Computer Networks 191 (2021) 107991

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Assessing Profit of Prediction for SDN controllers load balancing

Hong Zhong [a,b,c], Jinpeng Fan [a,b,c], Jie Cui [a,b,c,*], Yan Xu [a,b,c], Lu Liu [d]

[a] Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China
[b] Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China
[c] Institute of Physical Science and Information Technology, Anhui University, Hefei 230039, China
[d] School of Informatics, University of Leicester, LE1 7RH, UK

ARTICLE INFO

Keywords:
Software-defined networking
Control plane load balancing
Switch migration
Taylor's formula
Profit assessment

ABSTRACT

Software-defined networking (SDN) provides programmable control and centralized management in data centers, making it a popular architecture. The large scale of networks has required to propose the geographical distribution of logically centralized control plane to achieve scalability and reliability. For solving the load imbalance among multiple controllers associated with the statically configured control plane, a switch migration mechanism is proposed to admit dynamic load balancing. Many studies have been carried out for solving the control plane load balancing problem based on the switch migration mechanism. However, previous studies focus on migrating the switches when the controllers are overloaded, thereby, wasting time in the switch migration phase and resulting in high latency. To address these problems, we propose the Assessing Profit Of Prediction (APOP) scheme, a load-balancing strategy in the multiple-controllers control plane based on the overloaded state prediction and profit assessment. We introduce Taylor's formula to predict the flow change in the network and assess the profit of migrating switches in advance, in order to decrease the migration time and minimize the harmful effects during the migration phase. The result of simulation experiments shows that our scheme performs effectively in reducing the migration cost in control plane load balancing.

1. Introduction

With the growing pressure of management due to the increasing number of network elements in the traditional networks, a new networking structure called software-defined network (SDN) was proposed and has been growing over the past few years. The intelligence of SDN decouples the control plane from the data plane, which provides a logically centralized management mechanism to replace the distributed functions on the specific hardware [1]. As the brain of the network, the controller provides the northbound to access the programmability directing switches in the data plane to accomplish the forwarding tasks by southbound, the OpenFlow protocol [2]. The efficiency of SDN, which consisted of three levels, enables global sight of networks and overall management. With the continuous extension of networking scale, admittance of scalability and reliability of the centralized control plane becomes a key issue in SDN. An effective approach to solve the problem is by deploying distributed controllers, with each controller managing a set of switches and corresponding each other to access a global network monitoring and centralized control capacity [3]. The structure of the distributed SDN network is shown in Fig. 1. However, deployment of the static controller-switch mapping results in load imbalance in distributed controllers. In case of multiple controllers, the load imbalance could harmfully affect the performance of the control plane. When the flow peak occurs, because of the controller placement problem, the traffic flow is distributed unevenly in the data plane; therefore, the OpenFlow requests generated in switches are uneven as well. Some controllers may handle several requests, crossing their bottleneck, thus resulting in high latency and high loss rate of packets, while others may handle less-than-required number of requests, wasting the energy utilization. Therefore, a dynamic switch migration mechanism among multiple controllers becomes an urgent demand that aims to balance the load and ensure the stability of the network when the flow peak occurs [4].

In the distributed control plane, due to the dynamic traffic, the flow requests generated by each switch vary with time, which easily causes the load of each controller to change over time as well. Usually, the following three cases result in introduction of switch migration to handle the load in the control plane. First, if the aggregated traffic load goes beyond the sum capacity of all the controllers, new controllers would be added in and switch migration occurs to update the network

* Correspondence to: School of Computer Science and Technology, Anhui University, No. 111 Jiu Long Road Hefei, Anhui Province, China.
E-mail addresses: cuijie@mail.ustc.edu.cn (J. Cui), l.liu@leicester.ac.uk (L. Liu).

https://doi.org/10.1016/j.comnet.2021.107991
Received 13 June 2020; Received in revised form 11 January 2021; Accepted 7 March 2021
Available online 22 March 2021

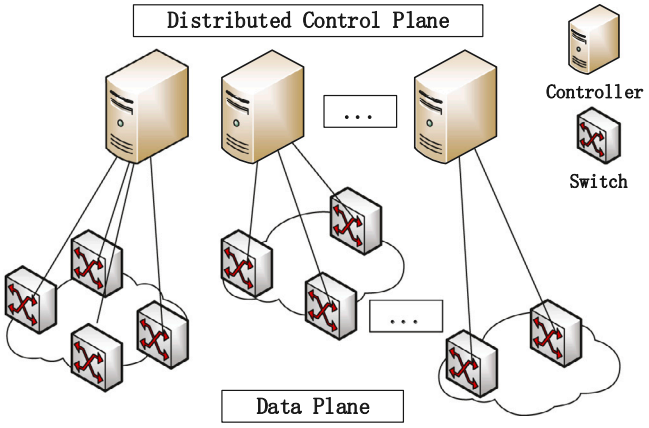


Fig. 1. The structure of distributed SDN network.

topology [4,5]. Second, controllers may crash down accidentally because of the single point of failure (SPOF), or to achieve green IT [6] and energy efficiency [7,8], controllers may shut down forwardly for saving communication cost and power; therefore, switches have to be migrated to the controllers that are still working to keep the normal operation of the network. Third, when the load of some controllers exceeds their maximum capacity and could not maintain their stable operating states, if there exist other controllers with low load, switches should be migrated to ensure load balancing.

Many studies introduce the switch migration mechanism to achieve load balancing among multiple controllers in the control plane. The OpenFlow protocol v.1.3 [9] allows each OpenFlow switch to connect to several controllers, with each controller having three patterns on the switch: master, equal, and slave. However, each switch could only have one master controller, which has full authority to access it. Hence, the concept of switch migration is to change the role of controller to that of the switch. In the beginning, most studies attempt to find out how to decide the threshold to trigger migration. Others pay more attention to map controller-switch pairs to ensure global stability. However, all of them prefer to trigger migration when controllers are overloaded, a few consider the transfer cost during migration, and no one relates to predicting controllers facing the risk of overload and migrating switches in advance.

When migration occurs, a controller has to stop accepting PACKET_IN messages from the migrated switch; therefore, it also stops issuing response messages such as FLOW_MOD or PACKET_OUT to guide packets forwarding in the data plane. In other words, network latency or even packet loss would happen during the migration. To summarize, migration time is a key issue affecting the network's performance. A switch's migration time varies with the status of its connected controller, i.e., it increases with the controller's load, which is gauged by the controller's CPU usage [10]. The communication between controllers and switches during a migration process is as follows: First, the overloaded controllers send ACK request messages to other controllers for finding out unloaded controllers, and then the other controllers, in turn, send back ACK reply messages. This step consumes the CPU resource of the controllers. Second, the overloaded controllers select the switches to be migrated and send role transfer messages to switches, which consume the CPU and bandwidth resources simultaneously. Finally, switches transfer their master controllers and finish the migration. The communication details during the migration phase are shown in Fig. 2. With the increasing of the controller's CPU usage, the switch migration time would increase with the response time of the controller. On the basis of this, we propose the Assessing Profit of Prediction (APOP) scheme, which aims to migrate the switch before the controller is overloaded to save migration cost during the load balancing. In the APOP scheme, we focus on the following three objectives:

- Shift the controller's load in advance to prevent packet loss and to reduce the migration cost during the load balancing.
- Minimize the network fluctuation during the migration process.
- Ensure the balance rate in the control plane after the load balancing.

