This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

A Switch Migration-based Decision-making Scheme for Balancing Load in SDN

Chuan'an Wang[1, 3], Bo Hu[1*], Shanzhi Chen[2], Desheng Li[3], Bin Liu[3]

[1]State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100081, China
[2]State Key Lab of Wireless Mobile Communication, China Academy of Telecommunication Technology, Beijing 100081, China
[3]School of Information and Network Engineering, Anhui Science and Technology University, Chuzhou 233100, China

*Corresponding author: B. Hu (E-mail: hubo@bupt.edu.cn)

This work was supported in part by the National Science and Technology Major Projects for the New Generation of Broadband Wireless Communication Networks under Grant 2016ZX03001017, the National Natural Science Foundation of China for Distinguished Young Scholars under Grant 61425012, the National High-Tech R&D (863) Program under Grant 2014AA01A701 & 2015AA01A705, and the Key Project of Natural Science Research of Universities in Anhui under Grant KJ2016A176&KJ2015A236.

***Abstract*—Elastic scaling and load balancing with efficient switch migration are critical to enable the elasticity of software defined networking (SDN) controllers, but learning how to improve migration efficiency remains a difficult problem. To address this issue, a SMDM (switch migration-based decision-making) scheme is put forward that could be made aware of the load imbalance by a switch migration trigger metric; the migration efficiency model for this scheme is built to make a tradeoff between migration costs and the load balance rate. An efficiency-aware switch migration algorithm based on greedy method is designed to utilize the migration efficiency model and thus guide the choice of possible migration actions. We implement a proof of the scheme and present a numerical evaluation using Mininet emulator to demonstrate the effectiveness of our proposal.**

***Index Terms*—Software defined networking, Switch migration, Migration efficiency, Migration cost, Load balancing**

I. INTRODUCTION

As an emerging technology, SDN makes it easy to manage networks and enable innovation and evolution by decoupling the control plane from the data plane. The intelligence of SDN is shown by the fact that a logically centralized controller manages switches by providing them with rules that can dictate their packet handling behavior [1].With the continuous extension of network scale, the scalability of the centralized controller becomes a key issue in SDN [2]. Deploying distributed controllers is a promising approach to solve the problem, and each controller manages part of the switches in the network. However, static switch-controller mapping results in load imbalances and sub-optimal performance in cases of uneven load distribution among controllers [3].

Dynamic switch migration is a promising approach to elastic scaling and load balancing. In practice, switch migration occurs in three cases. Firstly, if the aggregated traffic load goes beyond the capacity of all controllers, the new controllers should be added and the switches would be moved to them. Secondly, as a controller is shut down or to sleep for saving communication cost and power, its switches should be migrated away. Thirdly, even if there is no change in the number of deployed controllers, switch migration operation must be performed by migrating selected switch to other controllers when an individual controller load is beyond its capacity. We call this operation as load balancing.

With live switch migration, the performance and scalability in distributed controllers may be effectively increased. However, such migration has to be performed with a well-designed mechanism to decide which switch and where it should be migrated, and we define it as switch migration problem (SMP). In solving SMP, most existing studies only consider utilizing the available controllers without taking into account migration efficiency. As an example, controller prefers to migrate a switch to a new master controller with more efficiency for eliminating overload. In fact, seldom researches have been made on considering it, i.e., in [3], the authors design a synthesizing distributed algorithm for SMP, but the switch is randomly selected for migrating. In this paper, we focus on the third switch migration case.

To address the impact of migration efficiency on migration cost in the context of switch migration, we propose an SMDM scheme. In SMDM scheme, our primary objective is how to elect an efficient controller as a master controller to improve load balance factor, as well as which switch with low migration cost should be selected for migrating. The scheme focuses on solving the following three sub-problems of switch migration:

- How to measure the load imbalance of controllers and decide whether to perform switch migration.
- How to make a tradeoff between migration costs and load balance rate.
- How to employ a migration plan that utilizes the migration efficiency model to guide the choice of possible migration actions.

The main contributions of our work compared to related works are as follows.

- We use the aggregate load value to indicate the real load information and provide the switch migration-triggered metric.
- We build a migration efficiency model to make a tradeoff between migration costs and the load balance variation.
- On the basis of the optimal migration efficiency conditions, the migration plan is formulated as a set of the migration actions, and SMDM algorithm is designed by the greedy method.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

The rest of the paper is organized as follows. Section II provides an overview of related works. An SMDM scheme is proposed in Section III. We design SMDM algorithm and implement it in Section IV. We describe the evaluation setting and discuss the performance in Section V. We finally present our conclusions in Section VI.

