A Load Balancing Approach for Distributed SDN Architecture Based on Sharing Data Store

Jiacong Li, Bo Lei, Ni Li, Hang Lv
China Telecom Corporation Limited Research Institute
Beijing, China
Email: lijc50@chinatelecom.cn

Abstract—**Software-defined networking (SDN) uses a centralized control plane to manage the whole network. Distributed SDN architecture has been proposed to resolve scalability and reliability problem. One challenge of multiple controllers is load balancing problem, that uneven load distribution among controllers. Dynamic switches migration is an effective way to solve this problem. However, the existing schemes focus on effective migration algorithms and different metrics to realize the load balancing of control plane in SDN, and do not consider the communication cost between controllers and computation cost of migration scheme. In order to address this issue, we propose a load balancing approach based on sharing data store. Sharing data store has the information of all controllers, and the *calculator* module in it can compute switches migration scheme based saved information. It can also reduce the computation and communication pressure of controllers. Simulation experiments exhibit the feasibility and effectiveness of our approach.**

Keywords load balancing, switches migration, sharing data store, multiple controllers, SDN

I. Introduction

Software-defined Networking (SDN) has been provided for flexible network management by decoupling the control plane from the data plane. It is proposed to improve the network performance [1-2]. SDN mainly relies on a centralized control plane to manage the whole network. This centralized control plane has a view of the whole network, based on which, it calculates the forwarding rules for switches, while the switches are just responsible for forwarding the data packets according to these forwarding rules. When the scale of the network is large, multiple controllers are required, then distributed SDN architecture [3] has been proposed.

In distributed SDN, the large-scale network is split into small parts and each part has a controller. Though this method reduces the overload of one controller, brings another challenge that how to maintain load balancing among multiple controllers. This problem can be studied from two phase, one is controllers deployment phase, the other is operation phase. This paper focuses on the latter one.

When the network in operation phase, the load of each controller may fluctuate irregularly. And more switches may be needed to join the network. Thus it is necessary to adjust the number of switches that one controller controls dynamically. There are some problems should be considered. Firstly, each controller needs to collect the load of itself and calculate which switches should be migrated, which may lead to the reduction of computing ability. Moreover, each controller has to keep communication with others to get their load information, which may take up extra computing resources of control plane. Additionally, it is hard for overload controller to calculate the solution of load balancing, because most of computing resource have been occupied. If reserve enough resources to keep load balance, it must affect the performance of the whole network.

Numerous previous work has been done in SDN load balancing area. Some of them put emphasis on data plane [4-5]. Others are related to the placement of distributed controllers [6], switch migration [7-12] and so on. However, most existing approaches for control plane based on controllers, which need to communicate with others and calculate the migration scheme. These methods may occupy extra resources of controller, and reduce the efficiency of the whole network. Towards addressing this gap, this paper makes the following contributions:

- We propose a sharing data store mechanism to guarantee the load balancing for SDN control plane. Furthermore, we add some data structure to calculate the load of the controllers based on our previous work [13].
- We propose a load balancing approach for control plan based information in sharing data store, including load measurement, switches and target controller selection.
- We implement and validate our approach in a controlled environment and conduct simulations to show the feasibility and effectiveness of the approach.

The rest of this paper is organized as follows. Section II discusses the related work. Section III describes the load balancing approach based on sharing data store design and implement. Section IV describes our experimental methodology and evaluates the effectiveness of our approach, and this paper is concluded in Section V.

II. Related Works

The load balancing solutions for control plane in distributed SDN architecture can be classified into two categories: distribute controller deployment and switches migration. The former solution is suitable for deployment phase. Tao Hu et al. [6] proposed reliable and load balance-aware multi-controller deployment (RLMD) strategy. This approach can improve the performance of the network during controller deployment, but it cannot adjust the load of controllers dynamically in operation phase.

