A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time

Jie Cui, Qinghe Lu, Hong Zhong, Miaomiao Tian, and Lu Liu, *Member, IEEE*

***Abstract*—Software-defined networking (SDN) has become a popular paradigm for managing large-scale networks including cloud servers and data centers because of its advantages of centralized management and programmability. The issues of scalability and reliability that a single centralized controller suffers makes distributed controller architectures emerge. One key limitation of distributed controllers is the statically configured switch-controller mapping, easily causing uneven load distribution among controllers. Previous works have proposed load-balancing methods with switch migration to address this issue. However, the higher-load controller is always directly considered as the overloaded controller that need to shift its load to other controllers, even if it has no response time delay. The pursuit of absolute load-balancing effect can also result in frequent network delays and service interruptions. Additionally, if there are several overloaded controllers, just one controller with the maximum load can be addressed within a single load-balancing operation, reducing load-balancing efficiency. To address these problems, we propose SMCLBRT, a load-balancing strategy of multiple SDN controllers based on response time, considering the changing features of real-time response times versus controller loads. By selecting the appropriate response time threshold and dealing with multiple overloading controllers simultaneously, it can well solve load-balancing problem in SDN control plane with multiple overloaded controllers. Simulation experiments exhibit the effectiveness of our scheme.**

***Index Terms*—Software-defined network, load-balancing, multiple controllers, response time, switch migration.**

I. Introduction

With the rapid development of Internet and innovations in technology, SDN is revolutionizing the networking industry by enabling programmability, easier management, and faster innovation [1]. Many benefits are made possible by the its centralized control plane architecture, which allows the network to be programmed by the application and controlled by one central entity. However, like any other centralized system, the single centralized controller faces the issues of scalability and reliability. For large-scale networking scenarios consisting of several hundreds of thousands of servers, a single centralized controller is difficult to manage these network topologies. Hence, the next feasible step is building a logically centralized, but physically distributed control plane, which can benefit from the scalability and reliability of the distributed architecture, while preserving the logical control of the SDN [2].

Some previous works have explored the architecture of distributed SDN controllers [3]–[5]. However, the mapping between a switch and a controller is statically configured, making it difficult for the control plane to adapt to traffic load variations [1]. Therefore, the distributed architecture introduces a new challenge: load-rebalancing the controllers when uneven load distributions occur [6].

The concept of multiple-controller SDNs has also been put forward. For example, OpenFlow v.1.3, an SDN southbound interface protocol, can functionally address controller failover and load-balancing [7]. An SDN controller has three roles: master, slave, and equal. A controller may change its role at any time, but only the controller in master state has full access to the switch. It allows dynamic switch migration can be an effective and easy approach to load-balancing among distributed controllers. If seriously uneven load distributions occur among controllers, load-balancing with switch migration should be done. Additionally, if the controller pool is grown or shrunk as per traffic conditions, load-shifting across controllers should also be conducted.

With dynamic switch migration, the performance and scalability of distributed controllers may be effectively increased. Dixit *et al.* [8] first propose an efficient protocol to enable switch migration across multiple controllers, called ElastiCon. In their method, the nearest-neighbor controller was selected as the immigration controller for receiving load-shifting. In [9], a dynamic and adaptive load-balancing (DALB) algorithm was proposed. The maximum load controller was selected by comparing its load value to an adaptive load collection threshold. Then, a heavy-load switch is chosen to migrate to a low-load controller. Many methods have been proposed to solve the election issue of outmigration and immigration controllers [2], [10]–[12]. During the process of load-balancing, we usually need to judge when the controllers' loads are unbalanced and make an effective decision about how to shift loads.

Manuscript received April 15, 2018; revised August 12, 2018 and September 25, 2018; accepted September 28, 2018. Date of publication October 16, 2018; date of current version December 10, 2018. The work was supported by the National Natural Science Foundation of China (No. 61572001, No. 61872001, No. 61702005), the Natural Science Foundation of Anhui Province (No. 1708085QF136), the Open Fund for Discipline Construction Institute of Physical Science and Information Technology, Anhui University and the Excellent Talent Project of Anhui University. The associate editor coordinating the review of this paper and approving it for publication was K. Shiomoto. *(Corresponding author: Hong Zhong.)*

J. Cui, Q. Lu, H. Zhong, and M. Tian are with the School of Computer Science and Technology, Anhui University, Hefei 230039, China, also with the Anhui Provincial Key Laboratory of Network and Information Security, Anhui Normal University, Wuhu 241002, China, and also with the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: zhongh@ahu.edu.cn).

L. Liu is with the School of Electronics, Computing and Mathematics, University of Derby, Derby DE22 1GB, U.K. (e-mail: l.liu@derby.ac.uk).

Digital Object Identifier 10.1109/TNSM.2018.2876369

Therefore, a threshold value is necessary for selecting overloaded controllers. Moreover, switch migration might be frequently executed if we pursue absolute load-balancing across multiple controllers. Migration costs are unavoidable, and the message exchanging costs cannot be negligible during switch migration. In [13], a switch migration and decision-making (SMDM) scheme was presented to get an optimum decision for the tradeoff between migration costs and the load-balancing rate. The largest load controller over the load threshold value is always directly set as the overloaded controller in near-current load-balancing methods without determining whether the controller load is close to its bottleneck. It invariably leads to frequent load-shifting. Migrating a switch from one controller to another can cause disruption to ongoing flows, which can severely impact the various applications in the datacenter.