II. RELATED WORKS

Traditional SDN implementation relies on a centralized controller and has several limitations related to performance and scalability [2]. Some research works have proposed that deploying distributed controllers is a promising approach to solving the problem [4-6].To achieve more performance and scalability in distributed controllers, MF Bari et al. [7] provide a framework that adjusts the number of active controllers and delegates each controller. This framework could minimize flow setup time while incurring very low communication overhead. But it easily leads to network instability because it has to perform a reassignment of the entire control plane based on the collected traffic statistics.

To improve the performance of the control plane in SDN, some works advance the idea that multifaceted aspects should be taken into consideration, e.g., maximize the performance of each physical controller [8-10], offload the controller by delegating some work to the forwarding devices [11,12] and enable a cluster of controller nodes to achieve a distributed control plane [13,14]. These works have proposed a logically centralized control plane and try to address the global view and states consistency of distributed control plane, which could achieve better scalability and reliability with separate controllers, but this approach would lead to load imbalance among controllers when an uneven huge traffic load arrive these distributed controller.

In addition, to enable a scalable SDN control plane, G Cheng et al. [15] provided a game decision mechanism to dynamically migrate switches from heavy load controllers to light controllers whereby switch migration decisions are formulated as a centralized available resource utilization maximization problem with constraints on multiple resource dimensions. In [16], a framework that automatically adjusts the switch to controller connectivity is proposed, however it only considers the flow setup latency involved in setting up paths from controllers to switches without considering the increase in migration cost.

Similarly, some research on controllers' load balancing has been done. C Liang et al. [17] propose a dynamic load balancing method based on switch migration mechanism for clustered controllers. The proposed method can dynamically shift the load across the multiple controllers through switch migration, but it needs to cluster the controllers before performing a switch migration, and which leads to the increase of response time. A distributed nearest migration algorithm (DNMA) for load balancing is proposed [18]. In DNMA method, to save migration time, the nearest neighbor controller is selected as the immigration controller for receiving load shifting, however it maybe bring about new load imbalance without considering the nearest neighbor's load. In [3], a maximizing resource utilization migration algorithm (MUMA) is designed. When load imbalance occurs, the controller randomly selects a switch for migrating, and coming migration activity is broadcast to the controller's neighbors. However, after migration, controllers need to state the synchronization for the global network view.

III. SWITCH MIGRATION-BASED DECISION-MAKING SCHEME

In this section, we propose an SMDM scheme for making switch migration decision whereby the migration efficiency model is built to make a tradeoff between migration costs and the load balance rate. We consider that the scheme can create a migration plan to utilize the migration efficiency model and thus guide the choice of possible migration actions. The scheme is described in three phases. First, we measure the load diversity of different controllers and decide whether to perform migration. Then, we predict the migration cost and migration efficiency, which can be used to guide the choice of possible migration actions. Finally, we provide a switch migration plan, which is also one of the core procedures for performing load balancing across controllers.

A. LOAD BALANCE DETECTION

In this work, we consider a SDN network G that consists of N controllers $C=\{c_1,c_2,\cdots,c_N\}$ and K switches $S=\{s_1,s_2,\cdots,s_K\}$. Let $L_{c_i}\in S$ denote the switches set managed by controller c_i.

As the literature [19, 20] indicates, the processing of PACKET_IN events is generally regarded as the most prominent part of the controller load. When a switch receives a new flow, it requests the controller to calculate the flow path and install appropriate rules. Thus, the cost of completing the operation should also be considered. Here, we consider two major types of load information for computing aggregate load and use the aggregate load value to indicate the real load information for controllers. The load γ_{c_i} for each controller c_i is calculated using the following equation:

$$\gamma_{c_i}=\sum_{s_k\in Lc_i} f_{s_k}\cdot d_{s_k c_i} \tag{1}$$

where f_{s_k} represents the number of Packet_in messages sent from switch s_k to controller c_i and $d_{s_k c_i}$ is the minimal path cost from switch s_k to controller c_i .

