Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 8, August 2018, pp. 1848-1856
DOI: 10.23940/ijpe.18.08.p22.18481856

Sequenced Switch Migration for Balancing Controller Loads in Large-Scale Software-Defined Networking

Lan Yao, Tao Hu*, and Julong Lan

National Digital Switching System Engineering R&D Center, Zhengzhou, 450002, China

Abstract

The proposal of the multi-controller has improved the scalability and reliability of software-defined networking (SDN), and the entire network is divided into several subdomains with the self-governed controller. Due to the dynamic changes of network traffic in different subdomains, a new challenge has emerged for balancing loads on the distributed controllers. Switch migration, a promising method, is designed to solve the unbalanced distribution of controller loads. However, existing schemes implement switch migration without considering migration sequences and are characterized by long migration time and poor controller processing efficiency. To solve the above problems, this paper proposes the Sequenced Switch Migration (SSM) approach, which considers migration parameters (e.g., delay, traffic, residual capacity, and failure probability) for multi-objective optimization and achieves efficient switch migration using a sequenced manner. By executing the parallel and heuristic algorithms, SSM can reduce the execution time of migration while ensuring the performance requirements of controllers. The simulation results show that compared with the existing schemes, SSM reduces the total migration time by about 32.7% and decreases controller load variance by 28.5% when controllers are under high load condition.

Keywords: software-defined networking; load balancing; switch migration; multi-objective optimization

(Submitted on May 16, 2018; Revised on June 13, 2018; Accepted on July 24, 2018)

1. Introduction

Software Defined Networking (SDN), a novel network architecture, has achieved full decoupling between the control plane and the data plane, and it brings abstractions to the control layer of the network [1]. Generally, SDN has the remarkable characteristics of decoupling between the control plane and the data plane, centralized control, and programmable ability, which enables flexible network management [2].

In the SDN architecture, the SDN controller is responsible for providing a programmatic interface, where applications could be written to implement management tasks and afford new functions [3], such as Ryu [4], NOX [5], Floodlight [6], and so on. The SDN forwarding device, such as the OpenFlow switch, includes flow tables consisting of flow entries, each of which determines how a packet performs. The controller updates these flow tables and instructs the switches on which actions they should take via southbound interface.

With the growth of network traffic and the network scale, the single and centralized controller brings about potential issues of scalability and reliability. Therefore, multiple controllers are deployed to improve the scalability of the control plane. However, the existing multi-controller deployments statically map the control relationships between switches and controllers, and they cannot adapt for time-varying traffic, resulting in load imbalance among controllers [7]. This undesirable situation will increase the controllers' response time to process requests from switches and reduce their throughputs, eventually degrading the control plane's performance.

Switch migration, as a new dynamic adjustment for controller load, is proposed to solve controller load imbalance. ElastiCon [8] is the first switch migration framework based on dynamic multi-controller architecture and provides three modules. The load measurement module collects the load of each controller and sends the load information to the load

* Corresponding author.
E-mail address: hutaondsc@163.com

adaptation decision module. The action module conducts control actions (e.g., migrating switch) to achieve the dynamic control of controllers and switches by migrating switches into the closest neighbor controller. In [9], the authors introduce the switch migration algorithm based on game theory, while setting the controller and switch as the gamer and commodity, respectively. By increasing or decreasing the commodity value of switch, the controller selects the optimal element to implement a transaction. In [10], the authors define the Switch Migration Problem (SMP) and a Network Utility Maximization (NUM) problem with the objective of maximizing the number of serving requests under the available control resource. The Distributed Hopping Algorithm (DHA) is designed to achieve optimal switch migration via the Log-Sum-Exp function. In [11], the Load Informing Strategy (LIS) is presented to make the load balancing decision locally as rapidly as possible. The overload controller collects load information of all other controllers before switch migration, but the efficiency of load balancing is not high because it is constrained by controller performance. In [12], the authors propose BalCon, which is based on two key observations: an effective switch migration should consider the communication patterns of the SDN switches, and the switch migration should be processed at the granularity of clusters. In [13], the authors consider switch migration from the perspective of scalable and crash-tolerant load balancing. The multiple controllers use JGroups to coordinate actions for switch migration, and this method supports controller failover without switch disconnection to avoid the single point of failure problem.