To avoid frequently unnecessary operations, we take full advantage of the changing features of response time and controller load and propose a novel load-balancing scheme, the SDN multiple controller load-balancing strategy based on response time (SMCLBRT). In case the controller has a normal response time for every flow request, the process of migrating switches is unnecessary and unwanted. If its response time has increased significantly, we assess the controller load as close to its capability bottleneck. Thus, an effective load-balancing method should be triggered. Additionally, if there are several overloaded controllers, only the controller with the maximum load can be handled in a single operation, which could reduce efficiency. To address this problem, we design a method that can handle multiple overloaded controllers with just one operation.

In this paper, we focus on designing a logical load-balancing strategy via switch migration. To the best of our knowledge, it is the first work to measure controller loads and perform load-balancing operations by using response time. In summary, the main contributions of this paper are as follows.

- We validate the changing features of response time versus controller load, which can accurately judge the growth trend of controller's response time as load increase.
- We use the response time to measure the controller's load, and we get an appropriate response time threshold to achieve a better detection of overloading controllers so that they can be handled ahead of time.
- To improve load-balancing efficiency, we propose a novel switch migration algorithm to handle several overloaded controllers in a single load-balancing operation if there is more than one controller overload.

We have structured the paper as follows. Section II discusses related works. In Section III, we detail the design approach of SMCLBRT. Then, Section IV illustrates an implementation environment and Section V discusses performance evaluation from four aspects. Finally, we conclude with our findings and future work in Section VI.

II. Related Works

This section reviews recent research achievements of distributed controllers and load-balancing approaches that support the representation of our research background and its theoretical foundation. We address three aspects, including the SDN load-balancing method, SDN-distributed controllers, and load-balancing in the SDN control plane. The first aspect explains the traditional load-balancing method and the SDN-based load-balancing method. Then we detail the distributed architecture of the SDN control plane and its challenges. The last aspect shows some research progress of load-balancing in the SDN control plane.

A. Load Balance With SDN

In traditional networks, load-balancing implies two meanings. First, the large number of concurrent accesses or network packets should be allocated to multiple node devices to reduce user response times. Second, a single heavy-load operation should be assigned to multiple node devices for parallel processing to significantly improve overall performance [14]. The main application scenarios of load-balancing include server and link load-balancing. Traditional network load-balancing technology includes hardware and software implementation. For example, NGINX [15] and HAProxy [16] are based on the forwarding of a four-layer interaction technology or an agent of the seven-layer protocol. However, in traditional networks, it is not easy to realize the load-balancing strategy globally, because it is difficult to obtain the whole network status. Additionally, the traditional load-balancing method is also difficult to adapt to changes and adjustments of the network state [17].

As a novel network paradigm, SDN can outsource the control permissions of programmable network switches to a software controller. The SDN controller manages all the network devices to ensure the intelligent network. It allocates network resources dynamically and flexibly, according to different users' needs and global network topology. For the data layer, it communicates with the network infrastructure through a standard southbound protocol; For the application layer, it provides control of the network resources through an open northbound interface. The most fundamental task of an SDN controller is the correct implementation of the network policy, such as the intended load-balancing behavior. With the programmability and flexibility of SDNs, a controller is usually used as a load balancer in traditional networks for traffic-scheduling and load-management. The way of software customizing approach can reduce the requirements of hardware equipment. Therefore, SDN can be used to achieve link and server load-balancing in traditional networks. A fuzzy synthetic evaluation mechanism [18] is a path load-balancing solution based on SDN which effectively balances traffic and avoids unexpected breakdowns caused by link failure. It increases the utilization and reliability of network paths. Reference [19] proposed an OpenFlow-based dynamic load-balancing strategy for datacenter networks, and it enables the efficient use of the network resources capacity. Load-balancing based on server response time [20] is also an effective server load-balancing scheme. It achieves a better load-balancing effect in comparison with the traditional Round Robin and Random schemes.

B. SDN Multiple Controllers

In some network cases, a centralized SDN controller (*e.g.,* NOX [21]) can be well-used in traditional network traffic control and management. However, as the management and control center of network flows, the performance of the SDN controller is also affected by its processing load. With the deployment of SDN in application scenarios including cloud computing and big data, the massive requests of unmatched traffic to the single SDN controller can make the safety and performance of the SDN control plane a potential bottleneck of the whole network [22], [23].

The distributed SDN controller architecture can be a good solution for limiting the control layer processing capability of centralized controllers, including HyperFlow [3] and Onix [4]. Beacon [24] improves the performance of each controller in the control plane with its multithreaded designs. Kandoo [25] implements a distributed control plane through a cluster of SDN nodes to improve the scalability and reliability of the control plane. Reference [26] designed a hybrid hierarchical control plane for flow-based large-scale SDN networks to improve the scalability of the SDN control plane. These methods effectively solve the performance bottlenecks of a single centralized controller. However, multiple SDN controllers may bring some new challenges, including load imbalances between multiple controllers caused by the uneven distribution of network traffic and the problem of control information synchronization.

C. Load Balance in SDN

In the SDN distribution control plane, the mapping between OpenFlow switches and SDN controllers is static. With the dynamic change and instantaneity of network traffic, there is an uneven load distribution among controllers [8]. The static switch-controller mapping causes load imbalances and reduces the overall resource utilization, leading to sub-optimal performance [13].