To achieve the aforementioned objectives, our main contributions compared to related works are as follows:

- We design a prediction mechanism to select the controller that is going to be overloaded in advance.
- We assess the profit for selecting the controller, which could bring the best networking performance after migration.
- We analyze the switch's state to prevent harmful effects to the network stability, and to bring better balance rate in the control plane.

The rest of the paper is organized as follows. Section 2 provides an overview of related works. Section 3 gives our formulation of the load balancing problem. Section 4 proposes the APOP scheme, with the design and implementation of the algorithm. Section 5 presents the experiment's evaluation of the scheme and discussion of the performance. Section 6 concludes the paper.

2. Related works

The traditional SDN implementation relies on a single controller to provide centralized control and management in networks, such as NOX [11]. However, with rapid development in cloud computing, massive data become handling in data centers. Explosive requests generated in the data plane crashes in the control plane for directing forwarding rules [12]. As the result, setting up only one single controller faces the limitation of scalability and reliability in the control plane. Some previous studies proposed kinds of SDN architectures with distributed controllers such as [3,13,14]. They suppose to leverage multiple controllers to share requests management for reducing the computation pressure on a single controller. With multiple distributed controllers, the strategies like Beacon [15] and Kandoo [16] focus on improving the performance of each controller to enhance the global performance of the control plane.

Leveraging the geographical distribution control plane consisted of multiple controllers can effectively improve the performance of SDN networks. However, the implementation with multiple controllers brings a difficulty that the controller-switch mapping is statically configured, making it hard to adapt the varying of requests in the data plane; therefore, how to dynamically balance the load distribution among multiple controllers becomes a new challenge. The role transfer mechanism has been supported in OpenFlow v.1.3 [9], which implies that an SDN controller has three roles to a switch: master, slave, and equal. The controller with master or equal state has full access to the switch, while the slave controller could not manage the switch. A controller may change its role to transfer its management authority of switches. Meanwhile, the mechanism which allows dynamic switch migration can be an effective approach to balance the load in the control plane.

On the basis of the switch migration mechanism, many studies propose dynamic load balancing strategies to access the scalability and reliability in the distributed control plane. Dixit et al. [4] firstly propose the ElastiCon, an efficient protocol to enable switch migration among multiple controllers. In their scheme, overloaded controller shifts its load to the nearest-neighbor controller. Zhou et al. [17] propose the DALB algorithm. They select the most overloaded controller by comparing the load with an adaptive threshold, and then migrate the heaviest-loaded switch to a low-loaded controller. However, these studies only consider how to deal with overloaded controllers, but not ensure the stability of the distributed control plane, which may

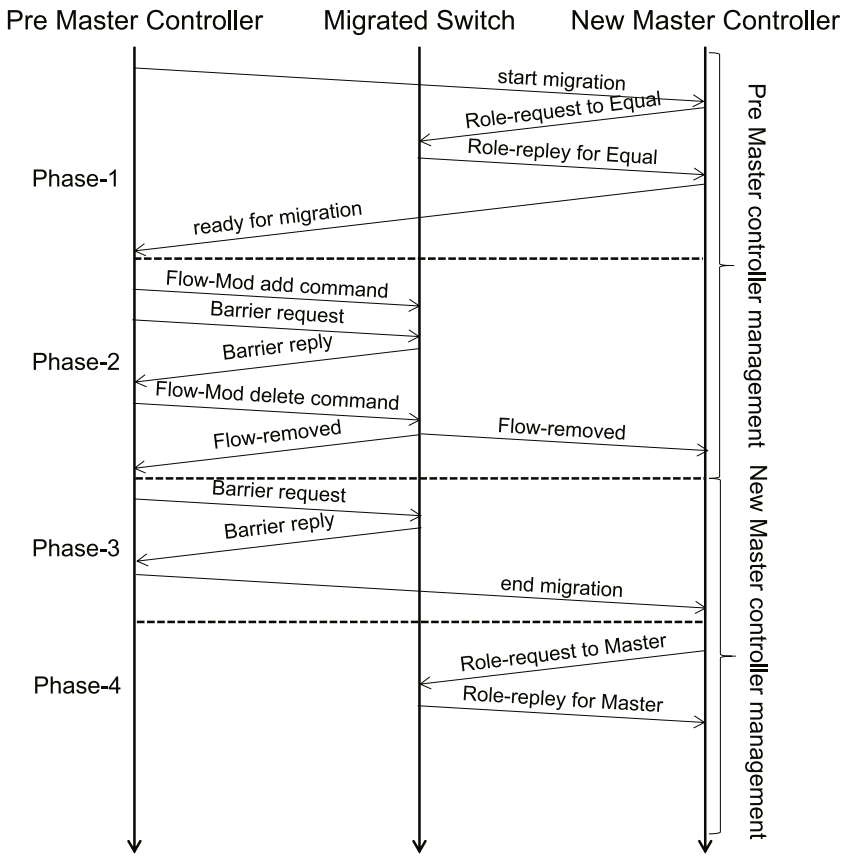


Fig. 2. Communication details in the migration phase.

lead to load imbalance again after one turn of load balancing, and may frequently trigger switch migration execution. Cheng et al. [18] propose the DHA strategy, which defines the load balancing problem among controllers as the Switch Migration Problem (SMP). They design a distributed algorithm that could run on every controller independently, to maximize the global system utilization. Cimorelli et al. [19] introduce game theory and propose a distributed algorithm, which grants the authorities of migrating decision to the switches to choose fitful controllers without communicating between each other. They suggest that controllers have no necessary to determine which switch to be selected to migrate, so that part of pressure in the control plane could be shifted into the data plane. In [20], Yu et al. propose the load informing strategy, that controllers periodically report their load information to others for detecting the overloaded controllers. These studies give kinds of distributed algorithms to access the scalability of the control plane which consisted of multiple controllers. However, they do not consider about the transfer cost during the switch migration phase.