In order to solve the load balancing problem for control plane in operation phase, switches migration has been proposed [7]. In this protocol, authors put forward a distributed nearest migration algorithm to save migration time by selecting the nearest controller to receive load shifting, which might bring about new load imbalance. Then many researchers have done a lot of researches to solve this problem. Zhuo Li et al. [8] designed a load balancing mechanism of SDN controller based on reinforcement learning. Kai-Yu Wang et al. [9] proposed a load adjustment mechanism for application to each controller. To solve the frequent and unnecessary switch migration problem in absolute load-balancing, Yaning Zhou et al. [10] proposed a load balancing mechanism based on switches group for multiple controllers. This mechanism not only balanced the load among controllers, but also solved the load oscillation and improved time efficiency. Jie Cui et al. [11] presented a novel load-balancing strategy based on real-time response time, called SMCLBRT. It not only can avoid unnecessary migrating costs and sufficiently utilize the controller's performance, but also improve the processing rate of the load balancers.

However, the switch migration approaches mentioned above may affect the performance of controllers, because they need monitor load in real time and compute migration scheme when the load exceeds the threshold. At last, they must execute the migration scheme. Kai-Yu Wang et al. [9] and Yaning Zhou et al. [10] designed a load collector, balancer and migration on controller, which may occupy the resource and lead to the controller performance degradation. The method that [11] proposed is based on real-time response time, which is also depend on controller monitoring.

Some researchers put emphasis on migration cost. C. Wang et al. [12] proposed a migration efficiency model to make a tradeoff between migration cost and the load balance rate. This scheme is finely designed to select a reasonable migration pairing. However, the SMDM scheme is also running on the controller. Additionally, in distributed SDN architecture, controllers need to communicate with each other to calculate migration scheme, which leads to the migration cost increases.

In order to reduce the communication and computation pressure of the controller, we propose a load balancing mechanism of control plane based on sharing data store, which has been designed in previous work [13]. Additionally, we propose a switches migration scheme with considering the performance of the network after migration.

III. The Proposed Load Balancing Approach Design and Implementation

In this section, we first supplement the information that should be saved in sharing data store and describe the process of switches migration with sharing data store. In addition, we provide a switches migration scheme considering migration cost. We define the way to calculate the load of the controller, and provide two algorithms to find the switches migration group and target controller respectively.

A. The Improved Sharing Data Store

In order to keep the load balancing of the control plane without the increase of communication between controllers, we supplement information stored in sharing data store to calculate the migration scheme. The sharing data store architecture has been proposed in previous work [13].

Information Supplement. In our previous work, we have design the the architecture of sharing data store as Fig.1. (a) shows. In order to calculate the switches migration scheme, the data store should save the performance information of each controller, which can be seen in Fig.1. (b).

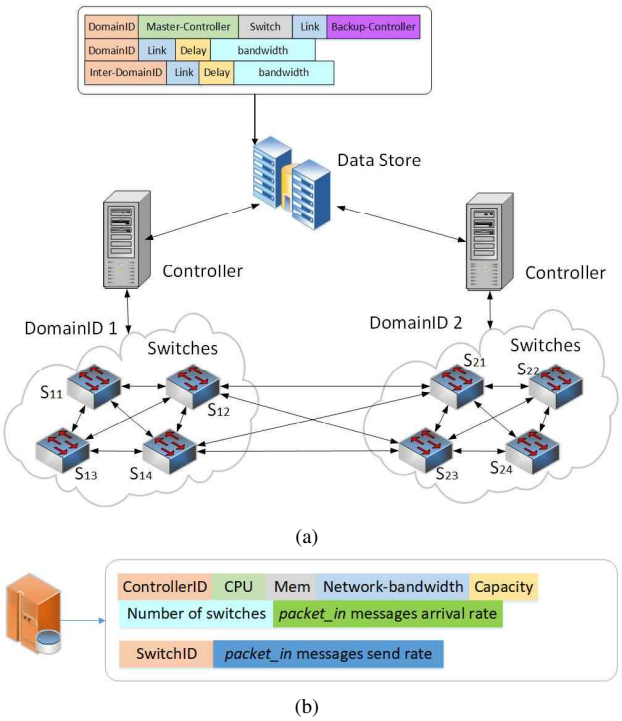


Fig. 1. The information stored in the data store

As Fig.1. (b) shows, the data store has performance information of all controllers, including the switches under control, CPU utilization, memory usage, network bandwidth occupancy, capacity and the PACKET_IN messages arrival rate. With these data, the data store can compute the load ratio of each controller. Combining the load ratio and distance shown in Fig.1.(a), the calculator module in data store can compute the switches migration scheme, then the data store saves the current network state and sends the scheme to the related controllers. When the scheme has been executed successfully, the new network state will be stored. Otherwise the data store sends roll back message to the related controllers to recover to the last network state.