Based on the above analysis, the shortages of existing migration schemes are that they do not clearly describe the migration sequence and can result in long migration time and poor controller throughput due to the unreasonable migration design.

For example, in Figure 1, there are four controllers (C1, C2, C3, C4) in the network. When the switches send many packet-in requests to controllers C1 and C2, they become overloaded controllers due to their exhausted control capacities. At this time, switches S1 and S2 controlled by C1 and C2 will be migrated into the other controllers. Because of the capacity constraint, both C3 and C4 can only take over the migrated switches from C1 or C2. In the absence of rational planning for switch migration, it is easy to cause migration conflict and increase migration time. More specifically, both C1 and C2 will migrate their switches (S1 and S2) into C3 with minimum hops, but C3 cannot simultaneously accept those switches, and this phenomenon is defined as migration conflict. Moreover, if the migration continues, the C3 will become a new overloaded controller. Therefore, it is necessary to optimize the migration sequence to avoid migration conflict.

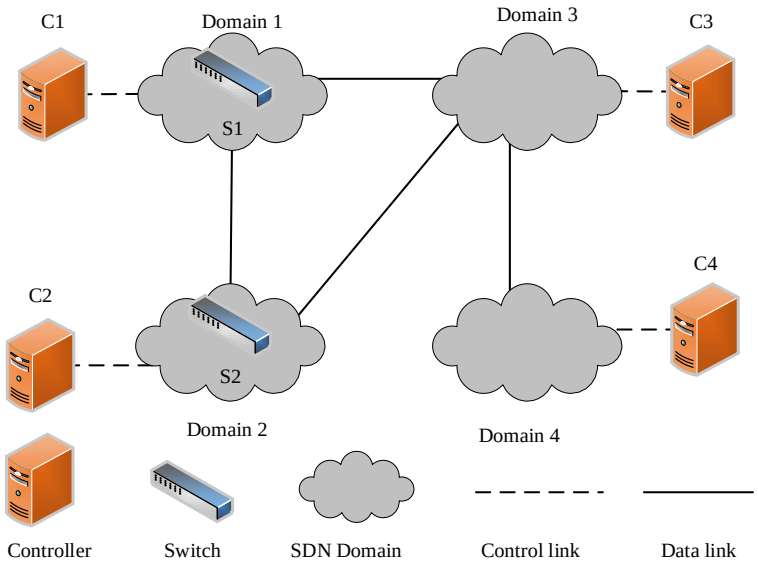


Figure 1. A motivating example

In this paper, we propose Sequenced Switch Migration (SSM) to optimize the process of switch migration. The main contributions of this paper are summarized as follows:

- We clearly introduce the sequence of switch migration and devise a sequenced strategy for switches to optimize the switch migration process and reduce the migration time.
- We divide SSM into three steps: Step 1 selects the migration object to determine the emigration domain, migrating switch, and immigration domain; Step 2 computes migration parameters (available migration duration, processing migration duration); Step 3 executes dynamic switch migration in a sequenced manner.

- We evaluate the performance of SSM against baseline schemes. The results show that SSM can reduce the total migration time by about 32.7% and decrease controller load variance by at least 28.5% when controllers are under high load condition.

The rest of this paper is organized as follows. Section II presents the modeling and formulation. Section III shows the sequenced switch migration design. Section IV presents the simulation results. Section V concludes this paper.

2. Modeling and Formulation

2.1. Network Modeling

The network topology is presented by an undirected graph $G = (V, E)$, where V and E are the set of nodes and links, respectively. The network contains M controllers and N switches, so $|V|$=M+N. G is divided into M domains, and each domain is only controlled by one controller. There is no overlap between any two domains. In other words, one switch only belongs to one domain. C denotes the set of controllers, and S is the set of switches. h_{im} is the number of hops in the shortest path (path length for short) from switch s_i to controller c_m. The flow request rate of switch s_i is λ_i. α_{m} is the processing capability of controller c_{m}. The binary variable x_{im} is the control mapping from switch s_i to controller c_m, as shown in Equation (1).

$$x_{im} = \begin{cases} 1, \ i^{th} \ switch \ \ is \, controlled \, by \ m^{th} \, controller \\ 0, \, otherwise \end{cases} \tag{1}$$

The primary notations used in this paper are listed in Table 1.