Hence, the load diversity between controller c_i and controller c_j is:

$$h_{c_i c_j}=\frac{\gamma_{c_i}}{\gamma_{c_j}} \tag{2}$$

Derived from above, the controller load diversity matrix $H_{K\times K}$ can be represented as:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

$$H_{K\times K}=\begin{Bmatrix} 1 & h_{c_1c_2} & h_{c_1c_3} & \cdots & h_{c_1c_{k-1}} & h_{c_1c_k} \\ h_{c_2c_1} & 1 & h_{c2c_3} & \cdots & h_{c_2c_{k-1}} & h_{c_2c_k} \\ & & \vdots & & & \\ h_{c_{k-1}c_1} & h_{c_{k-1}c_2} & h_{c_{k-1}c_3} & \cdots & 1 & h_{c_{k-1}c_k} \\ h_{c_kc_1} & h_{c_kc_2} & h_{c_kc_3} & \cdots & h_{c_kc_{k-1}} & 1 \end{Bmatrix} \quad (3)$$

Given the threshold σ of load diversity, the switches migration-triggered metric is:

$$\exists h_{c_ic_j}>\sigma,\ \ i,j=1,2,\cdots,K \quad (4)$$

From (4), it can be seen that if the load diversity between any two controllers goes beyond the predetermined threshold σ , switch migration is performed. In particular, switch migration occurs without load diversity detection when a controller is added to or moved from the deployed controllers set C, and this case is referred to as an update to the set of deployed controllers.

To effectively identify migrated switches and reduce the complexity of migration, we check the load diversity matrix and decide which controllers should be selected as the set OM_S of outmigration controllers, and which should be selected as the set IM_S of immigration controllers1. If the load diversity $h_{ij}>\sigma$, the controller ci and cj are added into OM_S and IM_S, respectively. Note that a transit controller ct, which is responsible for receiving load shifts and then sending them to the destination controller, shoud be added to IM_S when the transmission path between source controller and destination controller is too long.

B. MIGRATION EFFICIENCY MODEL

An appropriate migration cost and efficiency model has a guiding significance for deploying the migration scheme and migration optimization. We assume that the network G global view is taken as common knowledge by different controllers. When switch migration occurs, the load balance of network G improves, but it also brings additional migration costs. For convenience, we give the concept of migration cost.

Definition 1: The **migration cost** is mainly formed by two components: (1) the increase in load cost; (2) the message exchanging cost. When a switch s_k is migrated from controller c_i to controller c_j, the migration cost $r_{s_kc_j}$ can be defined as follows:

$$r_{s_kc_j}=r_{mc}+r_{lc} \quad (5)$$

where r_{mc} is the cost of message exchanging for switch s_k migration and r_{lc} is the increase in the load cost, which is defined by the following equation:

$$r_{lc}=\begin{cases} f_{s_k}d_{s_kc_j}-f_{s_k}d_{s_kc_i}, & f_{s_k}d_{s_kc_j}>f_{s_k}d_{s_kc_i} \\ 0, & f_{s_k}d_{s_kc_j}\le f_{s_k}d_{s_kc_i} \end{cases} \quad (6)$$

We use the controllers' load variance as a load balance factor and let $\bar{\gamma}$ be the average load of N controllers. Then, the load balance factor of network G before migrating switch s_k is:

$$\Gamma=\frac{1}{K}\sum_{i=1}^{k}(\gamma_{c_i}-\bar{\gamma})^2 \quad (7)$$

After switch s_k migration, the new load balance factor becomes:

$$\Gamma^*=\frac{1}{K}\Big(\sum_{i=1,i\neq j}^{k}(\gamma_{c_i}^*-\bar{\gamma}^*)^2+(\gamma_{c_j}^*-\bar{\gamma}^*)^2\Big) \quad (8)$$

where $\gamma_{c_i}^*=\gamma_{c_i}-f_{s_k}d_{s_kc_i}$ and $\gamma_{c_j}^*=\gamma_{c_j}+f_{s_k}d_{s_kc_j}$ are renewal γ_{c_i} and γ_{c_j}, respectively. $\bar{\gamma}^*$ is the renewal average load of N controllers .

To make a tradeoff between migration cost and the load balance rate, we give the definition of migration efficiency.

Definition 2: The **migration efficiency** of moving switch s_k to controller c_j can be defined as the ratio of load balance variation to migration cost:

$$\tau_{s_kc_j}=\left|\Gamma^*-\Gamma\right|/\gamma_{s_kc_j} \quad (9)$$

Therefore, the purpose of the migration efficiency model is to select a controller as the destination controller for receiving load shifting based on the principle of migration efficiency maximization.

C. MIGRATION STRATEGY

The purpose of switch migration strategy is to migrate some switches from outmigration controllers to immigration controllers with higher migration efficiency. We formulate the switch migration operation as a series of migration actions and present a triplet <c_u,s_e,c_v> definition of migration action, where $c_u\in OM\_S$ is a outmigration controller and s_e is the switch managed by c_u for being migrated, and the $c_v\in IM\_S$ is an immigration controller . When a controller c_u is selected, our focus is which switch should be selected as s_e and how to elect the controller c_v from *IM_S*.