Switches Migration. When the controller becomes overload, then some switches should be migrated away from it. In general, this controller needs to calculate switches migration group and the target controller, which may lead to extra resource consumption. In order to avoid unnecessary cost, we use sharing data store to compute the migration scheme.

When the controller becomes overload, it sends message to data store and gets the migration scheme from the data store.

Fig. 2 depicts the process of switches migration. In this case, controller C1 is the overload controller, then some switches controlled by C1 need to be migrated to controller C2. When the load ratio of controller C1 is heavier than threshold, the following steps are executed:

Phase 1. Controller C1 sends an overload warning message to data store, then the data store calculates which switches should be migrated away from Controller C1 and which controller should to control these switches. The data store sends the result to Controller C1. When the Controller C1 knows the target controller, they establish connection between each other to make preparation for migration.

Phase 2. Controller C1 sends a barrier request message to switches to interrupt these switches sending requests. The switches reply this request and stop sending messages. After controller C1 processes the request received before, it sends a flow mod message to switches. After switches reply it, controller C1 sends role mod message to change the master controller of migrated switches. Next Controller C1 sends a start migration message to controller C2 to start the migration process. Then controller C2 sends a barrier cancel request message to migrated switches to cancel their barrier state. After that, it sends an end migration message to controller C1.

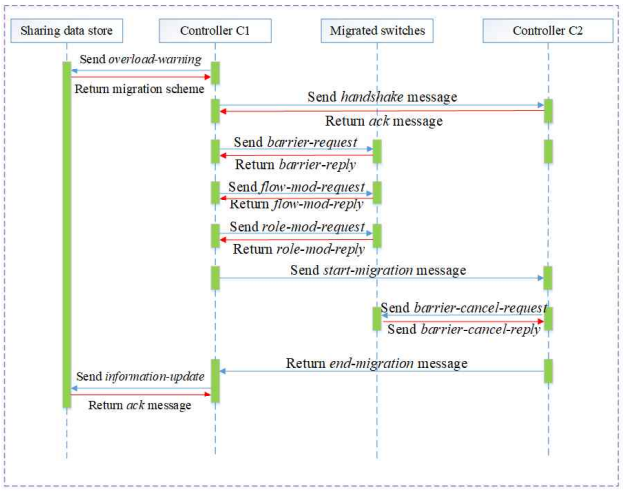


Fig. 2. The process of switches migration

Phase 3. At last, controller C1 sends an information update message to data store to update the new network state, and then data store confirms with an ack message.

B. Switches Migration Scheme

In order to compute the switches migration scheme without the increase of computation and communication of each controller, we use the calculator module in the data store. Because the data store has the whole network state and the performance information of each controller, it can compute the switches migration scheme with enough data. A summary of the mathematical symbols used in this section and their meanings are shown in Table I.

TABLE I
THE SUMMARY OF SYMBOLS USED IN THIS PAPER

Symbol	**Definition**
C	Set of controllers $C_1, C_2 \ldots C_i$
S	Set of switches $S_1, S_2 \ldots S_i$
M	Set of migration switches groups
OC	Overload controller during migration
TC	Target controller of migration scheme
$S_k \in MAP_{C_i}$	Switch S_k controlled by controller C_i
L_{C_i}	Load of controllerC_i in the network
$L\_max_{C_i}$	Load capability of controller C_i
n_{S_k}	Number of packet_in messages S_k sends
R_{C_i}	Resource utilization of controllerC_i
LR_{C_i}	Load ratio of controller C_i
L_{S_k}	Load of switch S_k in the network
L_m	Load size should be migrated away
$delay(S_k, C_i)$	Delay between S_k and C_i