Migration cost is an important issue who affects the network's performance, so it could not be negligible in the load balancing problem. Actually, in the early work ElastiCon [4], they have considered to select the nearest-neighbor controller to shift load for saving migration time. However, it is not the key point in this study. In [21], Liang et al. propose a mechanism to cluster controllers before shifting the load. They attempt to save the turns during the migration to reduce the network's fluctuation. However, their strategy increases the response time because of the clustering step. Wang et al. [22] propose the SMDM, which considers the distance from the migrated switch to the new master controller as a key issue who affects migration cost. On the basis of this knowledge, they define the migration efficiency, which makes a trade-off between migration cost and the load balance rate. To access less time for searching in the imbalance detection, Chen et al. [10] propose the FCLB, which uses the genetic algorithm to achieve fast convergence in the imbalance detecting phase. In [23], Zhou et al. formulate the switch migration as a 3-D earth mover's distance model to protect the strategically important controllers in the network, which could prevent saturation attacks. However, all of the above schemes trigger the switch migration when controllers cross their bottleneck, which may take a long migration time during the load balancing.

3. Problem formulation

In an SDN network, once a switch receives a packet, it would check its flow table for matching the forwarding rule. If the packet first arrives and there is no flow entry to match, the switch would encapsulate the packet into a PACKET_IN message and would send it to its master controller, after that, the master controller computes the forwarding rule and install a new flow entry in the switch by sending a FLOW_MOD message. The procedure is illustrated in Fig. 3. Generally, in all types of the message processing in a controller, the PACKET_IN message processing is regarded as the most significant load [24]. Thus, load imbalance among multiple controllers occurs mainly because of the enormous variety of PACKET_IN messages in the distributed control plane.

In the distributed control plane, each controller manages a set of switches. The requests from the switches would be aggregated at the processing queue of the connected controllers. With the increasing number of requests, the controller's CPU usage increases, and the response time increases as well. Besides, the response time increases as a non-linear trend with the controller's CPU usage. As shown in Fig. 4, the response time increases more rapidly when the controller's CPU usage approaches its bottleneck. However, when the number of requests crosses the controller's processing capacity, packet loss would happen and it harmfully affects the performance of the network.

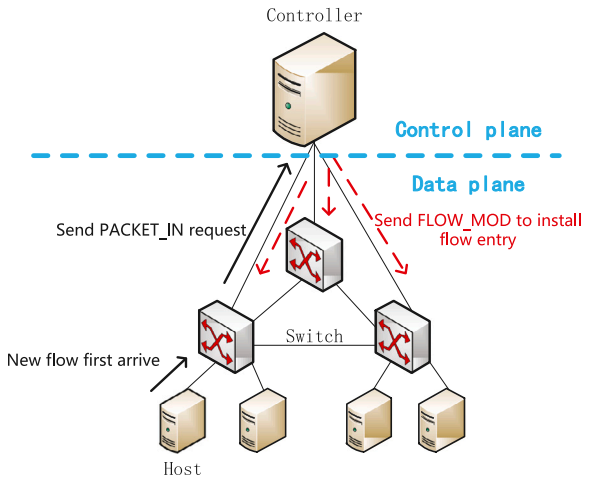


Fig. 3. The procedure of the new flow arrival in the SDN network.

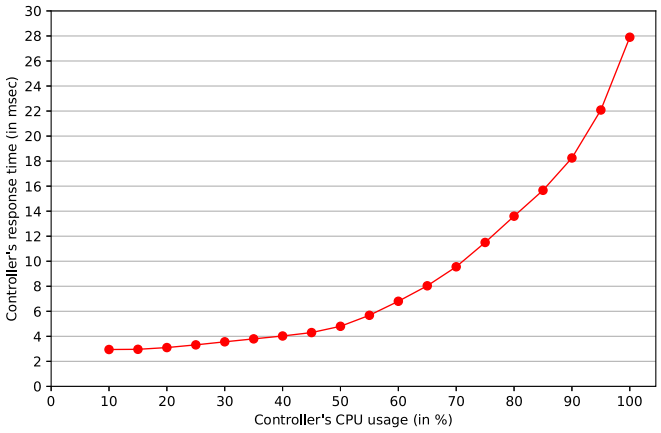


Fig. 4. The controller's response time increasing with the CPU usage.

Table 1
List of notations.

Notation	Description
s_i	ith switch
c_j	jth controller
α_j	Processing capacity of the jth controller
l_t^i	Switch ith load at time t
L_t^j	Controller jth load at time t
x_{ij}	Whether ith switch's master controller is jth controller
λ_1	Low-loaded state ratio between migration time and controller's load
λ_2	Heavy-loaded state ratio between migration time and controller's load
P_j	Profit for saved migration time of the jth controller
ω	Interval of the time slot
$\xi_{t+\omega}$	Total load difference of the control plane after load balancing

In this paper, we define a distributed SDN network G consisted of N controllers and M switches, which denoted as $C = \{c_1, c_2, \ldots, c_N\}$ and $S = \{s_1, s_2, \ldots, s_M\}$, respectively. Each controller's processing capacity in terms of the number of packets that it can handle per second is denoted as $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$. The rest of the used notations of this paper are summarized in Table 1.

We denote the ith switch's load in terms of the new requests it receives at time slot t as l_t^i, which means that it has to send such number of PACKET_IN messages to its master controller. That is to say, the jth controller's load in terms of the PACKET_IN messages it receives from its managed switches at time slot t could be denoted as L_t^j, which is given in Eq. (1).

$$L_t^j = \sum_{i=1}^{M} l_t^i * x_{ij} \tag{1}$$

Because the proposed scheme aims to migrate the switches in advance to reduce the migration cost during the load balancing, the focus of the controller's load should be pointed at the next time slot $t + \omega$. We temporarily define the prediction method as $f_{prediction}$, so the load of ith switch and jth controller at the next time slot could be denoted as Eqs. (2) and (3), respectively.

$$l_{t+\omega}^i = f_{prediction}(l_t^i) \tag{2}$$

$$L_{t+\omega}^j = f_{prediction}(L_t^j) \tag{3}$$

3.1. Profit definition

Two conditions would trigger the migration for load balancing. (1) When the predicted load of the controller crosses its processing capacity, which means that the packet loss would happen at the next time slot. In this condition, switch migration has to be executed immediately to prevent the packet loss in the future, which denoted as Eq. (4).

$$L_{t+\omega}^j > \alpha_j \tag{4}$$

(2) When the predicted load of the controller under its processing capacity, however, executes the switch migration can bring great profit, which is presented as saving a lot of migration time. As shown in Fig. 5, the migration time increases as a non-linear curve with the controller's response time, which is subject to the CPU usage. We set λ_1 and λ_2, which denote the ratio between migration time and controller's load, to discriminate the low-loaded state and heavy-loaded state of the controller. On the basis of this, the profit P_j could be defined as Eq. (5).