In OpenFlow 1.3 protocol [7], a switch can be connected to multiple SDN controllers, allowing its load to be migrated between different controllers. Recent works focused on load-balancing methods of distributed controllers using switch migration. A switch migration protocol for SDN multiple controller load-balancing was first proposed in [8]. 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. Zhou *et al.* [9] proposed a dynamic algorithm (*i.e.,* DALB) totally based on a distributed architecture with an adaptive load threshold. Liang *et al.* [10] proposed a dynamic load-rebalancing method based on switch migration for clustered controllers. This method also supports controller failover and avoids the single point of failure problem. However, it significantly increases the response time of the control layer. Cheng *et al.* [11] designed the maximizing resource utilization migration algorithm (MUMA). When the workload distribution is uneven, the overloaded controller randomly selects a switch to migrate. However, the overloaded controller still could be overloaded after migration. Yu *et al.* [27] proposed a load-balancing mechanism based on a load informing strategy. Each controller periodically reports its load information to other controllers so that an overloaded controller no longer collects all other controllers' load information before making local decisions. The above studies do not consider the multiple overloaded controllers and have no fine-grained judgment of controller load.

In more recent works focusing on SDN multiple controller load-balancing, [13], [28] have pursued the efficiency of switch migration and decision-making. SMDM [13] 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, it also might cause a new load imbalance after migration because of its long load-balancing time. Additionally, it has a sub-optimal load-balancing performance. Reference [28] proposed a load-balancing scheme based on switch groups. It selects a migrating switch group depending on excess workload, effectively reducing the number of decisions. However, it increases the number of migrated switches and pursues an overall load-balancing rate, which brings extra unnecessary migration costs.

III. THE DESIGN OF SMCLBRT

Usually, in an SDN network, once a switch receives a packet matching a non-corresponding flow table entry, it needs to encapsulate the header of the packet into a PACKET_IN message and send it to its master controller for routing and flow-table entry installation. The processing of PACKET_IN messages is generally regarded as the most prominent part of the controller load [29], [30]. Thus, the enormous distributional difference of PACKET_IN messages might lead to unbalanced workloads among multiple controllers. Some controllers may reach their performance bottlenecks with the significantly increasing number of response delays, whereas other controllers are under normal loads or in an idle state.

In this section, we propose the SMCLBRT scheme for balancing in the distributed SDN control plane with multiple overloaded controllers, making a fine-grained judgment on overloaded controllers based on response time. As with other switch migration schemes, load-balancing in the SDN control plane includes three phases: the measurement of the load imbalance of controllers; the decision-making of migration plans with the selection of overloaded controllers, main load switches, and immigration controllers; and the implementation of the migration plans to shift loads of overloaded controllers.

The system model is shown in Fig. 1. We consider an SDN network *G* consisting of *N* controllers $C = \{C_1, C_2, \ldots, C_N\}$ and *K* switches $S = \{S_1, S_2, \ldots, S_K\}$. These controllers have the same performances and divide the network into *N* domains. In each domain, the network traffic and controller workload are dynamically changed. Let $Q_{C_i} \in S$ denote the switch set managed by a master controller C_i. In the rest of this section, we explain the detailed design of SMCLBRT from four parts.

A. Response Time Acquisition and Load Measurement

In our scheme, a new module is added for periodic statistics and the calculation of response time at a time interval, *T*.

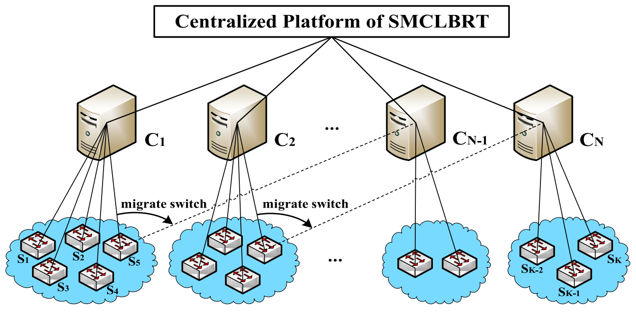


Fig. 1. System model.

We let S_k and T_n to represent the k^{th} switch and the n^{th} period, respectively. Let t_{arrive} denote the PACKET_IN message arrive time and t_{reply} denote the time that the controller replies a PACKET_OUT message or a FLOW_MOD message to the corresponding switch. Thus, we easily get the response time to the single PACKET_IN request, as shown in the Eq. (1).

$$t_{response} = t_{reply} - t_{arrive} \tag{1}$$

Meanwhile, we record the number of request messages from S_k, in T_n (denoted as $f_{S_k T_n}$), and use it to indicate the workload brought by S_k in T_n (denoted as $Load_{S_k T_n}$). Therefore, the sum-load of these switches in the set Q_{C_i} can be used to represent the total number of received request messages $f_{C_i T_n}$ or the workload $Load_{C_i T_n}$ of the controller C_i in the n^{th} period, as shown in the Eq. (2).

$$Load_{C_i T_n} = \sum_{S_k \in Q_{C_i}} f_{S_k T_n} \tag{2}$$

We denote the response time to the whole PACKET_IN requests sent from S_k in T_n as $t_{S_k\_response}$. Therefore, the total response time of all the accepted messages by C_i in T_n can be easily obtained. After getting these data, we can directly obtain the average response time to a single PACKET_IN message with a time interval *T*.

$$t_{C_i T_n} = \frac{\sum\limits_{S_k \in Q_{C_i}} t_{S_k\_response}}{Load_{C_i T_n}}. \tag{3}$$

B. The Appropriate Threshold Based on Response Time

As with the server load-balancing in traditional network, reasonable design and accurate judgement of overloaded controllers is inevitable and important. In previous researches on controllers' load-balancing, most of them use some indicators directly related to controllers' workload as threshold, such as load or average load diversity. However, the value of these indicators are generally empirical or random values, and it is difficult for them to achieve the best detection of load imbalance. If the threshold is too small, load imbalance is easy to occur due to the traffic transient, which will lead to frequent balancing operation, resulting in huge migration cost and poor network stability. However, a high threshold leads to a low balance rate. That is to say, some controllers have poor processing ability, while others are still in a relatively idle state. Therefore, selecting the appropriate threshold is particularly important for accurate judgement of load imbalance, and it is very helpful for timely detection and processing of overloaded controllers.