1) Load Measurement: The load measurement component runs on the data store to calculate the load of each controller and switch. The load of a controller comes from two sources: the number of PACKET_IN request messages from switches and the load of running server. The former mainly includes the sum of PACKET_IN messages that each switch sends. The latter one includes the utilization of CPU, memory and network bandwidth of running server. The load of controller can be represented as calculation formula (1) shows, where w means the weight of each parameter, and the sum of weights equal 1, as equation (2) shows. The load of running server is represented as formula (3) shows, where U_{CPU}, U_{mem} and U_{net} represent the utilization of CPU, memory and network bandwidth respectively.

$$L_{C_i} = w_1 \sum_{S_k \in MAP_{C_i}} n_{S_k} \quad w_2 \quad R_{C_i} \tag{1}$$

$$w_1 \quad w_2 = 1 \tag{2}$$

$$R_{C_i} = U_{CPU} \quad U_{mem} \quad U_{net} \tag{3}$$

The processing capacity of controller can be described by formula (4), where $N\_max_{C_i}$ means the max number of PACKET_IN messagse that controller can process, and $R\_max_{C_i}$ represents the max utilization of CPU, memory and network bandwidth of running server.

$$L\_max_{C_i} = w_1 \quad N\_max_{C_i} \quad w_2 \quad R\_max_{C_i} \tag{4}$$

$$LR_{C_i} = \frac{L_{C_i}}{L\_max_{C_i}} \tag{5}$$

Considering different physical machines has different processing ability, we use load ratio to represent the load condition of each controller, as formula (5) shows. When the load ratio exceeds the threshold, the data store starts to calculate the switches migration group and the target controllers. It is not necessary to calculate load ratio of

each controller in real time, otherwise it may lead to more communication between controllers and data store.

2) Switches Selection: It is important to select switches which should be migrated away from the overload controller. If select the light load switches, it cannot make the load balanced and need migration many times. If select the high load switches, it may lead the new controller overloaded. Besides the load of switches, delay between the migration switches and the overload controller should be considered.

First, we calculate the load size that should be migrated away to make load balanced. The equation (6) should be satisfied after migration, which two controllers have equal load ratio. In this equation, C_i means the overload controller, and the C_j represents the target controller.

$$LR_{C_i} = LR_{C_j} \tag{6}$$

According to the formula (5), the equation (6) can be represented by equation (7).

$$\frac{L_{C_i} \quad L_m}{L\_max_{C_i}} = \frac{L_{C_j} \quad L_m}{L\_max_{C_j}} \tag{7}$$

Then solve the equation (7), we get the load size as formula (8) shows.

$$L_m = \frac{L\_max_{C_i} \quad L_{C_j} \quad L\_max_{C_j} \quad L_{C_i}}{L\_max_{C_i} \quad L\_max_{C_j}} \tag{8}$$

If there are two target controllers, it can be solved by equation group (9). In this paper, the number of target controller is two at most, otherwise, the cost of migration may overload.

$$\begin{cases} \frac{L_{C_i} \quad L_m}{L\_max_{C_i}} = \frac{L_{C_j} \quad L_{m_j}}{L\_max_{C_j}} = \frac{L_{C_k} \quad L_{m_k}}{L\_max_{C_k}} \\ L_m = L_{m_j} \quad L_{m_k} \end{cases} \tag{9}$$

With the migration load size, we can compute the migration switches groups for overload controller. The procedure of calculation is shown as Algorithm 1.

Algorithm 1 Switches_Selection

Input: S={$\forall S_k$,$S_k \in MAP_{OC}$,L_m,n_S
Output: M

```
1: i = 0, j = 0, l = 0;
2: while (i   S.length) do
3:     M_i = InitM(), j = i, l = 0;
4:     while (j   S.length) do
5:         l = l   n_{S_j}
6:         if (l   L_m) then
7:             S_j ∈ M_i;
8:             j = j   1;
9:         else
10:            break;
11:        end if
12:    end while
13:    M_i ⊆ M;
14:    i = i   1;
15: end while
```

According Algorithm 1, there are many switches groups which satisfy the load size. Considering the migration cost and the performance of the whole network, it is necessary to make the number of migration switches minimum and the delay between them and overload controller maximum at the same time, as function (10) shows, where M_i represents the number of switches in migration group M_i.

$$F_1 = w_1 \quad \min M_i \quad w_2 \quad \max \sum_{S_k \in M_i} delay(S_k, OC) \tag{10}$$

The weight can be set in different situations flexibly. With function (8), algorithm 1 and function (10), the migration switches can be selected.