$$P_j = L_{t+\omega}^j * \lambda_2 - L_t^j * \lambda_1 \tag{5}$$

3.2. Load differences definition

After migration, we define the load of controller j as L_{new}^j, and have to measure the difference of each controller's load. Because the proposed scheme does not add or shutdown the controller during the migration phase, so the set of controllers C would not be changed and it is construct of N controllers. We define the average load of all controllers as Eq. (6).

$$Avg = \frac{\sum_{j=1}^{N} L_{new}^j}{N} \tag{6}$$

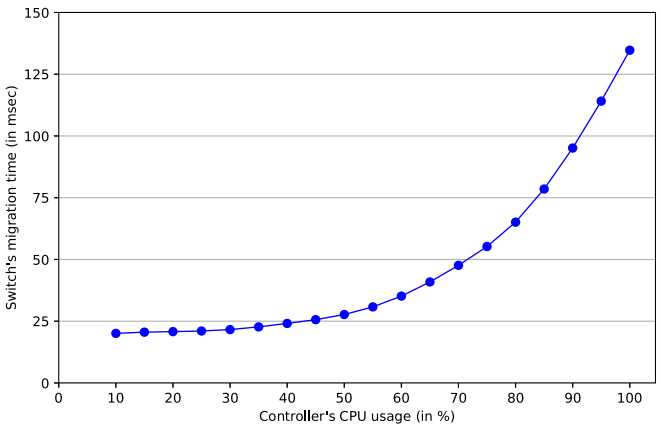


Fig. 5. The switch migration time increasing with the usage of its master controller's CPU.

Each controller's load difference is defined as Eq. (7).

$$d_j = |L^j_{new} - Avg| \tag{7}$$

Then, the metric of the total load difference among all controllers after load balancing can be defined as Eq. (8).

$$\xi_{t+\omega} = \frac{\sum_{j=1}^{N} d_j}{N} \tag{8}$$

3.3. Optimization problem

To conclude the problem formulation, the goal is to find the best migration policy which can minimize the total load difference $\xi_{t+\omega}$ after the load balancing in the data plane. Therefore, the optimization problem can be formulated as follows.

$$Minimize \quad \xi_{t+\omega} \tag{9}$$

subject to:

$$x_{ij} = 1 \tag{10}$$

$$L^j_{t+\omega} < \alpha_j \tag{11}$$

$$P_j - L^j_t * \lambda_1 > 0 \tag{12}$$

Eq. (10) ensures that the migrated switches are selected from the original master controller who has to be shifted load. As we expounded before, once the predicted load of a controller crosses its capacity, its switch has to be migrated immediately to prevent packet loss. This condition should not be included in the optimization problem; therefore, Eq. (11) ensures that the proposed scheme only focuses on the controllers whose predicted load does not cross their capacities.

Eq. (12) explains the guarantee of that the profit of shifting a controller overs the loss of ignoring a migration opportunity (if we ignore the prediction, the migration time at the low-loaded timing is going to be a loss). The profit P_j, which is denoted as the D-value between the prediction of the controller's load and the original load before the prediction, describes that how much time could be saved because of the prediction. On the contrary, the lost time during the load balancing is the time spent for migrating the switch at the time t. Therefore, the proposed scheme can ensure the profit of prediction overs the loss of migrating in advance.

4. Assessing profit of prediction scheme

In this section, we describe our framework of the Assessing Profit Of Prediction (APOP) scheme, which is illustrated in Fig. 6. It is constructed based on SDN structure, which aims to dynamically balance the load distribution among multiple controllers with the help of switch migration mechanism. In the control plane, it mainly consists of three modules: Monitor module, Load balancing trigger module, and Migration execution module.

4.1. Monitor module

The monitor module is set up for periodically collecting the statistic data, which is mainly the number of PACKET_IN messages from the controller's managed switches in each time slot. It stores the data in the database, and provides the historical information to the load balancing trigger module for analyzing whether to trigger switch migration.

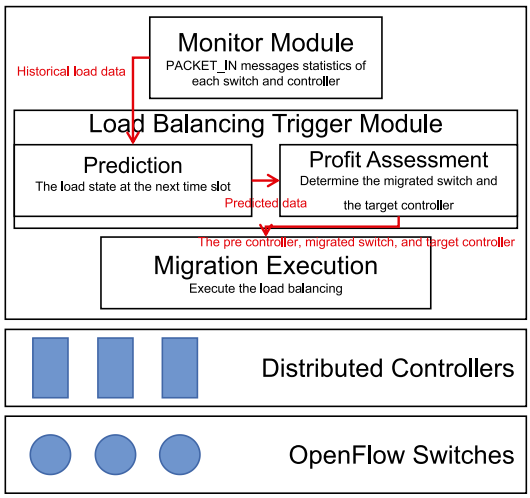


Fig. 6. The framework of the APOP scheme.

4.2. Load balancing trigger module

4.2.1. Prediction function

The function approximation is widely used in the prediction works. We choose the Taylor's formula, which is provided to describe the information in the vicinity of one point in a curve, for the prediction phase. Compared with the prediction methods based on machine learning or other long-term prediction methods, the Taylor's formula has such advantages as follows: (1) The Taylor's formula can straightly function on the generating curve, which consists of historical data, without any training phases; therefore, the mechanism does not depend on the dataset to train the model for prediction. (2) Because the proposed scheme does not care about the trend of the curve in the future, the focus is on the state of the next time slot for selecting the controllers facing overloaded. We have no need to predict a long-term trend, and the finitely spread Taylor's formula can perform a wonderful computational complexity. The conventional expression is given as $f(x) = \sum_{k=1}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$. The proposed scheme implements the Taylor's formula as the function $f_{prediction}$ to predict the number of PACKET_IN messages at time slot $t+\omega$ with the states in $t, t-\omega, t-2\omega...$ We present the function in Eq. (13).

$$\begin{aligned} l^i_{t+\omega} = {} & \frac{(l^i_t)}{0!} + \frac{(l^i_t)'}{1!}(t+\omega-t) \\ & + \frac{(l^i_t)''}{2!}(t+\omega-t)^2 + \cdots \\ & + \frac{(l^i_t)^{(n)}}{n!}(t+\omega-t)^n \\ & + \frac{1}{n-1}\sum_{k=1}^{n-1}(f_{prediction}(l^i_{t-k\omega}) - l^i_{t-k\omega}) \end{aligned} \tag{13}$$

At the end of Eq. (13), we implement $\frac{1}{n-1}\sum_{k=1}^{n-1}(f(x_k) - x_k)$ to correct the errors with the historical prediction. Considered to decrease the computational complexity, we do not spread the function infinitely. However, the one-step prediction would still perform accurately because of the continuous rising requests. The prediction algorithm's pseudo code is presented in Algorithm 1.