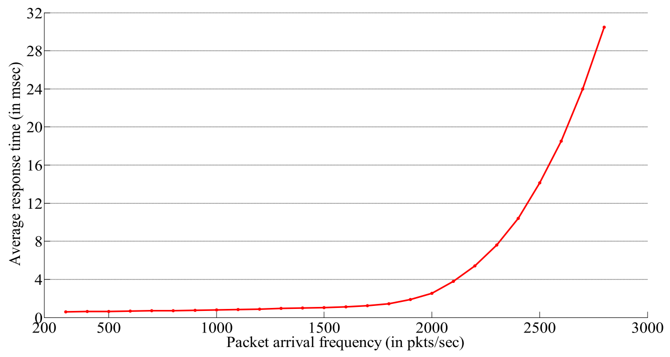


Fig. 2. Changing curve of response time to controller load.

From the user's point of view, the response time of their request messages is the most critical factor in determining the quality of the network [31]. Additionally, the response time can also measure the load difference of controllers with the same performance. The response time of the controller to the request directly affects the waiting time of the packet for forwarding. In other words, response time can affect network latency and network service quality. Therefore, in the rest of this subsection, we will make full use of response time to select the appropriate threshold, which can also achieve the tradeoff between balance rate and migration cost.

Firstly, we need to test the average response time under different load. We choose an SDN controller and get its average response time to a single PACKET_IN request message via the aforementioned method in Part A. In this test, the Java-based SDN controller, Floodlight [32], is selected to carry out a larger number of tests and experiments with an increasing PACKET_IN arrival rate from 300 packets-per-second to 3,000 packets-per-second with an interval of 100 for every period. Through many experiments, we select some the relatively stable data and obtain the average response times of the controller at different packet rates. By integrating the data and fitting the curve, we can draw these data into a curve diagram, as shown in Fig. 2.

Secondly, from the changing curve, we can get some change features of response time versus controller's load. In the beginning, the controller is in low-load, and the response time increases slowly as load increases, and the changing range is less than 0.5 ms. When the controller is under normal-load state, the change in response time is accelerated. However, when the controller is in heavy-load, the response time increases obviously.

Lastly, we need to select the appropriate response time threshold according to its variation feature. With the controller's load increasing from normal state to overloaded state, the response time increases faster. So we can obtain the appropriate threshold $t_{threshold}$ by calculating the extremum of response time variation. Let $g(t_{i-1})$ denote the average response time in the $(i-1)^{th}$ period. Thus, $g(t_i)$ can represent the average response time in the i^{th} period, when the packet arrival frequency has increased 100 packets-per-second.

So we can easily calculate the first and second order by the formulation (4) and (5) below at the end of i^{th} period.

$$g'(t_i) = \frac{g(t_i) - g(t_{i-1})}{t_i - t_{i-1}} \quad (4)$$

$$g''(t_i) = \frac{g'(t_i) - g'(t_{i-1})}{t_i - t_{i-1}} \quad (5)$$

From the formulation above, we can get the appropriate threshold $t_{threshold}$ on the extremum point in the change curve. Before this point, although the response time increases as load increases, it changes slowly and lightly. That is to say, the increasing on controller's load has little effect on users' message request. But after this point, the response time increases faster. If the controller's load keeps increasing, the response time will increase significantly. Therefore, if the controller's response time reaches this threshold, we need to quickly process this controller to prevent its response time from increasing rapidly. In this way, we can well select the appropriate response time threshold to realize the tradeoff between balance rate and balance cost.

C. Detection and Triggering of Load-Balancing

In Part B, the abrupt and continuous increase of response time can be used to choice the best time for load-balancing. Here, what we need to do is compare all controllers' response time to the appropriate threshold to find out which controller is easy to approach its performance bottleneck and tend to a heavy-load state.

After obtaining the set $A = \{t_{C_1 T_n}, t_{C_2 T_n}, \ldots, t_{C_N T_n}\}$ to represent the response time value of *N* controllers in a time interval, we can get the overloaded controllers. To effectively identify overloaded controllers and to reduce the complexity of choosing load-shifting pairs, we use two sets, *OM_C* and *IM_C*, to denote the overloaded controllers and low-load controllers. We check controllers' response time with the threshold and decide which controllers should be selected as the set *OM_C* of outmigration controllers and which should be selected as the set *IM_C* of immigration controllers. If the response time satisfies that $t_{C_i T_n} > t_{threshold}$, the controller C_i is added to *OM_C*. Accordingly, if $t_{C_i T_n} \leq t_{threshold}$, the controller C_i is added to *IM_C*. In time interval T_n, we calculate the set *OM_C* and *IM_C* by obtaining the average response time set *A* of all controllers.

In this subsection, we decide whether we need to carry out load-balancing by generating the two controller sets. We describe the algorithmic process of load-balancing detection and triggering as Algorithm 1.