3) Target Controller Selection: In order to avoid the load oscillation among controllers and improve the network balancing, it is necessary to find the right target controller. First we define the controller balance indicator based on variance as formula (11) shows, where $\overline{LR_C}$ represents the average load ratio of all controllers in the network, as formula (12) shows.

$$V = \sum_{C_i \in C} (LR_{C_i} \quad \overline{LR_C})^2 \tag{11}$$

$$\overline{LR_C} = \frac{\sum_{C_i \in C} LR_{C_i}}{C} \tag{12}$$

The value of V can be used to evaluate the load balance situation. The value of V is larger, the balancing of network is worse. In general, controller with the lightest load ratio may be selected, but if a new controller is put in the network, the load of it may increase rapidly. Additionally, the delay between the target controller and migration switches need to be considered. To avoid the load ratio of target controller grows fast, the data store sends the migration command to only one controller, until it receives the finished migration message, it can send another migration command. Considering the delay, the objective function can be described by function (13).

$$F_1 = w_1 \quad \min V \quad w_2 \quad \max \sum_{S_k \in M_i} delay(S_k, TC) \tag{13}$$

To make the value of V minimum and maintain the network stability at the same time, we assume that the migration times are two at most. Presuming there are n switches needed to be migrated, and two candidate target controllers. Thus the migration switches should be divided into two groups, the number of different combinations can be represented as formula (14) shows.

$$N_{group} = C_n^0 \quad C_n^1 \quad C_n^2 \quad \ldots \quad C_n^{\lfloor \frac{n}{2} \rfloor} \tag{14}$$

From formula (14), it easily can be seen that the number of n and target controllers decides the complexity of calculation. Therefore, we consider the size of n in function (10), and assume the number of target controllers is two at most in this paper. The calculation ability is depend on the calculator module in data store, this two parameters can be adjusted in different computation environment.

With the group size, migration switches groups and formula (13), we can select the final migration scheme, the procedure of calculation is shown in Algorithm 2.

Algorithm 2 Target controller_Selection

Input: $G_{i1} \cap G_{i2}=\emptyset$ && $G_{i1} \cup G_{i2}=M_i$ $M_i \in M$, $F_2, N_{group}, V,\ TC_1,\ TC_2$

Output: result = G_{best}, TC_{best}

```
i = 0;
while (i   N_group) do
    temp1 = F2(G_i1, TC1);
    temp1 = F2(G_i1, TC2);
    result = min reult, temp1, temp2 ;
    i = i   1;
end while
```

IV. Validity Evaluation

In this section, we conduct simulations to verify the validity of our approach. We first introduce our experimental environment, and then we discuss the metrics we used to evaluate our approach. Last, we show the results of the evaluation metrics of our method, and the results of the comparison with other approaches.

A. Experimental Environment

We use the network topology from Brite topology generator [14], which can generate multi-domain network randomly. We deploy OpenDaylight as our controller. We choose a physical machine to run the lightweight network simulation tool, Mininet [15]. In this experimental, we deploy five controllers in the network. For the validity evaluation, we apply great stress on the control plane traffic load and avoid emulating the high overhead data plane or transmitting packets through data plane. Thus we modify OpenvSwitches to inject PACKET_IN messages to a controller to simulate overload.