Table 1. Notations

Notation	Definition
C	Controller set
S	Switch set
h_{im}	The number of hops in the shortest path
λ_i	The flow request rate of switch
α_m	The processing capability of controller
D_{im}^{req}	The transmission delay of flow request
D_{mn}^{syn}	Controller synchronization delay
x_{im}	Binary variable: x_{im}=1 represents switch s_i is controlled by controller c_m ; otherwise x_{im}=0

2.2. Objective Function

Based on the above notations, we define several constraints for selecting migration objects, including the emigration domain, migrating switch, and immigration domain.

The transmission delay of flow request. D_{im}^{req} denotes the transmission delay from switch s_i to controller c_m,

$$D_{im}^{req} = h_{im} \cdot \frac{1}{\lambda_i} \tag{2}$$

Where x_{im} is control mapping of s_i and c_m, h_{im} is the path length from s_i to c_m, and λ_i is the flow request rate of s_i.

Controller synchronization delay. D_{mn}^{syn} denotes the synchronization delay between controllers c_m and c_n,

$$D_{mn}^{syn} = h_{mn} \cdot T \tag{3}$$

Where h_{mn} is the path length from c_m to c_n, and T is the number of controller synchronizations in the given period [7].

Therefore, the constraint of controller delay can be represented as the sum of transmission delay and synchronization delay, as shown in Equation (4).

$$D_m = \sum_{i=1}^{n} (D_{mn}^{syn} + D_{im}^{req}) \cdot x_{mn} \tag{4}$$

Control traffic. We assume each switch request must be sent to at least one controller [14]. Thus, we have L_m, the control traffic of controller c_m, as follows.

$$L_m = h_{im} \cdot \lambda_i \tag{5}$$

Residual control capacity. When implementing switch migration, we must consider the residual control capacity of the immigration domain. Thus, we compute the residual control capacity of controller c_m in Equation (6).

$$R_m = \alpha_m - \sum_{i=1}^{n} \lambda_i \cdot x_{im} \tag{6}$$

Controller failure probability. One controller has two statuses: active status and failure status [9]. A controller is in the active status by default and connects at least one switch, otherwise it is in the failure status. F_m denotes the failure probability of a controller,

$$F_m = \prod_{i}^{N} \prod_{m}^{M} p \cdot x_{im} \tag{7}$$

Where p is the probability of link failure, and x_{im} is the control mapping of s_i and c_m.

Switch migration cost: We generate the migration cost of controller c_m to show impact of metrics on switch migration:

$$\Gamma_m = \mu_1 \cdot D_m + \mu_2 \cdot L_m + \mu_3 \cdot R_m + \mu_4 \cdot F_m \tag{8}$$

Where μ_1, μ_2, μ_3, and μ_4 are the corresponding weights of D_m, L_m, R_m, and F_m, respectively [10].

2.3. Problem Formulation

We formulate the immigration domain selection problem, which is an optimization problem to select the immigration domain that can minimize the migration cost metric. The immigration domain selection problem is shown below:

$$Minimum \ \ \mu_1 \cdot D_m + \mu_2 \cdot L_m + \mu_3 \cdot R_m + \mu_4 \cdot F_m \tag{9}$$

subject to

$$\sum_{i}^{N} \sum_{m}^{M} y_{im} \cdot \lambda_i \leq \alpha_m \tag{10}$$

$$\sum_{m=1}^{M} x_{im} = 1 \tag{11}$$

$$\forall i, m \ \ x_{im} \in \{0,1\} \tag{12}$$

Equation (9) shows that this problem is a multi-objective mixed optimization, considering four objectives (controller delay, control traffic, residual capacity, and controller failure). Equation (10) demonstrates that the total switch requests do not exceed the residual processing capacity of each controller. Equation (11) indicates that each switch is only controlled by one controller. Equation (12) states that the connection relationship between a switch and a controller is a binary variable.