Switch s_e selection. Controller c_u chooses switch s_e based on the following considerations. First, controller c_u prefers to migrate a switch with less load and more efficiency for eliminating overload; then, it expects to select switch s_e, which could lead to a smaller load difference in γ_u and $\bar{\gamma}$ to migrate.

[1] Immigration controller is the destination controller for receiving load data shifting from migrated switches,and is seen as new master controller.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

Second, controller c_u expects switch s_k, which has a large latency to it. Thus, the probability that controller c_u is:

$$p_{s_e} = 1 - \frac{\left| \bar{\gamma} - \gamma_{L_{c_u} - \{s_e\}} \right| e^{d_{s_e c_u}}}{\sum_{s_k \in L c_u} (\gamma_{c_u} - \gamma) e^{d_{s_k c_u}}} \quad (10)$$

where $\gamma_{L_{c_u}-\{s_e\}}$ is the load of controller c_u after switch s_e is migrated from it.

Immigration controller c_v selection. When switch s_e is migrated to a controller c_i , if the sum of γ_{c_i} and γ_{s_e} is not beyond the load capacity of controller c_i ,the controller c_i will be added into temporary controller set *T_S*. Any controller c_i in *T_S* will calculate the migration efficiency $\tau_{s_e c_i}$. Guided by the concept of a migration efficiency model, a controller c_i is selected as the immigration controller c_v based on the following equation.

$$c_v = \arg\max_{c_i \in T\_S} \{\tau_{s_e c_i}\} \quad (11)$$

IV. SMDM ALGORITHM DESIGN AND IMPLEMENTATION

A. SMDM ALGORITHM

The SMP is an NP-hard bin packing problem [21]. On the basis of the optimal migration efficiency conditions, the SMDM algorithm is designed by the greedy method, and algorithm consists of two phases: load balancing detection and migration actions generation. We briefly describe the SMDM algorithm as follows.

Phase 1: Load imbalance detection

Assuming that any controller has gained its load information. Controller c_i calculate aggregated load γ_{c_i} and load diversity $h_{c_i c_j}$, and then create load diversity matrix $H_{K\times K}$. If an element $h_{c_i c_j}$ in $H_{K\times K}$ is beyond the threshold, the controller c_i and c_j is added into outmigration controllers set *IM_S* and inmigration controllers set *IM_S* respectively, and the switch migration operation is triggered at the same time.

Phase 2: Switch migration

In this phase, the switch migration operation is formulated as a series of migration actions presented by a triplet <c_u,s_e,c_v>. The objective of switch migration operation is to determine switch s_e selection and controller c_v selection. Steps of Phase 2 are elaborated as follows:

Step 1 Each controller c_u in *OM_S* chooses a switch s_e for migration based on a special selector defined in the formulation (10).

Step 2 Any controller c_i in *IM_S*, if $\gamma_{c_i} + \gamma_{s_e} \le \psi_{c_i}$ then add c_i to temporary controller set *T_S*, and calculate the migration efficiency $\tau_{s_e c_i}$. Let ψ_{c_u} be the load capacity of controller c_u.

Step 3 controller c_u is select as immigration controller c_v based on formulation (11).Add <c_u,s_e,c_v> to migration actions set *P* and update the state of c_u and c_v.

The pseudocode of SMDM is shown as following which runs on each individual controller independently.

SMDM algorithm

Phase 2: Load detection

1: Scanning for the load of controllers
2: if (Switch migration trigger) then
3: add c_i to *OM_S* , c_i to *IM_S*
4: do switch migration ( )
5: end if

Phase 2: Switch Migration

1: initialize migration actions set *P*={ };
temporary controller set *T_S* ={}
2: let ψ_{c_u} be the load capacity of controller c_u
3: repeat
4: for each controller c_u in *OM_S* do
5: calculate the migration probability p_{s_k} of switches managed by c_u
6: $s_e = \arg\max_{s_k \in L_{c_u}} \{p_{s_k}\}$
7: for each controller c_i in *IM_S*
8: if $\gamma_{c_i} + \gamma_{s_e} \le \psi_{c_i}$ then
9: calculate the migration efficiency $\tau_{s_e c_i}$
10: end if
11: end for
12: $c_v = \arg\max_{c_i \in T\_S} \{\tau_{s_e c_i}\}$
13: add <c_u,s_e,c_v> to *P*
14: update the state of c_u and c_v
15: end for
26: until (load diversity $\forall h_{c_i c_j} < \sigma$)

B. SMDM IMPLEMENTATION FOR LOAD BALANCING

We describe a load balancing framework based on SMDM as illustrated in Figure 1. We consider the framework that dynamically balances the load distribution among the controllers and optimizes migration efficiency. It has five core modules: the monitoring module, load balance detection module, migration efficiency calculation module, migration strategy decision module and migration execution module.