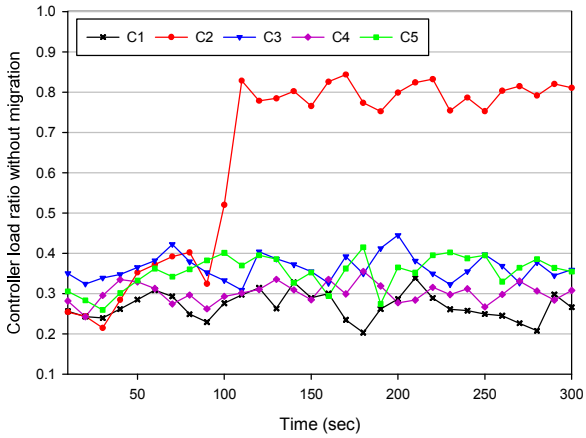
B. Evaluation Metrics

The following performance metrics are used to evaluate our approaches. First, load ratio, we use this metric to testify our approach can make load balancing for control plane. Secondly, we compare migration time of our approach with different ones, that have been proposed in [10-11]. Last we compare migration cost of different approaches, this metric includes the number of migration switches and delay between switches and controllers after migration.

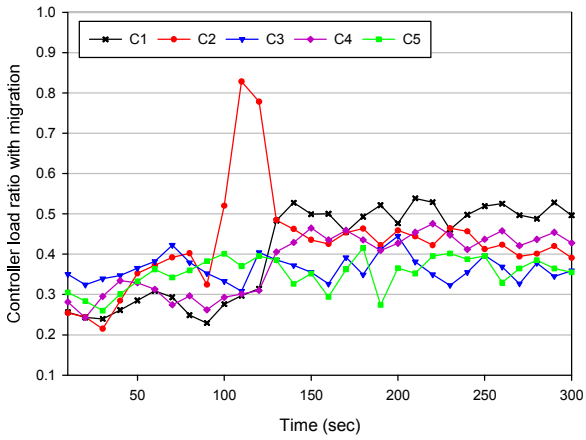
C. The Result of Evaluation

1) Balance Validity: Fig.3. shows the load ratio of five controllers under two different scenarios. One switch-controller mapping remains static and the other one uses the our approach. To simulate a controller overload while pushing one of controllers close to performance bottleneck, we adjust the sending rate of PACKET_IN messages at 100s according to the OpenDaylight performance test report by SDNCTC [16]. Thus, the load of C2 becomes heavy.

It can be seen in Fig.3.(a), without any switches migration processing, the controller C2 continue to be overloaded in the static mapping model. On the contrary, as shown in Fig.3.(b), based on the sharing data store, after switches migration, the load ratio of C2 are quickly down to light load controllers. Therefore, our scheme effectively realize load balancing of the control plane.



(a)



(b)

Fig. 3. The load ratio of five controllers under different scenarios

2) Balance Time: Fig.4. shows the load ratio of overload controller during load balancing processing with three different schemes. When we use our scheme, the overload controller C2 transfers its switches to other controllers C1 and C4. The load ratio of five controllers tend to be balanced at about 125 s. When we use Scheme II [10] and Scheme III [11] under the same circumstances, the network tends to be balanced at 130 s and 125 s, respectively.

It can be seen in Fig.4. that our scheme can balance the uneven network more quickly than the other two schemes, but the migration start time of our approach is a little later than others. Because the overload controller needs to communicate with the sharing data store first in our scheme. After receiving the migration scheme, the load ratio of controller C2 down to light more quickly than other schemes and the load ratio of C2 is lighter than others, due to the

overload controller has better performance.

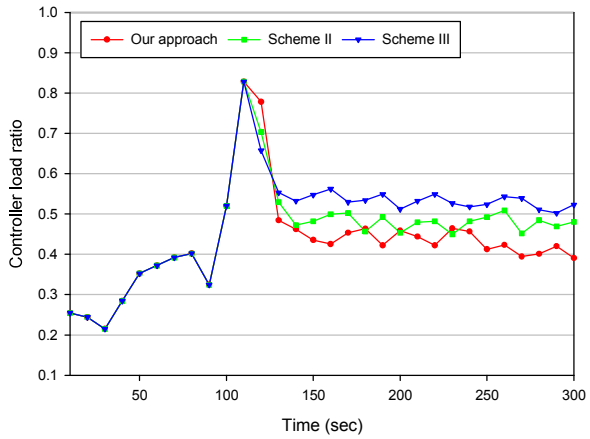


Fig. 4. The load ratio of overload controller under different scenarios

3) Migration Cost: The migration cost includes the number of migration switches and the average delay between switches and controllers after migration. These metrics can affect the whole network service due to migration. As shown in Fig.5, our scheme has the same number of migrated switches with Scheme III, less than scheme II. Moreover, the average delay between switches and controllers after migration is minimum among three load balancing approaches.