It is hard to solve the multi-objective mixed optimization problem in polynomial time. Therefore, we propose a heuristic method to solve it. However, the traditional heuristic methods (e.g., simulated annealing, greedy algorithm) are suitable for solving the problem within two optimization objectives. Thus, we introduce the genetic algorithm, which is appropriate for solving the problem with three optimization objectives. The implementation is shown in Step 1 in Section III.

3. Sequenced Switch Migration (SSM)

3.1. Overview

Sequenced Switch Migration (SSM) is a solution for solving migration sequence and reducing migration time. SSM implements switch migration from the perspective of optimizing the migration sequence to improve load balancing. Figure 2 shows the overall design of SSM, and it consists of three steps: (1) migration object selection; (2) migration parameter computation; and (3) dynamic switch migration. In step 1, we select the migration object to determine the emigration domain, migrating switch, and immigration domain by considering the flow request rate and migration cost. In step 2, we compute migration parameters defined as the available migration duration and the processing migration duration of switch for preparing switch migration. In step 3, we implement dynamic switch migration in a sequenced manner by comparing migration parameters.

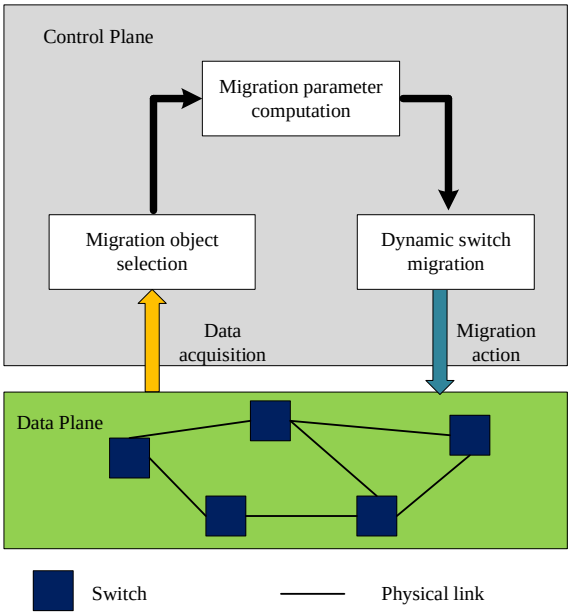


Figure 2. SSM overview

3.2. Detailed Description

In this subsection, we detail the procedure of the SSM as follows, which includes three steps.

3.2.1. Step 1: *Migration Object Selection*

Based on the above definitions, we select three migration objects (emigration domain, migrating switch, and immigration domain) one by one. The process of step 1 is shown in Algorithm 1. Specifically, we find the controller with the minimum residual capacity and select its controller's domain as the emigration domain (lines 1 to 2). The traffic rate of network is obtained based on the historical information. In the emigration domain, we select the switches, whose flow request rates are larger than the domain's average load, as the migrating switches (line 3). We use a genetic algorithm to solve it (lines 4 to 8).

Algorithm 1 Migration objects selection

Input: network G=(V, E)

Output: emigration domain G_h ; migrating switch S_M ;

immigration domain G_l

1: Compute controller residual capacity $\alpha_h = \min\{\alpha_1, \alpha_2, \cdots, \alpha_M\}$

2: Select the domain with α_h as **emigration domain** G_h

3: In G_h , Select switches ($\lambda_i < \overline{\lambda}$) as **migrating switch set** S_M

4: Use genetic algorithm to solve immigration domain selection

5: Neighbour domain of G_h is $\{G_{h-1}, G_{h-2}, G_{h-3}, \cdots, G_{h-r}\}(r < M)$

6: Compute Γ_r for G_{h-r} , $G_{h-m}(\Gamma_m \le \Gamma_{avg}), G_{h-n}(\Gamma_n > \Gamma_{avg})$

7: Crossover and Mutation for G_{h-m}, G_{h-n}

8: Select the domain with $\min\Gamma$ as **immigration domain** G_l

3.2.2. Step 2: *Migration Parameter Computation*

In this step, we compute the duration parameters for switch migration. The purpose of this step is to compute several parameters for switch migration. In the set of migration switches, switches are migrated one by one, and thus switches are sequenced to each other. To prevent many switches from being migrated to the same controller, we need to carefully design a metric to decide which switches in the set of migration switches should be migrated during each migration process. The process of step 2 is shown in Algorithm 2. We first compute the available migration duration, the processing migration duration, and the TTM of each switch (lines 1 to 3). We also set the dynamic load thresholds for the controllers. Based on this threshold, we prevent switches from being too excessive or too few in the domain.