4.2.2. Profit assessment

We have three targets to achieve in this phase: (1) Select the controller who needs to be shifted load; (2) Select the new master controller to receive the shifted load, i.e., the migrated switch; (3)

Algorithm 1 Prediction Method

Input: l_t^i, $l_{t-\omega}^i$, $l_{t-2\omega}^i$, $l_{t-3\omega}^i$, $l_{t-4\omega}^i$
Output: $l_{t+\omega}^i$

Initialize:
d_1: The first derivative.
d_2: The second derivative.
$Correct$: The correction of the function.
$l_{t+\omega}^i$: The load prediction of ith switch.

$d_1 = \frac{l_t^i - l_{t-\omega}^i}{\omega}$
$d_2 = \frac{d_1 - (l_{t-\omega}^i - l_{t-2\omega}^i)/\omega}{\omega^2}$
$l_{t+\omega}^i = l_t^i + \frac{d_1}{1!}\omega + \frac{d_2}{2!}\omega^2$
for $k = 2$; $k < 4$; $k++$ **do**
    $d_1 = d_2$
    $d_2 = \frac{d_1 - (l_{t-k\omega}^i - l_{t-(k+1)\omega}^i)/\omega}{\omega^2}$
    $Correct = Correct + l_{t-(k-1)\omega}^i + \frac{d_1}{1!}\omega + \frac{d_2}{2!}\omega^2 - l_{t-(k-2)\omega}^i$
end for
$l_{t+\omega}^i = l_{t+\omega}^i + Correct$
return $l_{t+\omega}^i$

Select the switches who can minimize the loss during the migration. As we expounded before, there are two conditions to select a controller to shift load for load balancing: (1) Once the prediction $L_{t+\omega}^j > \alpha_j$, the controller has to shift load immediately; (2) If the prediction $L_{t+\omega}^j$ whose response time crosses the threshold between λ_1 and λ_2, the controller has to be assessed whether the profit is over the migration time that before it crosses the threshold, which is given in Eq. (12).

To discriminate the state of the controller's load between the low-loaded and the heavy-loaded, the proposed scheme has to determine a large difference between the value of λ_1 and λ_2. According to Fig. 5, we locate the breakpoint in the curve, and calculate the slopes from the start point to the breakpoint and from the breakpoint to the end point. We set the slope of the smooth segment as the λ_1 and the slope of the abrupt segment as the λ_2.

After determining the controller c_j who needs to be shifted load, the new master controller for receiving shifted switch would be selected from the low-loaded controllers and it should achieve Eq. (14). Then the proposed scheme greedily selects the switches who can ensure minimum loss, for minimizing the difference δ between the load of the previous controller and the new controller as shown in Eq. (15). Every selected switch has to satisfy Eq. (16) in each iteration. The complete profit assessment algorithm's pseudo code is given in Algorithm 2.

$$min\{L_{t+\omega}^j - L_t^j\} \tag{14}$$

$$\delta = |(L_t^{pre} - l_t^{migrated}) - (L_t^{new} + l_t^{migrated})| \tag{15}$$

$$min\{l_{t+\omega}^i - l_t^i\} \tag{16}$$

4.3. Migration execution module

The migration execution module is in charge of executing the migration decision made from Algorithm 2. Once the controller who needs to shift load c_j, $s_{migrated}$ and c_{new} are determined, the controller c_j would send a ROLE_TRANSFER message to the switch $s_{migrated}$. Then the switch $s_{migrated}$ communicate with its previous master controller c_j and its new master controller c_{new}. The task of role transfer would re-map the controller-switch mapping and the network G would be updated.

Algorithm 2 Profit Assessment

Input: c_j
Output: c_{new}, $s_{migrated}[]$

Initialize:
c_{new}: The controller to receive the shifted load.
$s_{migrated}[]$: The set of switches to be migrated.
δ: The difference between previous controller and new controller.

Phase 1:
if $L_{t+\omega}^j > \alpha_j$ **then**
    Go to **Phase 2**
end if
if $P_j - L_t^j * \lambda_1 > 0$ **then**
    Go to **Phase 2**
end if
return **null**

Phase 2:
for c_k in C && k != j **do**
    if c_k in the low-loaded state && $min\{L_{t+\omega}^k - L_t^k\}$ **then**
        $c_{new} = c_k$
        $\delta = L_t^j - L_t^{new}$
    end if
end for
for $x_{ij} = 1$ **do**
    if $min\{l_{t+\omega}^i - l_t^i\}$ && $|(L_t^j - l_t^i) - (L_t^{new} + l_t^i)| < \delta$ **then**
        Add s_i to $s_{migrated}[]$
        Delete s_i from S
        $\delta = |(L_t^j - l_t^i) - (L_t^{new} + l_t^i)|$
        $L_t^j = L_t^j - l_t^i$
        $L_t^{new} = L_t^{new} + l_t^i$
    end if
end for
return $s_{migrated}[]$, c_{new}

5. Performance evaluation

In this section, we perform our experiments and compare our scheme with our previous work. We implement our scheme in a simulated environment, which is constructed in Ubuntu 16.04 LTS, with the hardware of 4 cores CPU and 8GB memory. We use the Floodlight controller [25] to establish 5 controllers. The topology we use is BT Asia-Pacific Topology from the Topology Zoo [26], and we run it in the Mininet platform [27]. To measure the processing capacity of the controller, we use the SDN controller benchmark CBench [28] to imitate the BT Asia-Pacific Topology, i.e., 20 nodes. As shown in Fig. 7, the processing capacity of the controller is around 40000 packets per second, and it catches its bottleneck with the CPU usage arrived 100 percent.

To test the performance, we singly simulate our scheme to observe the effect on multiple-controllers load balancing. As shown in Fig. 8, we use iPerf [29] to generate OpenFLow requests into the controller c_1, c_2 and c_5. The packet arrival crosses the bottleneck of the controller c_1 and c_2 at about 150th second, while the load of c_5 increases for a while but not catch its bottleneck. When the APOP algorithm runs, as shown in Fig. 9, the overloaded controllers successfully shift load before they catch their bottleneck. Because of the bad profit assessment of c_5, the controller denies shifting its load during the load balancing. We set the time slot ω as 1 s, which is fine-granted enough for the overload detection in the network.