D. Decision on Switch Migration

To better improve the efficiency of load-balancing, we propose a switch migration method based on Algorithm 1 to achieve multiple switch migration operations in a single load-balancing detection, when there are several overloaded controllers. It can well-realize the load-shifting of multiple controllers, greatly saving load-balancing time. If the load-balancing operation in Part C is activated, we can get the outmigration and immigration controller sets. From the

Algorithm 1 Load-Balancing Detection and Trigger

```
Input: $A = \{t_{C_1 T_n}, t_{C_2 T_n}, \cdots, t_{C_N T_n}\}$, $t_{threshold}$
Output: OM_C, IM_C
1: initialize controller set $OM\_C = \{\ \}$ and $IM\_C = \{\ \}$
2: let i be the serial number of the compared controller
3: for $i = 1 \rightarrow N$ do
4:     select $t_{C_i T_n}$ from $A$
5:     if $t_{C_i T_n} > t_{threshold}$ then
6:         add $C_i$ to $OM\_C$
7:     else
8:         add $C_i$ to IM_C
9:     end if
10: end for
11: return OM_C, IM_C
```

sets, *OM_C* and *IM_C*, the steps below can elaborate the decision-making of switch migration set *P* which are formulated as a series of migration actions indicated by the triplet, $\langle C_u, S_e, C_v \rangle$. For the first, if the judgement condition, $OM\_C \cap IM\_C$ isNotEmpty, is TRUE, both the set *OM_C* and *IM_C* are not NULL, that is to say, there must be an overloaded controller that need to be balanced and a low-load controller that can accept load-shifting. So in this case, load-balancing with switch migration is required. Then we need to conduct the steps below to generate the migration action for each overloaded controllers.

Step 1: Choose the first controller C_u with the largest response time set *OM_C*, which has the biggest workload. Then, the switch with the most serious impact in the overloaded controller will be selected to migrate. Of course, switch S_e, with the maximum switch load, is also selected.

Step 2: In set *IM_C*, all controllers' response times are basically close, and the load comparison cannot be judged according to the response time value. Thus, we directly choose the first controller C_v with the smallest workload to receive the load migration. At this point, add the selected triplet $\langle C_u, S_e, C_v \rangle$ into switch migration set *P*. Then, we remove C_u from *OM_C* and C_v from *IM_C*.

Step 3: Next, we can get three updated sets: outmigration controller set *OM_C*, immigration controller set *IM_C*, and switch migration set *P*. Then, we repeat *Step 1* and *Step 2* to select new triplets into *P* until there is an empty set in *OM_C* and *IM_C*.

The pseudo-code of switch migration decision is given as Algorithm 2.

Finally, we obtain the switch migration actions and finish the load-shifting by changing management roles of migrated switches. Through the switch migration set generated by Algorithm 2, we have made migration decisions for multiple overload controllers. The real implementation of load-balancing is done by the migration execution module. By completing this implementation process, our scheme can truly realize the multi-controller load-balancing operation.

IV. The Implement of SMCLBRT

We next describe a load-balancing framework based on SMCLBRT, as illustrated in Fig. 3. We consider a framework

Algorithm 2 Switch Migration Decision

```
Input: OM_C, IM_C, load information of all the switches
Output: P: switch migration actions set
1:  initialize migration set $P = \{\ \}$
2:  while ( $OM\_C \cap IM\_C$ isNotEmpty ) do
3:      $\mathrm{C}_u = \underset{C_i \in OM\_C}{MAX}\{Load_{C_i T_n}\}$
4:      $\mathrm{S}_e = \underset{S_k \in Q_{C_u}}{MAX}\{Load_{S_k T_n}\}$
5:      $\mathrm{C}_v = \underset{C_i \in IM\_C}{MIN}\{Load_{C_i T_n}\}$
6:      add $\langle C_u, S_e, C_v \rangle$ to P
7:      remove $C_u$ from OM_C
8:      remove $C_v$ from IM_C
9:  end while
10: return P
```

that can dynamically balance the load distribution among multiple controllers. It can finely distinguish which controller will have a high delay close to its performance bottleneck. The following four modules are needed to support the centralized management platform: monitoring module, load balance detection module, switch migration decision module, and migration execution module.

The monitoring module is used for periodic collection and statistics of average controller response times. Meanwhile, it also stores real-time load information of each controller and their switch sets. This module provides the response time data to the load balance detection module. If the load-balancing operation is activated, it also offers switch migration decision module load information.

The load balance detection module periodically calculates the mean of all controllers' average response times and compares it with the set threshold. If there is a load imbalance, this module will trigger the switch migration module and provide the sets, *OM_C* and *IM_C*, to it. Our scheme is designed in a network environment with unified controller performance. However, the performance of the controller is different between different network environments. So we need to retest the change curve and find the appropriate response time threshold using the same method in Part B. On the other hand, the lifetime of flow entries of different types of controllers is different, so the time interval, *T*, also needs to be adjusted. Therefore, we reserve a submodule to better configure the time interval and response time threshold, so that SMCLBRT can well-adapt to different network environment requirements. *The switch migration decision module* is responsible for generating switch migration decisions and does not need to perform the switch migration process. It implements Algorithm 2, which can output the migration actions expressed as a set *P* of triplet $\langle C_u, S_e, C_v \rangle$ by obtaining controller sets *OM_C* and *IM_C* and switches' load information. Then, this module provides the generated migration actions to corresponding migration execution modules.

The migration execution module provides the specific execution of the migration actions set *P*. For each migration action, it informs the involved controllers about mapping changes. It also requires controllers to send the corresponding migration messages to change control roles [8].

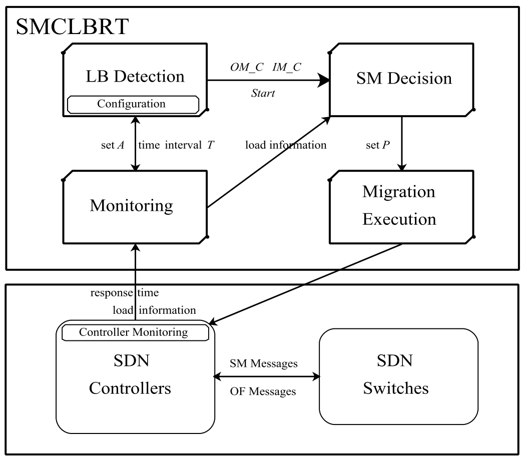


Fig. 3. Basic architecture of SMCLBRT.