Monitoring module tracks real-time load information, calculates the aggregate load value for each controller, and provides the load data to the load balance detection module. However, if the deployed controllers set C is updated, the monitoring module delivers a signal to the migration strategy decision module to perform switch migration without executing the load balance detection module.

Load balance detection module measures the load diversity of different controllers and decides whether to perform switch migration. In this paper, the switch migration can be triggered when the load diversity meets equation (4).

Migration efficiency calculation module builds the migration efficiency model, which is used to make a tradeoff between the migration efficiency and migration cost caused by migrating switches. It then provides these migration factors for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

the migration strategy decision module to make a migration plan.

Migration strategy decision module is responsible for creating the migration plan. In this paper, the migration plan is formulated as set *P* of the migration actions, and each migration action is a triplet $<c_u,s_e,c_v>$. We implement our SMDM algorithm in this module. It utilizes those migration factors from the migration efficiency calculation module to generate corresponding migration actions.

Migration execution module is placed in each controller. Its function is to coordinate the migration actions for switch migration and change the switch-controller mapping.

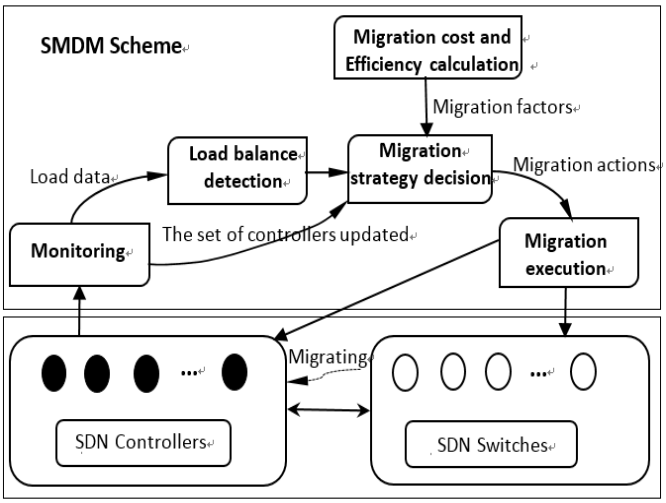


Fig. 1. Load balancing framework based on SMDM

We uses distributed storage mechanism to store load information, which could provide a logically central view for all controllers. Each controller runs such an SMDM framework instance, and we install Beacon controller [23] in an individual machine to simulate single centralized controller. Then we implement load balancing detection module and migration strategy module in Beacon controller, and we invoke IBeacon Provider to interact with Open Flow switches.

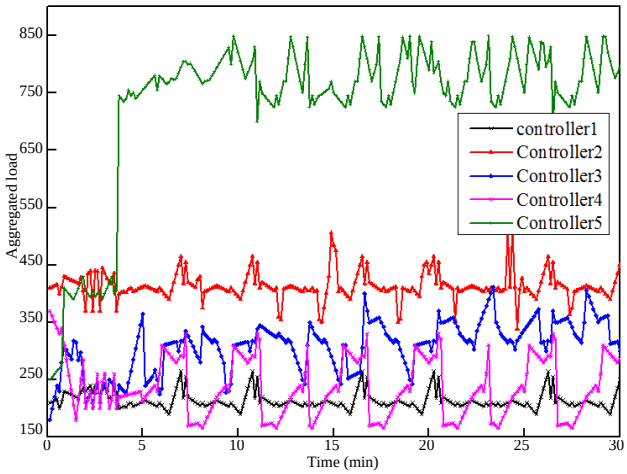
V. EVALUATION

To assess the performance of the SMDM approach, we deploy Mininet, Beacon controller, and our SMDM on multiple physical machines to emulate the testbed, and each physical machines runs Ubuntu 16.04 LTS with JDK8. We modified Mininet to enable us to run the software-based virtual Open Flow switch instances on different machines. We use the real internet service topologies BT North America (36 nodes and 76 links) from Internet Topology Zoo [22]. Note that we are primarily concerned with the controller load and need not emulate the high overhead data plane or the actual transmission of packets through switches. Therefore, one server runs Mininet, and the other five servers run SDN controller instances. The six servers have the same configuration with 4GHz Intel Core i7 processor and 16 GB of DDR3 memory.