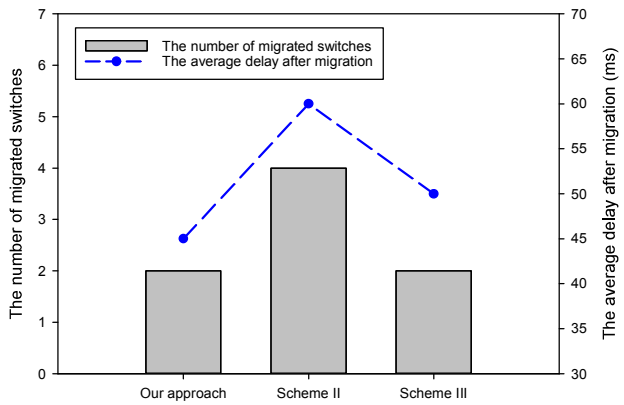


Fig. 5. The migration cost of different schemes

A few reasons can explain the result shown in Fig.5. Scheme II selects some switches as a switch group to save migration time, but it increases the number of migrated switches, and ignores the delay between switch and controller during migration. Scheme III selects the most influential switch to migrate, it can efficiently reduce the burden of the overloaded controller, but it ignores the performance of the network after migration. Thus it increases the delay on normal network services.

V. Conclusion

In this paper, we have investigated the uneven load distribution among multiple SDN controllers. In order to reduce computation resource consumption, we propose a load balancing approach based on sharing data store. We first supplement load computation information in the data store, then design the load balancing process based on the sharing data store. Additionally, we improve the switches migration scheme, considering the number of migrated switches and the performance of the whole network during migration. Simulations show that our approach can realize load balancing of control plane in SDN quickly and effectively.

References

[1] H. Kim, N. Feamster “Improving network management with software defined networking.” In: Communications Magazine, IEEE. 2013, 51(2), pp.114-119.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. M. Parulkar, L. L. Peterson, J. Rexford, S. Shenker, and J. S. Turner, “Openflow: enabling innovation in campus networks,” Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

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

[4] M.C. Nkosi, A.A. Lysko, S. Dlamini, “Multi-path Load Balancing for SDN Data Plane,” in The International Conference on Intelligent and Innovative Computing Applications (ICONIC), December 2018.

[5] C. Cui , Y. Xu, Research on Load Balance Method in SDN[J]. International Journal of Grid & Distributed Computing, 2016, 9(1):25-36.

[6] T. Hu, P. Yi, J. Zhang, et al. Reliable and load balance-aware multi-controller deployment in SDN[J]. China Communications, 2018, 15(11):184-198.

[7] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, “Towards an elastic distributed sdn controller,” in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN ’13. New York, NY, USA: ACM, 2013, pp.7–12.

[8] Li Z, Zhou X, Gao J, and Qin Y, “SDN Controller Load Balancing Based on Reinforcement Learning,” in IEEE 9th International Conference on Software Engineering and Service Science, November 2018.

[9] K.Y. Wang, S. J. Kao, and M.T. Kao, “An efficient load adjustment for balancing multiple controllers in reliable SDN systems,” in IEEE International Conference on Applied System Innovation (ICASI), April 2018.

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

[11] J. Cui, Q. Lu, H. Zhong, et al. A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time[J]. IEEE transactions on network and service management, 2018, 15(4):1197-1206.

[12] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, “A switch migration-based decision-making scheme for balancing load in sdn,” IEEE Access, vol. 5, pp. 4537–4544, 2017.

[13] J. Li, Y. Wang, W. Li, et al. Sharing data store and backup controllers for resilient control plane in multi-domain SDN[C]// Ifip/ieee Symposium on Integrated Network & Service Management. IEEE, 2017.

[14] [Online]. http://www.cs.bu.edu/brite/.

[15] [Online]. http://mininet.org.

[16] [Online]. https://www.sdnctc.com/download/resource_download/id/6.