Algorithm 2 Migration parameters computation

Input: metric h, T, λ , p, x

Output: available migration duration $Y(s_i)$; processing migration duration $T(s_i)$; dynamic load threshold L_m^{thres}

1: Compute **available migration duration:** $Y(s_i) = h_{im} \cdot D_m$

2: Compute **processing migration duration:** $T(s_i) = D_m$

3: Compute TTM of s_i : $TTM(s_i) = Y(s_i) - T(s_i)$

4: Compute $L_{\max}$, $L_{\min}$, and L_{avg} ,which are the maximum, minimum and average control traffic load of domain controller.

5: Get **dynamic load threshold:**

$L_m^{thres} = \min[(L_{\max} - L_{\min})(L_{\max} / L_{\min}), L_{avg}]$

3.2.3. Step 3: *Dynamic Switch Migration*

SSM implements dynamic switch migration in a sequenced manner by comparing migration parameters. We define a successful switch migration as meeting two requirements: (i) a switch's TTM is larger than zero; and (ii) a switch's target switch's load does not exceed the switch's load threshold. Algorithm 3 shows the process of step 3. In brief, for one migrated switch, if its available migration duration is less than the processing migration duration, then it will be forbidden to migrate until meeting the migration requirement. Initially, a migrating switch's available migration duration is larger than its processing migration duration, and SSM selects a switch to implement the first migration (line 1). In the next migration, the number of switches in the migrating switch set will be reduced. The available migration duration of rest switches will subtract one, and SSM then compares zero with TTM to determine whether the migration conditions are met (lines 4 to 7). If the migrating switch set is empty or the emigration controller load exceeds the dynamic load threshold, the migration ends (lines 8 to 12).

Moreover, the frequency of switch migration depends on the number of overloaded controllers. Referring to relevant literature and data, we design this specific principle: when the overloaded controller number exceeds one-third of the total number of controllers, the switch migration is triggered.

Algorithm 3 Dynamic switch migration

Input: emigration domain G_h ; migrating switch S_M ;
immigration domain G_l
Output: New connection relationships
1: $\exists s_m^M, \lambda_m \cdot h_{ml} = \max\{\lambda_1 \cdot h_{1l}, \lambda_2 \cdot h_{2l}, \cdots\}$
2: $G_h = G_h \setminus \{s_m^M\}$, $G_l = G_l \bigcup \{s_m^M\}$, $S_M = S_M \setminus \{s_m^M\}$
3: $L_l^* = L_l + h_{ml} \cdot x_{ml} \cdot \lambda_m$
4: $Y(s_n^M)^* = Y(s_n^M) - 1$
5: ***if*** $\exists s_n^M \in S^M, TTM(s_n^M) = Y(s_n^M)^* - T(s_n^M) < 0$
6: s_n^M cannot meet the migration conditions, Reset $Y(s_n^M)^*$
7: ***end if***
8: ***if*** $S_M = \phi$ ***or*** $L_l^* \geq L_l^{thres}$
9: ***then*** Migration end
10: ***else*** Return line 6
11: ***end if***
12: Return new connection relationships between switches and controllers

4. Simulation

4.1. Experiment Setting

We select ONOS [15] as the experimental controller and use Mininet [16] as a test platform. ONOS is programmed by Java and supports multiple versions of OpenFlow protocols. Mininet, developed by Stanford University, is set as the test platform. The physical devices contain five servers, whose configurations are Intel Core i5 3.3GHz 8GB RAM. The controller runs on the server as the software. The operating system is Ubuntu 16.04. SSM is designed in the application layer of the ONOS controller. Moreover, we select the authoritative network topologies (Abilene with 12 nodes and 15 links; Internet2 OS3E with 34 nodes and 42 links) to make the experiments more persuasive. We use Iperf to generate TCP flows to simulate the distribution of the network traffic [17]. The average flow requests are 200 packets/s. The controller capacity is limited to 2000 packets/s. The link bandwidth is finite, thus we set the number of switches managed with one controller from 5 to 20 [10].