In this paper, special focus lies on quantifying the trade-off between the load balance rate when using the switch migration approach and the migration cost in terms of accuracy that is entailed. For this purpose, we first compare the average response time of our SMDM method with two other switch migration methods, DNMA [18] and MUMA [3]. Then, we evaluate the cost from switch migration. Finally, we measure the average migration operation execution time by varying the number of migration switches.

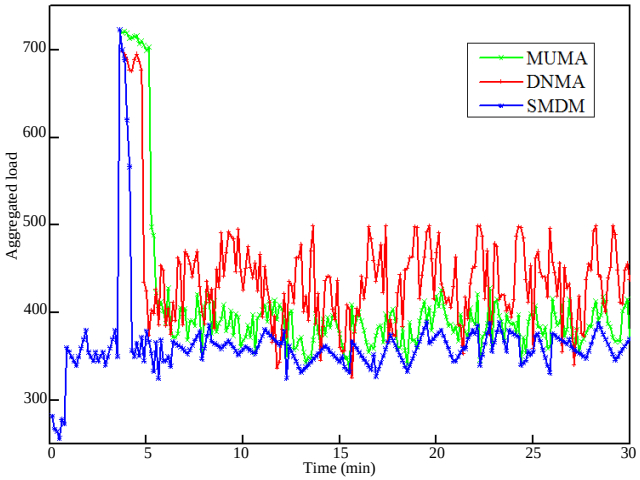
In MUMA method ,a synthesizing distributed algorithm hopping is designed for switch migration problem (SMP) by approximating the optimal objective via Log-Sum-utility function.When load imbalance occurs,the controller randomly selects a switch for migrating,and coming migration activity is broadcast to the controller's neighbors. After migration, controllers need to state the synchronization for the global network view.While in DNMA method,to save migration time, the nearest neighbor controller is selected as the immigration controller for receiving load shifting, however,it maybe bring about new load imbalance without considering the nearest neighbor's load.

Load balancing. We run each simulation for 30 minutes. Figure 2 shows the controllers' load distribution under two scenarios: one during the SMDM approach and one under static switch-controller mapping. To simulate the uneven load distribution among the controllers, we use aggregated load to stress the controllers by adjusting the different sending rates of generated Packet-In messages from the switches. From figure 2a, we can see that controller5's load becomes heavy at about 275 sec, and that continues to be the case because of the lack of intervention.Controller5's load tends to balance again at about 300 sec in Figure 2b, which is affected by the SMDM method under migrating some switches from controller5 to light load controllers (e.g, controller1 and controller4).When we use the SMDM and DNMA, the Controller5's load tends to balance also at about 310 sec and 325 sec, respectively. From the result of figure 2b, we can observe that the SMDM has the least time on load balancing.



(a) under static switch-controller mapping

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access



（b）controller5's load distribution under the three approaches

Fig. 2. Measured controllers' load distribution

Response time. We know that if load imbalance occurs, response time will shot up.We use average response time to measure the effect of the three switch migration methods in the simulation. Figure 3 shows their response time curves. We observe that the response time and migration execution time of SMDM are lower than MUMA and DNMA. Compared with MUMA, DNMA has shorter migration execution time, while it has higher response time. There are three reasons to explain this result. First, to save migration time, MUMA randomly selects switches for migrating and performs reassignment between controllers and switches when the load becomes imbalanced, while SMDM only selects the switches with highly probabilities for migrating based on given formulation (10). Second, SMDM selects efficient controllers as immigration controllers for receiving load shifting by switches migration, which could improve the migration efficiency and avoid new potential load imbalance. Third, Although DNMA has a distributed control plane, its nearest neighbor migration method could easily lead to new load imbalance, so DNMA suffers from highest response time.

The cumulative distribution function (CDF) of response time of each drop is depicted in Figure 4.We can see that our proposed SMDM is less vulnerable to the number increase of switches than MUMA and DNMA.

