering the number of migrated switches in the proposed model, and as shown in Fig. 3d, before the stressing period (i.e., ≤100 s) no migration has occurred, and during the stressing period, only 9 switches were migrated



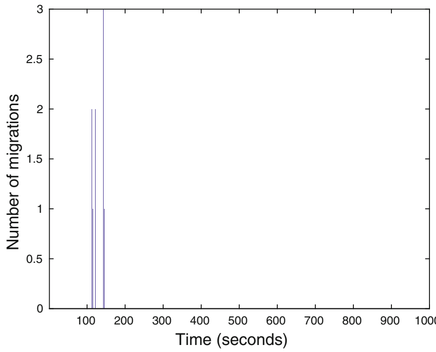

(a) Loads before switch migration




(b) Loads after switch migration




(c) Controllers' utilization




(d) Number of migrations

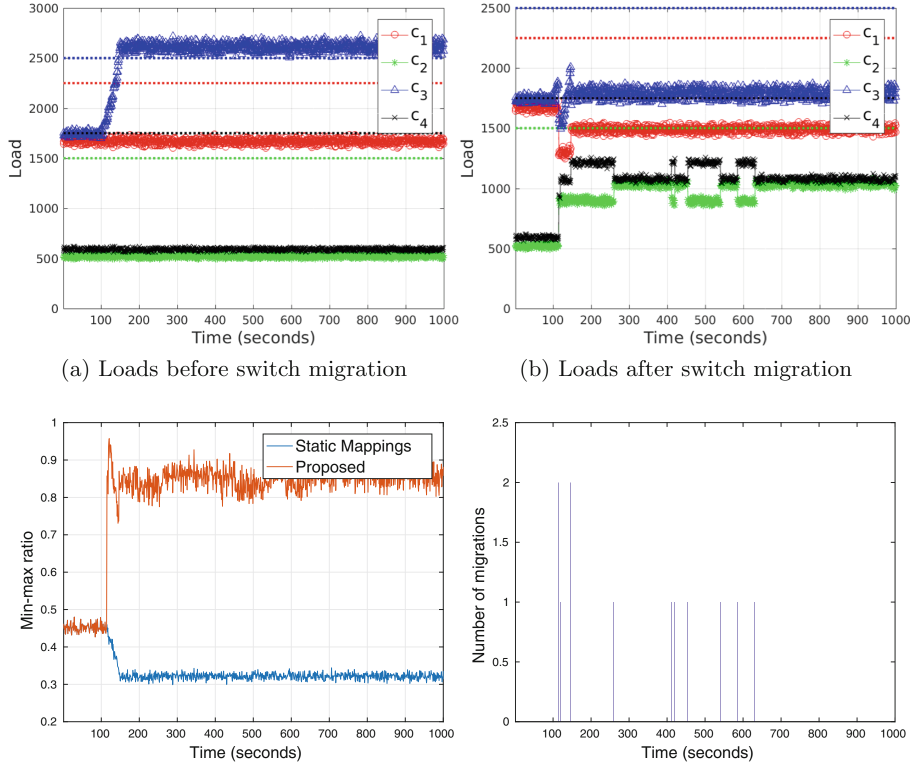
**Fig. 3.** Simulated results of 4 homogeneous controllers and 16 switches.

(with maximum of 3 switches at time 143 s) in order to cope with the traffic change. After the stressing period, no switch migration was required. Therefore the proposed model is able to maintain high utilization with low control plane overhead.

### 4.2 Heterogeneous Controllers

As shown in Fig. 4a, in this scenario the controllers have different capacities, but the total capacity is the same as in the previous test (i.e., 8000): $\gamma_1 = 2250$, $\gamma_2 = 1500$, $\gamma_3 = 2500$, and $\gamma_4 = 1750$. The simulation results of this scenario are shown in Fig. 4.

Comparing Figs. 4a and b, it is easy to see that the proposed switch migration model was successfully able to shift the loads of all controllers under their limited capacities, leading to a better controllers' utilization.




(a) Loads before switch migration

(b) Loads after switch migration

(c) Controllers' utilization

(d) Number of migrations

**Fig. 4.** Simulated results of 4 heterogeneous controllers and 16 switches.

When considering the min-max ratio in Fig. 4c. The static model produced in average ≈34%, while the proposed model produced ≈81%, with improvement of ≈47%. When considering only the time horizon after the stressing period, the static model in average produced ≈32%, and the proposed model produced ≈85%, with improvement of ≈53%.

Considering the number of migrated switches in Fig. 4d, no switch is migrated before the stressing period, 5 switches are migrated during the stressing period, with a maximum of 2 switches at two instances of time. After the stressing period, 7 (with maximum of 1 switch) switches are migrated, which might slightly increase the control plane overhead when compared to the homogeneous scenario. In fact, these 7 migrations has happened in a large time horizon (i.e., $1000-151 = 849$ time slots).

When comparing the results of the static model in both homogeneous and heterogeneous schemes, the latter produced better min-max ratio because controllers with small capacities can be easily utilized. The proposed model, however, produced better min-max ratio in the homogeneous scheme. In what concerns to the number of migrated switches, it is possible to experience slightly more switch migrations in the heterogeneous scenario.

## 5 Conclusions

This article has presented a model for solving the switch migration problem in software-defined networking. The problem is modeled such that it considers the load balancing of the controllers and minimizes the controller plane overhead created by the migrated switches. The model is formulated as a mixed integer linear programming, and the experimental results show that the proposed model can efficiently solve this problem for homogeneous and heterogeneous controllers. The results show that homogeneous controllers produce better utilizations and a slightly lower control plane overhead. As a future work, a robust and fast algorithm will be developed to solve the switch migration problem for large SDN networks.