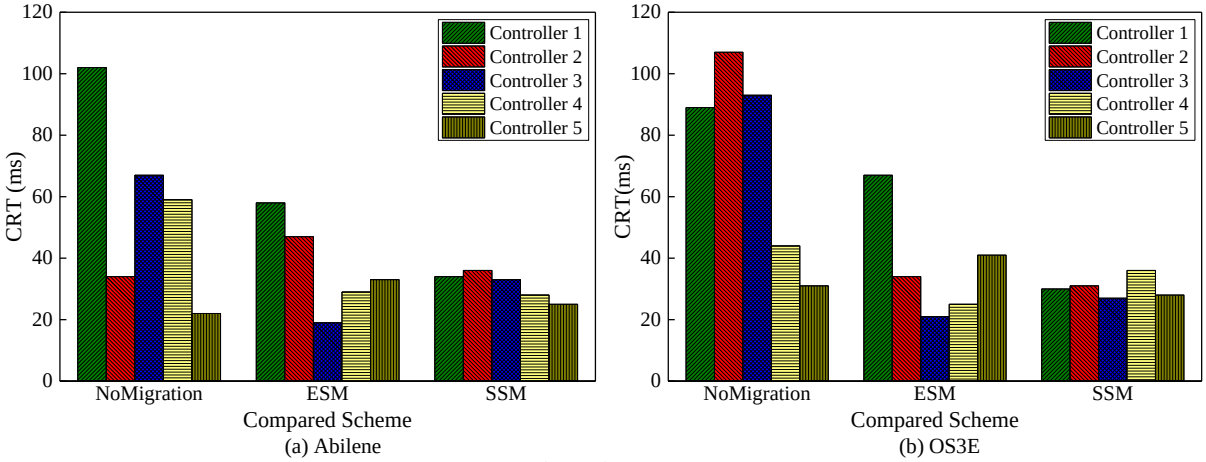
4.2. Compared Schemes

- **NoMigration:** the connections between controllers and switches are static in the network.
- **Existing Switch Migration (ESM):** ESM migrates the switches into the controllers with the minimum hops, which creates the intuitive contrast effects [8].
- **Sequenced Switch Migration (SSM):** the switch migration is implemented in a sequenced manner to reduce migration cost and improve load balancing.

4.3. Simulation Results

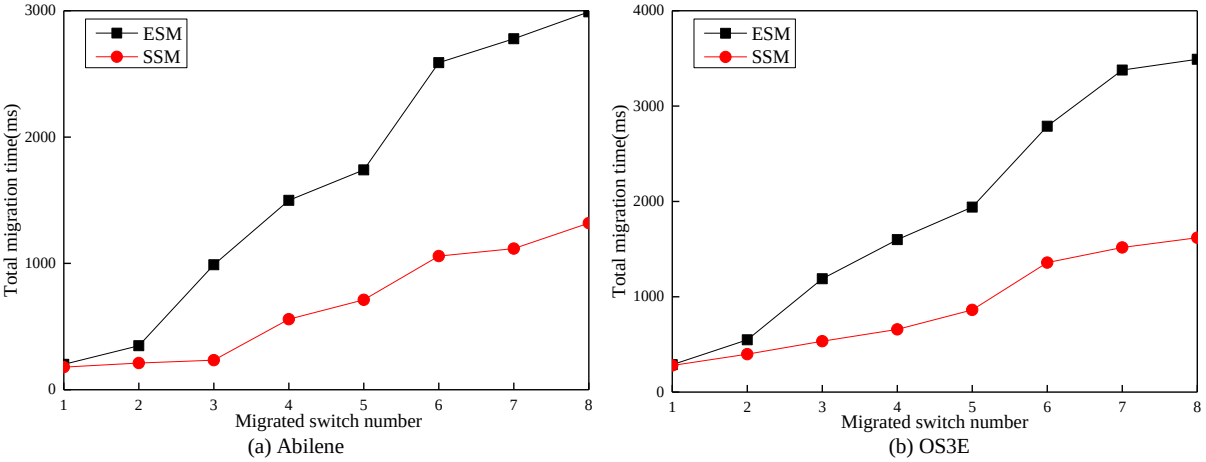
Figure 3 shows the results of the controller response time (CRT) to switch in Abilene and OS3E topologies, respectively. Because of the uneven controller loads, controllers need more time to process switches' requests in NoMigration schemes. ESM reduces the longest CRT but cannot promise the balanced load allocation among controllers, leading to a high variation in CRT. SSM selects migration objects by considering multiple metrics, and its sequenced migration design ensures that each controller has an approximately equal load. Therefore, SSM performs best, and its CRT is reduced by about 26.9%.