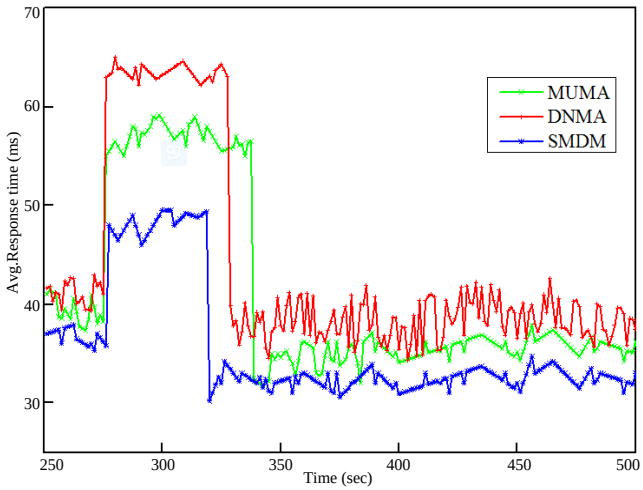


Fig. 3. Response time

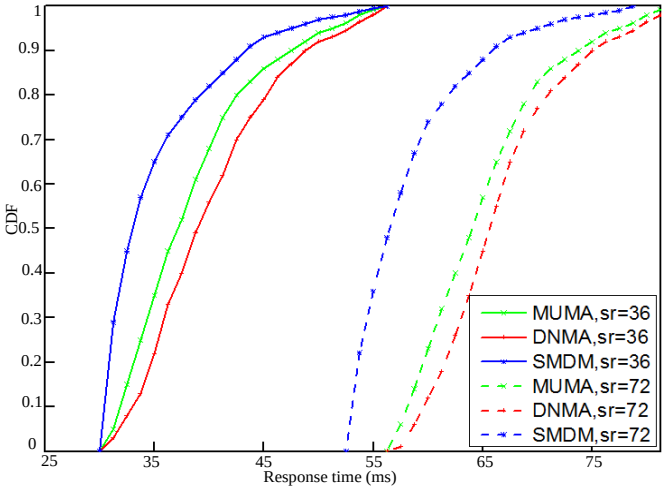


Fig.4. Response time CDF of three methods with different numbers of switches sr

Migration cost. We use migration cost and average response time to quantify the migration efficiency and analyze the performance of SMDM, DNMA and MUMA. Figure 5 shows the migration cost and average response time. We can see that the migration cost of SMDM is somewhat higher than that of DNMA, while the average response time of SMDM is significantly lower than that of DNMA. MUMA has the highest migration cost, and the average response time is also higher than that of SMDM. Two reasons explain the above results. Compared with SMDM, DNMA has the lowest migration cost due to its neighbor node selection strategy and lower rate of message exchange during the switch migration process. However, it also creates a very high response time because of the new load imbalance. In addition, to achieve a better load balance, MUMA most likely migrates a switch multiple times, so it has the highest migration cost, while SMDM only migrates several switches from the *OM_S* to *IM_S* via our migration strategy decision.

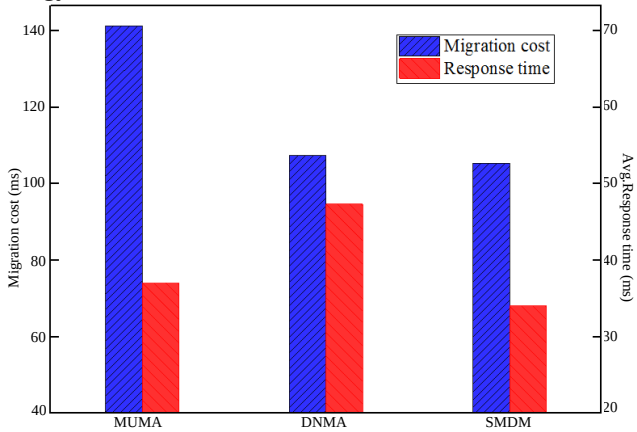


Fig. 5. Migration cost and average response time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2684188, IEEE Access

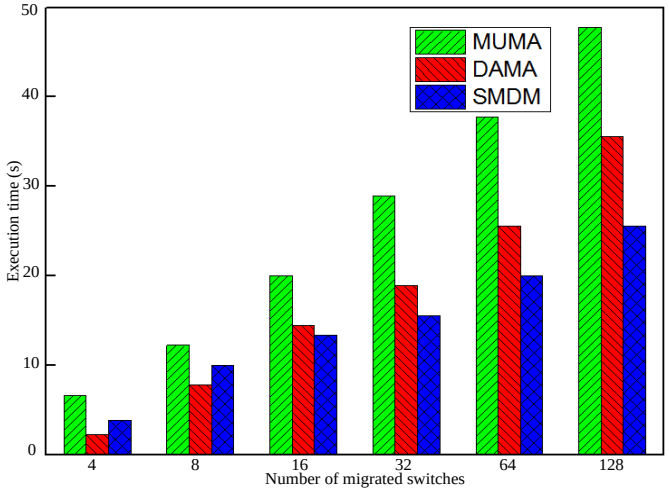


Fig. 6. Migration execution time

Migration execution time. Furthermore, we built a wireless network topology in which we measure the average execution time by varying the number of migration switches from 4 to 128. Figure 6 shows the trend in average migration execution time variation of MUMA, DNMA and SMDM. We observe that the average migration execution time of MUMA increases fastest with the increase in the number of migrated switches because controllers need to state the synchronization for the global network view after migration, while the average migration execution time of SMDM and DNMA exhibit a moderate increase. Compared with the migration execution time, we can conclude that SMDM will evidently not increase response time.