We introduce the load balancing ratio [30] to evaluate the load distribution after the switch migration. The *ratio* is denoted as Eq. (17),

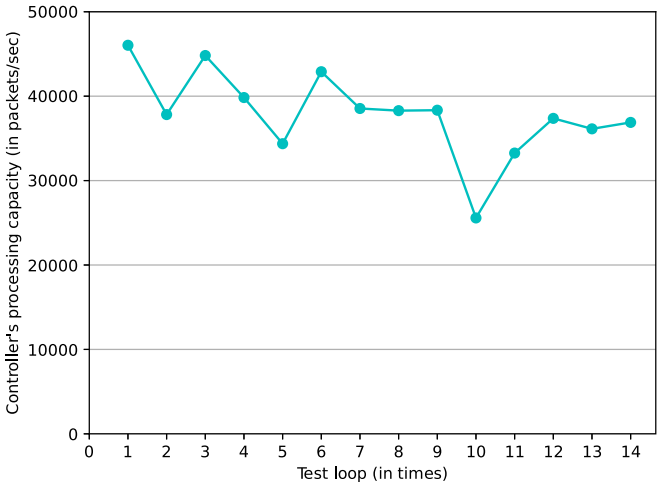


Fig. 7. The processing capacity of the controller when it catches its bottleneck.

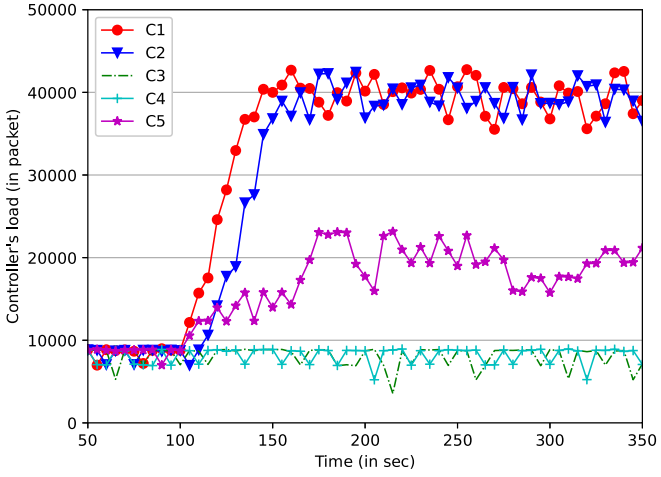


Fig. 8. The controllers load distribution without any load balancing mechanism.

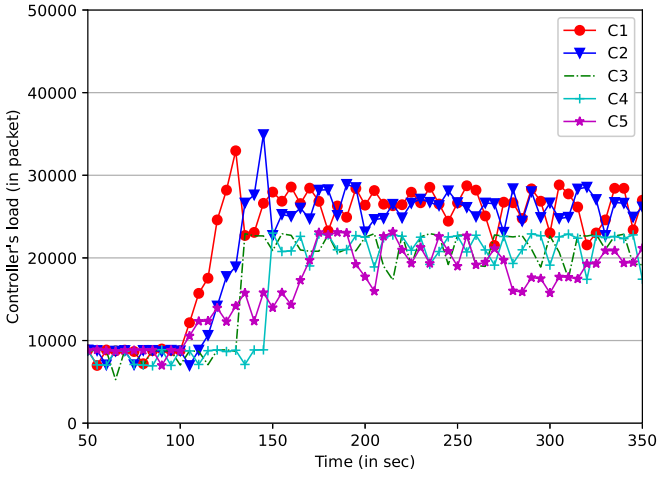


Fig. 9. The controllers load distribution with the APOP algorithm.

and the performance is illustrated in Fig. 10. Because of the switch selection mechanism in the APOP, great load balancing ratio is achieved

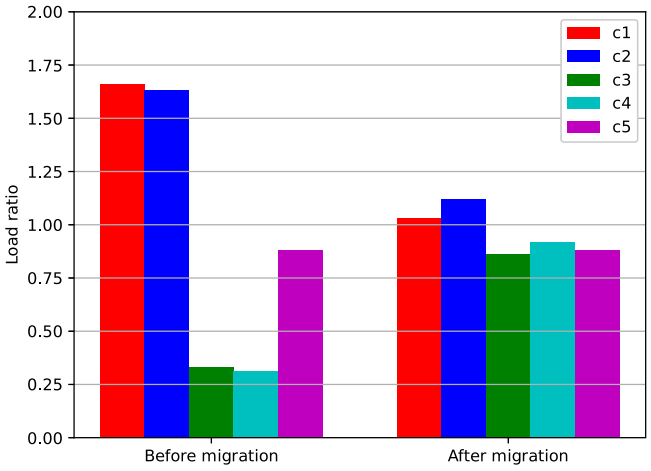


Fig. 10. The load ratio before and after the load balancing.

after the load balancing.

$$ratio = \frac{avg(Load_{c_j})}{\frac{1}{N}\sum_{k=1}^{N} Load_{c_k}} \tag{17}$$

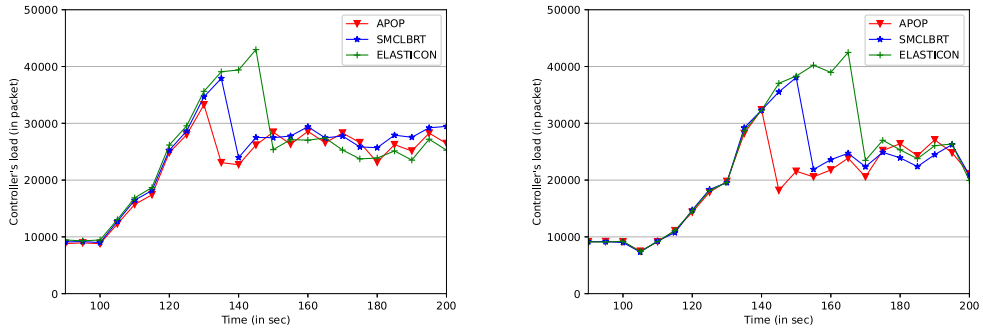
To intuitively express the performance of our scheme, we compare our strategy with ELASTICON [4] and SMCLBRT [30]. In the ELASTICON, Dixit et al. firstly use switch migration mechanism with OpenFlow v.1.3 to balance load among multiple controllers. They trigger load balancing with a stated threshold and migrate the switch to the nearest neighbor-controller to save migration cost. In the SMCLBRT, Cui et al. trigger the switch migration based on the response time of a controller. We simulate the same experimental test environment for these three schemes and run each simulation for 5 min. The comparison of the three schemes is shown in Fig. 11.

We illustrate the performance of the controller $c1$ and $c2$ in Fig. 11, because of the statically configured threshold, the ELASTICON triggers the migration when the controller has been overloaded, which may waste a long time during the migration process. The SMCLBRT triggers the migration based on the response time of the overloaded controller; therefore, they can detect the anomalous response time and shift the load before the controller accessed its bottleneck. With the prediction mechanism, the APOP can save much migration cost during the load balancing.

The benefit brought by APOP is that we reduce the migration time during the load balancing, which decrease the fluctuation during the flow peak timing. Besides, because of the migration execution is below the bottleneck of the facing overloaded controller, the response time between the controller and its switches is saved a lot as well. The average response time during the migration and the average migration time are illustrated in Figs. 12 and 13, respectively.