Figure 4 shows the results of total migration time in Abilene and OS3E topologies, respectively. Because NoMigration does not implement switch migration under controller load imbalance, we only compare ESM and SSM. It is clearly seen that, regardless of the topology, SSM has obvious advantages in terms of reducing migration time compared with ESM. When the migrated switch number is small, the three schemes have similar performance. However, as the migrated switch increases, the ESM will lack planning for switch migration, and unordered switch migration will increase the migration time for the network. In contrast, the proposed sequenced switch migration reasonably arranges the multiple switch migration by setting the available migration duration and processing migration duration. Therefore, SSM reduces the total migration time by about 32.7%.



(a) Abilene (b) OS3E

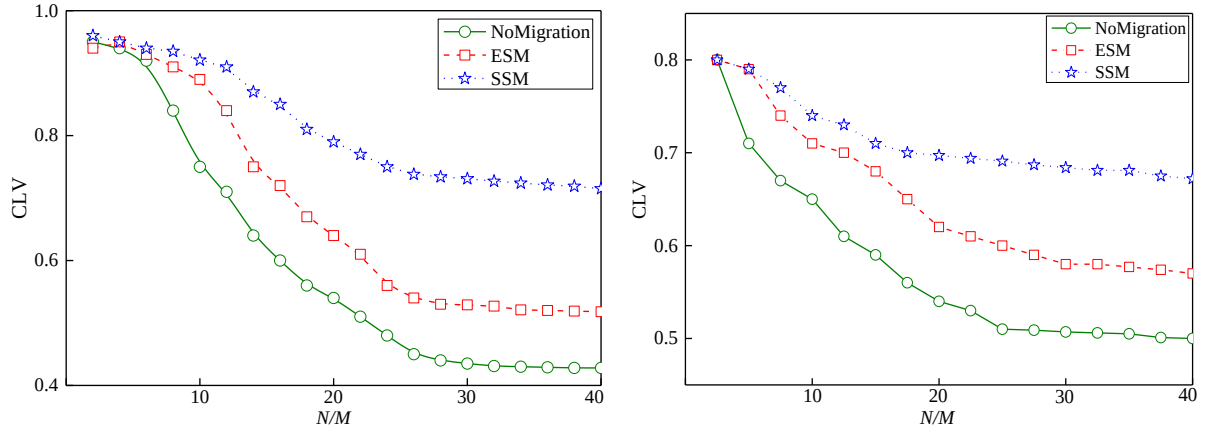
Figure 3. Comparison of CRT for NoMigration, ESM and SSM



(a) Abilene (b) OS3E

Figure 4. Comparison of total migration time for ESM and SSM

Figure 5 shows the results of the controller load variance (CLV), which represents the load balancing performance of controllers. Moreover, the smaller the CLV value, the better the performance of the control plane. Here, N/M is the ratio of switches and controllers. As the value of N/M increases, the number of switches and the controllers' control burden increases. By migrating switches, the performance of ESM is better than that of NoMigration. SSM has the best performance because its sequenced migration achieves much better load balancing among controllers than ESM, and its advantage becomes more obvious under high load. Compared with the other strategies, the CLV of SSM has been reduced by at least 28.5% when controllers are under high load condition.



(a) Abilene (b) OS3E

Figure 5. Comparison of CLV value for NoMigration, ESM and SSM

5. Conclusions

This paper designs SSM to optimize migration sequence and improve migration efficiency. SSM considers migration costs, including delay, traffic, residual capacity, and failure, in the migrating process and achieves efficient switch migration in a sequenced manner. We implement SSM in three steps: migration objects selection, migration parameters computation, and dynamic switch migration. Finally, the simulation results demonstrate the advantage of SSM against existing schemes. The total migration time has been reduced by about 32.7% and the controller load variance by at least 28.5%. In the future, we will focus on the fault tolerance ability of the control plane and research heterogeneous controllers in the network.