Based on the above results, our method enables the elasticity of SDN controllers via switch migration and can improve migration efficiency.

VI. CONCLUSION

In this paper, the primary objective is to make an efficiency switch migration scheme for load balancing in SDN Controllers. To this end, we first check the real-time controller load information collected by the monitoring module and decide whether to perform switch migration. Then, we built the migration efficiency model to tradeoff between the migration cost and the load balance rate. Finally, an efficiency-aware migration algorithm based on greedy method was designed to utilize the migration efficiency model and thus guide the choice of possible migration actions. In future work, we plan to implement the SMDM in a real large-scale wireless access network with more real-world traffic as well as evaluate the performance.

REFERENCES

[1] B. Lantz, B. Heller, and N. McKeown,"A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp.19-23.

[2] Dixit, A Hao, F Mukherjee, S Lakshman, T Kompella, et al. "Towards an Elastic Distributed SDN Controller," In *proceedings of HotSDN, ACM Press*, 2013, pp.7-12.

[3] G Cheng，H Chen，Z Wang，S Chen, "DHA: Distributed Decisions on the Switch Migration Toward a Scalable SDN Control Plane,",in *Ifip Networking Conference*, 2015.

[4] T. Koponen et al., "Onix: A Distributed Control Platform for Large-scale Production Networks," in *OSDI*, 2010, pp.8-15.

[5] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *proceedings of 2010 Internet Network Management Workshop / Workshop on Research on Enterprise Networking (INM/WREN)*, 2010, pp.3-3.

[6] 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*, 2012, pp.19-24.

[7] Md. Faizul Bari,Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *international Conference on Network and Service Management (CNSM)*,2013, pp.18-25.

[8] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood,"On controller performance in software-defined networks," in *USENIXWorkshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2012,pp.19-25.

[9] D. Erickson, "The beacon openflow controller," *In proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking*, 2013, pp. 13–18.

[10] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high performance networks," in *ACM SIGCOMM Computer Communication Review*, 2011, pp. 254–265.

[11] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks", *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4-17, Jan,2015

[12] J. C. Mogul and P. Congdon, "Hey, you darned counters!: get off my asic!" in *proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 25–30.

[13] 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*, 2010, pp. 3–3.

[14] 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*, 2012, pp.19–24.

[15] G Cheng，H Chen，H Hu，J Lan,"Dynamic switch migration towards a scalable SDN control plane," *International Journal of Communication Systems*, Vol 29,no.9,pp.1482-1499, June,2016.

[16] M. F. Bari et al., "Dynamic controller provisioning in software defined networks," in *proceedings of 9th Int. CNSM*, 2013, pp. 18–25.

[17] C Liang, R Kawashima, H Matsuo, "Scalable and Crash-Tolerant Load Balancing Based on Switch Migration for Multiple Open Flow Controllers," International Symposium on Computing & Networking, 2014, pp.171-177.

[18] Dixit A, Hao F, Mukherjee S, et al. Towards an elastic distributed sdn controller", in *Proc of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*,2013,pp.7-12.

[19] Tootoonchian A Gorbunov S and Ganjali Y, et al., "On Controller Performance in Software-de ned Networks", in *proceedings of HotICE*, 2012, pp.52-57.

[20] Yao G, Bi J, Li Y, Guo L,"On the capacitated controller placement problem in software de ned networks," *IEEE Communications Letters*, vol.18, no.8, pp.1339-1342 , August 2014.

[21] XL Qin, WB Zhang, W Wang, J Wei,et al., "Enabling Elasticity of Key-Value Stores in the Cloud Using Cost-Aware Live Data Migration," Journal of Software,vol.24,no.6,pp.1403-1417,June,2014 .

[22] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*,vol. 29,no.9,pp.1765- 1775, Oct. 2011.

[23] Erickson D, "The Beacon Open Flow Controller", In proceeding of 1st Workshop on Hot Topics in Software Defined Networking, 2013,pp.13–18.