## References

1. Bari, M.F., et al.: Dynamic controller provisioning in software defined networks. In: Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), pp. 18–25, October 2013. https://doi.org/10.1109/CNSM.2013.6727805
2. Benson, T., Akella, A., Maltz, D.A.: Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC 2010, pp. 267–280. ACM, New York (2010). https://doi.org/10.1145/1879141.1879175
3. Bizanis, N., Kuipers, F.A.: SDN and virtualization solutions for the Internet of Things: a survey. IEEE Access **4**, 5591–5606 (2016). https://doi.org/10.1109/ACCESS.2016.2607786
4. Casado, M., Freedman, M.J., Pettit, J., Luo, J., McKeown, N., Shenker, S.: Ethane: taking control of the enterprise. In: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2007, pp. 1–12. ACM, New York (2007). https://doi.org/10.1145/1282380.1282382
5. Chen, T., Matinmikko, M., Chen, X., Zhou, X., Ahokangas, P.: Software defined mobile networks: concept, survey, and research directions. IEEE Commun. Mag. **53**(11), 126–133 (2015). https://doi.org/10.1109/MCOM.2015.7321981
6. Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V., Kompella, R.: Towards an elastic distributed SDN controller. ACM SIGCOMM Comput. Commun. Rev. **43**(4), 7–12 (2013). https://doi.org/10.1145/2534169.2491193
7. Open Networking Foundation: OpenFlow Switch Specification. Technical report (2011)
8. Gao, X., Kong, L., Li, W., Liang, W., Chen, Y., Chen, G.: Traffic load balancing schemes for devolved controllers in mega data centers. IEEE Trans. Parallel Distrib. Syst. **28**(2), 572–585 (2017). https://doi.org/10.1109/tpds.2016.2579622
9. Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T.: Software-defined networking: challenges and research opportunities for future internet. Comput. Netw. **75**(A), 453–471 (2014). https://doi.org/10.1016/j.comnet.2014.10.015
10. Karakus, M., Durresi, A.: A survey: control plane scalability issues and approaches in Software-Defined Networking (SDN). Comput. Netw. **112**, 279–293 (2017). https://doi.org/10.1016/j.comnet.2016.11.017

11. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015). https://doi.org/10.1109/JPROC.2014.2371999
12. Liang, C., Kawashima, R., Matsuo, H.: Scalable and crash-tolerant load balancing based on switch migration for multiple OpenFlow controllers. In: 2014 Second International Symposium on Computing and Networking (CANDAR), Shizuoka, People's Republic of China, 10–12 December 2014, pp. 171–177 (2014). https://doi.org/10.1109/CANDAR.2014.108
13. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008). https://doi.org/10.1145/1355734.1355746
14. Sridharan, V., Gurusamy, M., Truong-Huu, T.: On multiple controller mapping in software defined networks with resilience constraints. IEEE Commun. Lett. **21**(8), 1763–1766 (2017). https://doi.org/10.1109/LCOMM.2017.2696006
15. Wang, C., Hu, B., Chen, S., Li, D., Liu, B.: A switch migration-based decision-making scheme for balancing load in SDN. IEEE Access **5**, 4537–4544 (2017). https://doi.org/10.1109/ACCESS.2017.2684188
16. Wang, T., Liu, F., Xu, H.: An efficient online algorithm for dynamic SDN controller assignment in data center networks. IEEE/ACM Trans. Netw. **25**(5), 2788–2801 (2017). https://doi.org/10.1109/TNET.2017.2711641
17. Wu, P., Yao, L., Lin, C., Wu, G., Obaidat, M.S.: FMD: a DoS mitigation scheme based on flow migration in software-defined networking. Int. J. Commun. Syst. **31**(9), e3543 (2018). https://doi.org/10.1002/dac.3543
18. Yao, G., Bi, J., Guo, L.: On the cascading failures of multi-controllers in software defined networks. In: Fu, X., Hilt, V., Wolf, T., Zhang, L., Zhang, Z.L. (eds.) 2013 21st IEEE International Conference on Network Protocols (ICNP). IEEE International Conference on Network Protocols Proceedings, Gottingen, Germany, 7–10 October 2013. IEEE; IEEE Computer Society; VDE, Information Technology Society; Gesell Informatik (2013)
19. Yao, G., Bi, J., Li, Y., Guo, L.: On the capacitated controller placement problem in software defined networks. IEEE Commun. Lett. **18**(8), 1339–1342 (2014). https://doi.org/10.1109/LCOMM.2014.2332341
20. Ye, X., Cheng, G., Luo, X.: Maximizing SDN control resource utilization via switch migration. Comput. Netw. **126**, 69–80 (2017). https://doi.org/10.1016/j.comnet.2017.06.022
21. Yeganeh, S.H., Tootoonchian, A., Ganjali, Y.: On scalability of software-defined networking. IEEE Commun. Mag. **51**(2), 136–141 (2013)
22. Zhang, Y., Cui, L., Wang, W., Zhang, Y.: A survey on software defined networking with multiple controllers. J. Netw. Comput. Appl. **103**, 101–118 (2018). https://doi.org/10.1016/j.jnca.2017.11.015
23. Zhou, Y., Zheng, K., Ni, W., Liu, R.P.: Elastic switch migration for control plane load balancing in SDN. IEEE Access **6**, 3909–3919 (2018). https://doi.org/10.1109/ACCESS.2018.2795576