Acknowledgements

This work was supported by the Project of National Network Cyberspace Security under Grant 2017YFB0803204, the National High-Tech Research and Development Program of China (863 Program) under Grant 2015AA016102, the Foundation for Innovative Research Group of the National Natural Science Foundation of China under Grant 61521003, and the National Natural Science Foundation of China under Grant 61502530.

References

1. M. Aslan and A. Matrawy. "On the Impact of Network State Collection on the Performance of SDN Applications," *IEEE Communications Letters*, Vol. 20, No. 1, pp. 5-8, 2016
2. T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based Software-Defined Networking: A Survey," *IEEE Access*, Vol. 6, pp. 15980-15996, 2018
3. X. F. Gao, L. H. Kong, W. C. Li, W. C. Liang, Y. X. Chen, and G. H. Chen, "Traffic Load Balancing Schemes for Devolved Controllers in Mega Data Centers," *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 2, pp. 572-585, 2017
4. J. H. Cox, S. Donovan, R. J. Clarky, and H. L. Owen, "Ryuretic: A modular Framework for Ryu," in *Proceedings of IEEE Military Communications Conference*, Baltimore, MD, pp. 1065-1070, 2016
5. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards An Operating System for Networks," *Acm Sigcomm Computer Communication Review*, Vol. 38, No. 3, pp. 105-110, 2008
6. T. Abdullah, "Testing of Floodlight Controller with Mininet in SDN Topology," *Sciencerise*, Vol. 5, No. 5, pp. 158-158, 2014
7. T. Hu, J. Lan, J. Zhang, and W. Zhao, "EASM: Efficiency-aware Switch Migration for Balancing Controller Loads in Software-defined Networking," *Peer-to-Peer Networking and Applications*, Vol. 2, pp. 1-13, 2018
8. A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, R. R. Kompella, "ElastiCon; An Elastic Distributed SDN Controller," *Acm Sigcomm Computer Communication Review*, Vol. 43, No. 4, pp. 7-12, 2013
9. H. C. Chen, G. Cheng, and Z. Wang, "A Game-Theoretic Approach to Elastic Control in Software-Defined Networking," *China Communications*, Vol. 13, No. 5, pp. 103-109, 2016
10. G. Z. Cheng, H. C. Chen, Z. M. Wang, and S. Q. Chen, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *Proceedings of Ifip NETWORKING Conference IFIP*, 2015
11. J. K. Yu, Y. Wang, K. Pei, S. J. Zhang, and J. C. Li, "A Load Balancing Mechanism for Multiple SDN Controllers based on Load Informing Strategy," in *Proceedings of Network Operations and Management Symposium IEEE*, pp. 1-4, 2016
12. M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, "BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration," in *Proceedings of IEEE International Conference on Cloud Engineering IEEE*, pp. 40-50, 2017
13. C. Liang, R. Kawashima, and H. Matsuo, "Scalable and Crash-Tolerant Load Balancing based on Switch Migration for Multiple Open Flow Controllers," in *Proceedings of second International Symposium on Computing and NETWORKING IEEE*, pp. 171-177, 2015
14. T. Hu, P. Yi, Z. Guo, J. Lan, and J. Zhang, "Bidirectional Matching Strategy for Multi-Controller Deployment in Distributed Software Defined Networking," *IEEE Access*, Vol. 6, pp. 14946-14953, 2018
15. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards An Open, Distributed SDN OS," in *Proceedings of the Workshop on Hot Topics in Software Defined NETWORKING ACM*, pp. 1-6, 2010
16. K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform," in *Proceedings of International Conference on Communiction, Computing & Systems*, 2014
17. J. Zhang, T. Hu, W. Zhao, and D. Qiao, "DDS: Distributed Decision Strategy based on Switch Migration Towards SDN Control Plane," In *proceeding of 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (*CyberC*), Nanjing, pp. 486-493, 2017