To illustrate in details, the APOP can execute the migration 14% earlier than the ELASTICON, and 5% earlier than the SMCLBRT. In the migration time economy, the APOP reduces 48% and 12% time over the ELASTICON and the SMCLBRT, respectively. Besides, we reduce 40% response time compared with the ELASTICON, and 8% to the SMCLBRT as well. However, flows in the network exists temporal burst; therefore, once the prediction operates a mistake output, the migration would become a loss in the load balancing. Based on this consideration, we measure the global overhead in the different proportion of the temporal burst. The measurement of the global profit against loss (with APOP) is based on Eq. (18), and the measurement of the global loss (without APOP) is based on Eq. (19), while the κ represents the proportion of the temporal burst. The calculated result is illustrated in Fig. 14.

$$(avg(P) * (1-\kappa) - avg(L_t * \lambda_1) * \kappa) * 100\% \tag{18}$$



(a) Controller c1 (b) Controller c2

Fig. 11. Comparison of migration timing among ELASTICON, SMCLBRT and APOP.

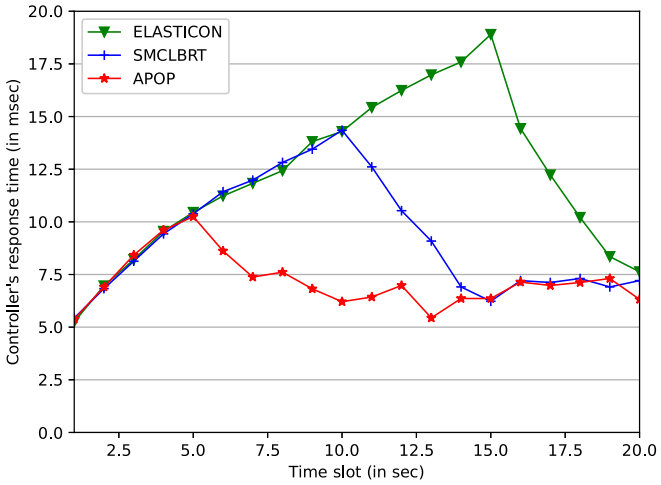


Fig. 12. The average response time of the controllers during the load balancing.

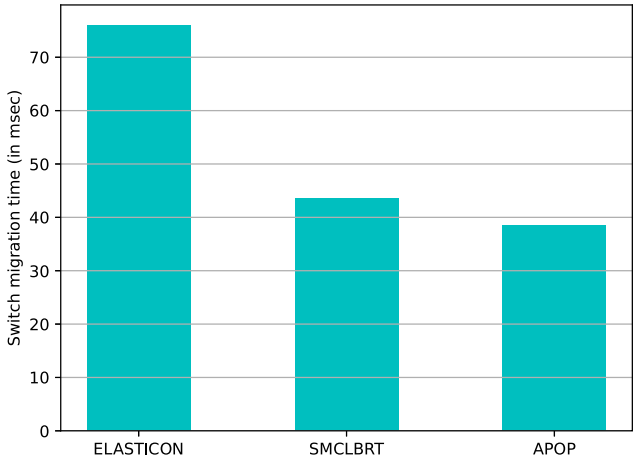


Fig. 13. The average switch migration time of the ELASTICON, SMCLBRT and APOP.

$$avg(L_{t+\omega} * \lambda_2) * (1 - \kappa) * 100\% \tag{19}$$

As can be seen in Fig. 14, although the temporal burst of flows might affect the prediction, with APOP, we can achieve positive profit in global; however, without APOP, the global loss is intolerable.

Based on the afore-presented simulations' result, a discussion about the strategies designed in our scheme is given in the following: (1) Because the capricious flows in the network are hard to predict, we choose the short-term prediction to avoid more harmful influence from

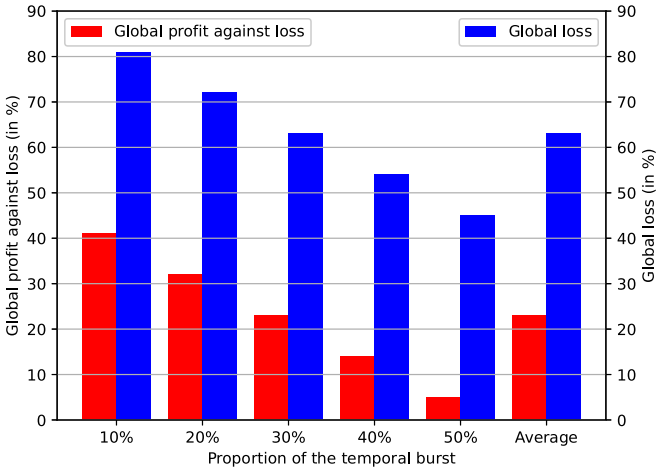


Fig. 14. The global overhead in the different proportion of the temporal burst.

the mistake result predicted by the long-term prediction. (2) With the profit assessment mechanism provided by APOP, the result predicted by the short-term mechanism, the Taylor's formula, could be efficiently evaluated to avoid low-profit migrations. (3) The profit assessment could promise a global positive profit against the mistake prediction influenced by the low-ratio of the temporal burst. The great performance of our APOP scheme proves that we can commendably increase the resource utilization and reduce the fluctuation occurs by the switch migration during the load balancing.

6. Conclusion

In this paper, we focused on investigating the problem of migration cost in control plane load balancing. Inspired by the previous works, we found that migration time is related to the load of the controller. The migration cost obviously increased with the controller closing to its handling bottleneck. On the basis of this, we considered predicting the behavior of the controller's load related to its managed switches, and proposed the APOP scheme for executing the migration phase in advance. The APOP scheme aimed to predict the flow traffic arriving state by the method of function approximation and assessed the profit of prediction for migration economy. We chose Taylor's formula, which showed a great effect in the prediction. The switch selection with fluctuation prevention and load balance insurance was designed in our scheme to maximize the resource utilization and reduce the harmful effects during the migration phase. Our scheme accessed the scalability and reliability of the SDN control plane. As presented in the simulation experiments, APOP achieved earlier migration execution and saved the

migration time and response time better than our previous work, which proved to be an effective strategy to approach distributed control plane load balancing in SDN.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

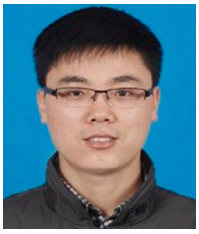
The work was supported by the National Natural Science Foundation of China (No. U1936220, No. 61702005, No. 61872001), the Special Fund for Key Program of Science and Technology of Anhui Province, China (No. 18030901027), the Open Fund for Discipline Construction, Institute of Physical Science and Information Technology, Anhui University, China. The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