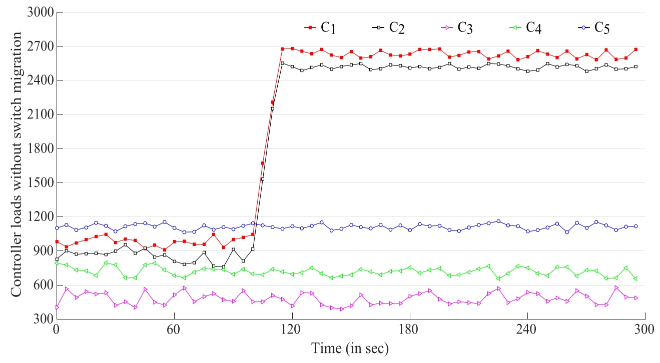
The centralized platform is designed to finely evaluate controller loads and perform real-time detection for load imbalance and multiple action decisions for overloaded controllers. Note that the centralized platform only needs to periodically acquire real-time average response times from all controllers. Consequently, it should not have any performance bottlenecks in normal circumstances. In each individual controller, the controller monitoring module is responsible for collecting PACKET_IN messages and calculating response times. Moreover, the migration execution also needs it to achieve changes to switch-controller mappings.

In the simulation experiment, we deploy Floodlight [32] as the SDN controller. The time interval, *T*, was set at 5 s in our simulation environment. A short time interval will greatly increase monitoring and computational overhead, and may result in frequent migration. A long time interval may miss out some important network state information, because the default survival time of flow entry in Floodlight controller running is 5 s.

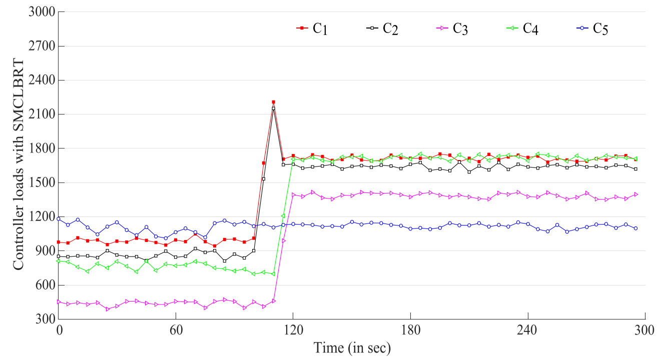
We simulate the real Internet service topology, BT Asia-Pacific (*i.e.,* 20 nodes) from the Internet Topology Zoo [33] to make this scheme more similar to the real network scenario. In the network environment of this paper, all switches are directly connected to each controller, but only managed by the master controller, which also conforms to the basic architecture of distributed SDN control plane. For the performance 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. Therefore, we choose a physical machine to run the lightweight network simulation tool, Mininet, and modify 20 Open vSwitches to inject PACKET_IN messages to corresponding controllers. Another five physical machines are needed to set up five controller nodes: C_1 to C_5. We divide the 20 nodes into five switch sets and assign them to five controllers for management.

V. Performance Evaluation

In the rest of this section, we mainly compare four schemes: Scheme I, static switch-controller mapping model; Scheme II, SMDM [13]; Scheme III, the method in [28]; and Scheme IV,



(a)



(b)

Fig. 4. Five controllers' load distribution. (a) under static switch-controller mapping; (b) under SMCLBRT.

SMCLBRT. We simulate the same experimental test environment for these four schemes and run each simulation for 5 min. By choosing the two controllers and continuously increasing the PACKET_IN request rate of their switches for 15 sec, we observe the load distribution changes under different schemes.

For the first, we compare Scheme IV with Scheme I to verify the validity of our load-balancing scheme. By running the simulation experiment for 5 min, we record the load changes in two scenarios, one is under static switch-controller mapping and one uses the SMCLBRT approach. The load distribution in two scenarios are shown in Fig. 4(a) and Fig. 4(b), respectively. In Fig. 4(a), without any load-balancing, the loads of C_1 and C_2 keep increasing in 110 sec with about 2 ms response time and in 115 sec with about 6 ms response time. Finally, their response time arrive more than 15 ms and they keep in heavy-load state continuously. However, with our SMCLBRT method, the load-increasing switches of C_1 and C_2 are migrated to the low-load controllers, C_3 and C_4, respectively in 115 sec. The increasing trend of C_1 and C_2 can be well prevented before 120 sec.

In the rest of this section, we compare with other schemes from the aspect of balance time, load distribution, migrated-switch number to verify the advantages of the overloading-controller detection by using the appropriate response time threshold and the simultaneous processing of multiple overloading controllers in our scheme.

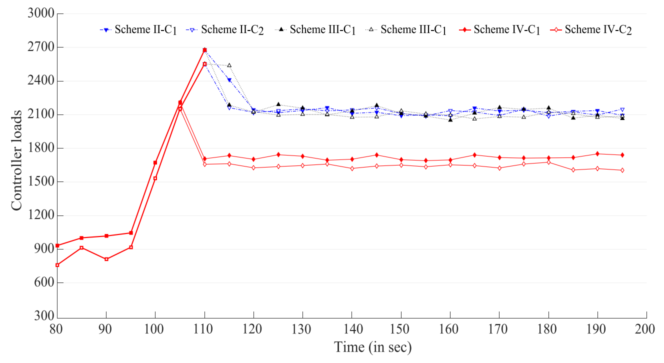


Fig. 5. Load changes of overloaded controllers.

A. Balance Time

Fig. 5 shows the overloaded controllers' load distribution before and after load-balancing operations under the three load-balancing schemes. The start time of load-balancing in our scheme is 105 sec, while the start time in Scheme II and III are 111 sec. Then, the end time of load-balancing in our scheme is 110 sec, while the end time in Scheme II and III are 120 sec.

From the results shown in Fig. 5, our scheme can balance the uneven load distribution in advance, and we use less time to deal with multiple overloaded controllers. From the user's point of view, our scheme can reduce the response time of the two overloaded controllers earlier and faster, which can also ensures a better overall response time.

Our scheme can start balance operation in advance, because we use the response time indicators to evaluate the controllers' ability to handle switches' request messages. So the load-balancing in our scheme starts earlier. On the other hand, in the balance processing, our scheme choose the switch that had the most serious impact on the response time of its master controller to migrate. In this way, the overloading controller can easily tend to normal-load state. Then, in the migrating processing, the two overloading controllers migrate their switches to low-load controllers simultaneous, which can achieve the parallel processing of multiple overloaded controllers in a single load-balancing detection operation.

B. Load Distribution

By continuously increasing the PACKET_IN request message of switches for 15 sec, different schemes can achieve a good load-balancing effect and reach a stable-load state before 120 sec. We measure the average load value of each controller in stable-load state under different schemes. Let *ratio* denote the load distribution ratio, as shown in the Eq. (6). It can roughly estimate balance rate.

$$ratio = \frac{Load_{C_i}}{\bar{Load}} \tag{6}$$

where $Load_{C_i}$ represents the workload of controller C_i, and $\bar{Load}$ represents the average workload of five controllers.

We compare the load distribution ratio in different scheme as shown in Fig. 6. Our scheme has a better balance rate after load-balancing operation than other schemes. Some reasons

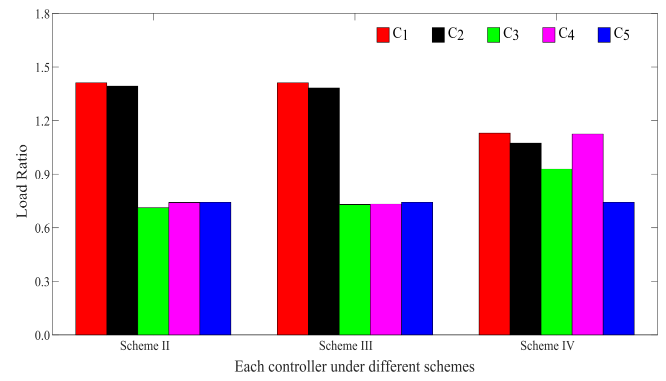


Fig. 6. Balance rate of different schemes.

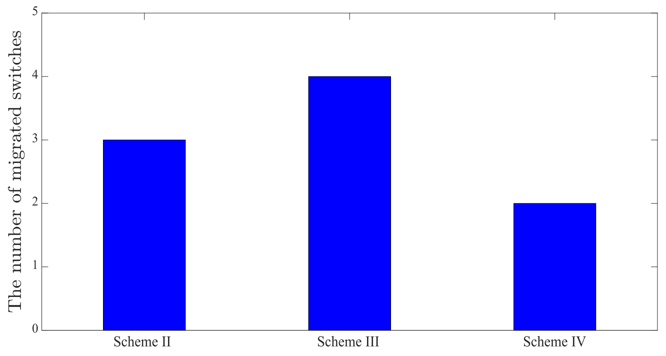


Fig. 7. The number of migrated switches.

can explain this result. The selection of the response time threshold can help us to find out and deal with the overloading controllers ahead of time. In the continuous increase process of request messages of switches, our scheme timely migrate the load-increasing switches to the low-load controllers, C_3 and C_4. Then, although the loads of C_3 and C_4 are continuously increase for 5 sec, the response time value of them are smaller than the time threshold. In Scheme II and III, load-balancing operations start after the continuous increase for 15 sec. Then, what they need to do is to reduce the workload of the two heavy-load controllers.

C. Migrated-Switch Number

Switch migration is the main load-balancing means for multiple SDN controllers. However, if there are many switches that need to be migrated, normal network services may be affected with some request packet lost. We compare the number of migrated switches in the load-balancing process under each simulation. As shown in Fig. 7, under three load-balancing approaches, our scheme has the minimum number of migrated switches after load-balancing. In the migration decision processing, our scheme always select the switch that the most impact on its master controller's response time for migration. Although this selection is unlike the migration decision in Scheme II that can generate the optimal migration triplet, we can reduce the controller's response time at one migration operation, and do not need to do another migration in the next load-balancing processing.

In this network environment, all switches are directly connected to each controller, but only managed by the master controller. The switch migration process need switch-controller communication and migration time to guarantee liveness, safety and serializability [8]. The message exchange and migration time of each migration process is essentially the same. Suppose that the communication overhead and the migration time of migrating a switch from one controller to another is *M_com* and *t_mig*. Let *Num_mig* to denote the number of migrated switch after the entire load-balancing is completed. So we can generally describe the communication overhead and migration time with *Num_mig***M_com* and *Num_mig***t_mig*, respectively. With the minimum number of migrated switches, our scheme can reduce some communication overhead and migration time in the overall balancing processing.