References

[1] B. Lantz, B. Heller, 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, ACM, 2010, p. 19.
[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.
[3] A. Tootoonchian, Y. Ganjali, Hyperflow: A distributed control plane for openflow, in: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, vol. 3, 2010.
[4] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, Towards an elastic distributed SDN controller, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 7–12.
[5] A. Krishnamurthy, S.P. Chandrabose, A. Gember-Jacobson, Pratyaastha: An efficient elastic distributed SDN control plane, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, 2014, pp. 133–138.
[6] D.-A. Dasilva, L. Liu, N. Bessis, Y. Zhan, Enabling green it through building a virtual desktop infrastructure, in: 2012 Eighth International Conference on Semantics, Knowledge and Grids, IEEE, 2012, pp. 32–38.
[7] J. Panneerselvam, L. Liu, N. Antonopoulos, An approach to optimise resource provision with energy-awareness in datacentres by combating task heterogeneity, IEEE Trans. Emerg. Top. Comput. (2018).
[8] Y. Carlinet, N. Perrot, Energy-efficient load balancing in a SDN-based data-center network, in: 2016 17th International Telecommunications Network Strategy and Planning Symposium, Networks, IEEE, 2016, pp. 138–143.
[9] E.L. Fernandes, C.E. Rothenberg, Openflow 1.3 software switch, Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC (2014) 1021–1028.
[10] Y.-T. Chen, C.-Y. Li, K. Wang, A fast converging mechanism for load balancing among SDN multiple controllers, in: 2018 IEEE Symposium on Computers and Communications, ISCC, IEEE, 2018, pp. 00682–00687.
[11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, NOX: towards an operating system for networks, ACM SIGCOMM Comput. Commun. Rev. 38 (3) (2008) 105–110.
[12] L. Cui, F.R. Yu, Q. Yan, When big data meets software-defined networking: SDN for big data and big data for SDN, IEEE Netw. 30 (1) (2016) 58–65.
[13] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al., Onix: A distributed control platform for large-scale production networks., in: OSDI, vol. 10, 2010, pp. 1–6.
[14] D. Levin, A. Wundsam, B. Heller, N. Handigol, A. Feldmann, Logically centralized?: state distribution trade-offs in software defined networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, 2012, pp. 1–6.
[15] D. Erickson, The beacon openflow controller, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, 2013, pp. 13–18.
[16] S. Hassas Yeganeh, 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, ACM, 2012, pp. 19–24.
[17] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, M. Zhu, A load balancing strategy of sdn controller based on distributed decision, in: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2014, pp. 851–856.
[18] G. Cheng, H. Chen, Z. Wang, S. Chen, DHA: Distributed decisions on the switch migration toward a scalable SDN control plane, in: 2015 IFIP Networking Conference, IFIP Networking, IEEE, 2015, pp. 1–9.
[19] F. Cimorelli, F.D. Priscoli, A. Pietrabissa, L.R. Celsi, V. Suraci, L. Zuccaro, A distributed load balancing algorithm for the control plane in software defined networking, in: 2016 24th Mediterranean Conference on Control and Automation, MED, IEEE, 2016, pp. 1033–1040.
[20] J. Yu, Y. Wang, K. Pei, S. Zhang, J. Li, A load balancing mechanism for multiple SDN controllers based on load informing strategy, in: 2016 18th Asia-Pacific Network Operations and Management Symposium, APNOMS, IEEE, 2016, pp. 1–4.
[21] C. Liang, R. Kawashima, H. Matsuo, Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers, in: 2014 Second International Symposium on Computing and Networking, IEEE, 2014, pp. 171–177.
[22] C. Wang, B. Hu, S. Chen, D. Li, B. Liu, A switch migration-based decision-making scheme for balancing load in SDN, IEEE Access 5 (2017) 4537–4544.
[23] Y. Zhou, K. Zheng, W. Ni, R.P. Liu, Elastic switch migration for control plane load balancing in SDN, IEEE Access 6 (2018) 3909–3919.
[24] A. Filali, S. Cherkaoui, A. Kobbane, Prediction-based switch migration scheduling for SDN load balancing, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, IEEE, 2019, pp. 1–6.
[25] S. Floodlight, OpenFlow controller, Web: https://github.com/floodlight/floodlight 2 (1).
[26] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The internet topology zoo, IEEE J. Sel. Areas Commun. 29 (9) (2011) 1765–1775.
[27] R.L.S. De Oliveira, C.M. Schweitzer, A.A. Shinoda, L.R. Prete, Using mininet for emulation and prototyping software-defined networks, in: 2014 IEEE Colombian Conference on Communications and Computing, COLCOM, IEEE, 2014, pp. 1–6.
[28] R. Sherwood, Y. Kok-Kiong, Cbench: an open-flow controller benchmarker, URL http://archive.openflow.org/wk/index.php/Oflops.
[29] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, K. Gibbs, Iperf: The tcp/udp bandwidth measurement tool. htt p, dast. nlanr. net/Projects, 38 (2005).
[30] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, A load-balancing mechanism for distributed SDN control plane using response time, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1197–1206.

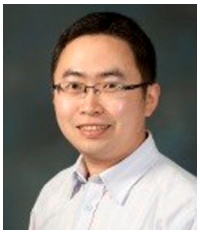
Hong Zhong was born in Anhui Province, China, in 1965. She received her Ph.D. degree in computer science from University of Science and Technology of China in 2005. She is currently a professor and Ph.D. supervisor of the School of Computer Science and Technology at Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). She has over 120 scientific publications in reputable journals (e.g. IEEE Journal on Selected Areas in Communications, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Multimedia, IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Network and Service Management and IEEE Transactions on Big Data), academic books and international conferences.

Jinpeng Fan is now a research student in the School of Computer Science and Technology, Anhui University. His research focuses on load balancing in software-defined networking (SDN).



Jie Cui was born in Henan Province, China, in 1980. He received his Ph.D. degree in University of Science and Technology of China in 2012. He is currently a professor and Ph.D. supervisor of the School of Computer Science and Technology at Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has over 100 scientific publications in reputable journals (e.g. IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Multimedia, IEEE Transactions on Network and Service Management, IEEE Transactions on Emerging Topics in Computing and IEEE Transactions on Circuits and Systems), academic books and international conferences.

Yan Xu is currently an associate professor of School of Computer Science and Technology at Anhui University. She received the BS and MS degrees from Shandong University in 2004 and 2007, respectively, and the Ph.D. degree from University of Science and Technology of China in 2015. Her research interests include information security and applied cryptography.



Lu Liu is the Professor of Informatics and Head of School of Informatics in the University of Leicester, UK. Prof Liu received the Ph.D. degree from University of Surrey, UK and M.Sc. in Data Communication Systems from Brunel University, UK. Prof Liu's research interests are in areas of cloud computing, service computing, computer networks and peer-to-peer networking. He is a Fellow of British Computer Society (BCS)