VI. Conclusion

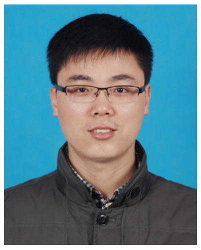
In this paper, we investigated the problem of multiple overloaded controllers. The first, we do some research on the change features of response time versus controller's load. Then, we selected the extremum point on the change curve as the appropriate response time threshold to get a trade-off between load balancing effect and migration cost. We presented our design of SMCLBRT, a novel load-balancing strategy via switch migration based on real-time response time. It makes a fine-grained judgment on controllers' workloads to timely find out the controller that might have a faster increase in its response time. Thus, our scheme provided a good load-balancing point for overloaded controllers to shift their load ahead of time. For quick balancing of multiple overloaded controllers, we also designed a switches selection algorithm to perform different switch migration operations simultaneously. The simulation results verify that our scheme can starts migration in advance and quickly reduce the workload for the overloading controllers. Therefore, SMCLBRT achieves load-balancing of multiple SDN controllers effectively and quickly. In this paper, we also have some problems that need to be improved and do some more research in the future. Our current simulation does not apply to most of load distribution patterns. In the next work in our simulation, we need to make more tests to well improve the adaptable of our scheme. The specific impact of migration processing on network quality need to be tested, and a real-time and adaptive selection of the appropriate response time threshold is also advanced in the future work. We also plan to study a better balancing approach of multiple overloaded controllers by considering the migration cost. In addition, we plan to consider some factors that cause the rapid and abnormal growth of controllers' workload, such as DDoS attack on SDN controller.

Acknowledgment

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

REFERENCES

[1] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[2] A. A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "Elasticon: An elastic distributed SDN controller," in *Proc. 10th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2014, pp. 17–28.
[3] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manag. Conf. Res. Enterprise Netw. (INM/WREN)*, 2010, p. 3.
[4] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. 9th USENIX Conf. Oper. Syst. Design Implement. (OSDI)*, Vancouver, BC, Canada, 2010, pp. 351–364.
[5] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized? State distribution trade-offs in software defined networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 1–6.
[6] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.
[7] B. Pfaff *et al.* (2012). *OpenFlow Switch Specification 1.3.0*. [Online]. Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf
[8] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 7–12.
[9] Y. Zhou *et al.*, "A load balancing strategy of SDN controller based on distributed decision," in *Proc. IEEE 13th Int. Conf. Trust Security Privacy Comput. Commun. (TRUSTCOM)*, 2014, pp. 851–856.
[10] C. Liang, R. Kawashima, and H. Matsuo, "Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers," in *Proc. 2nd Int. Symp. Comput. Netw. (CANDAR)*, Dec. 2014, pp. 171–177.
[11] G. Cheng, H. Chen, Z. Wang, and S. Chen, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.
[12] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards SDN," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 363–368.
[13] 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.
[14] M. Koerner and O. Kao, "Multiple service load-balancing with OpenFlow," in *Proc. IEEE 13th Int. Conf. High Perform. Switch. Routing*, Jun. 2012, pp. 210–214.
[15] W. Reese, "NGINX: The high-performance Web server and reverse proxy," *Linux J.*, vol. 2008, no. 173, Sep. 2008.
[16] V. Kaushal and A. G. Bala, "Autonomic fault tolerance using HAProxy in cloud environment," *Int. J. Adv. Eng. Sci. Technol.*, vol. 7, no. 2, pp. 54–59, Jul. 2011.
[17] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: Integrating software defined networking and network function virtualization," *IEEE Netw.*, vol. 29, no. 3, pp. 36–41, May/Jun. 2015.
[18] J. Li, X. Chang, Y. Ren, Z. Zhang, and G. Wang, "An effective path load balancing mechanism based on SDN," in *Proc. IEEE 13th Int. Conf. Trust Security Privacy Comput. Commun.*, Sep. 2014, pp. 527–533.
[19] R. Trestian, K. Katrinis, and G.-M. Muntean, "OFLoad: An OpenFlow-based dynamic load balancing strategy for datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 792–803, Dec. 2017.
[20] H. Zhong, Y. Fang, and J. Cui, "LBBSRT: An efficient SDN load balancing scheme based on server response time," *Future Gener. Comput. Syst.*, vol. 68, pp. 183–190, Mar. 2017.
[21] N. Gude *et al.*, "Nox: Towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.
[22] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE Netw.*, vol. 30, no. 1, pp. 58–65, Jan./Feb. 2016.
[23] L. Kuang, L. T. Yang, X. Wang, P. Wang, and Y. Zhao, "A tensor-based big data model for QoS improvement in software defined networks," *IEEE Netw.*, vol. 30, no. 1, pp. 30–35, Jan./Feb. 2016.
[24] D. Erickson, "The beacon OpenFlow controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 13–18.
[25] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 19–24.
[26] Y. Fu *et al.*, "A hybrid hierarchical control plane for flow-based large-scale software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 2, pp. 117–131, Jun. 2015.
[27] J. Yu, Y. Wang, K. Pei, S. Zhang, and J. Li, "A load balancing mechanism for multiple SDN controllers based on load informing strategy," in *Proc. 18th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Oct. 2016, pp. 1–4.
[28] Y. Zhou, Y. Wang, J. Yu, J. Ba, and S. Zhang, "Load balancing for multiple controllers in SDN based on switches group," in *Proc. 19th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Sep. 2017, pp. 227–230.
[29] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. 2nd USENIX Conf. Hot Topics Manag. Internet Cloud Enterprise Netw. Services (Hot-ICE)*, 2012, p. 10.
[30] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.
[31] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.
[32] FloodLight. *Open SDN Controller*. [Online]. Available: http://www.projectfloodlight.org/blog/2016/03/10/announcing-floodlight-v1-2/
[33] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

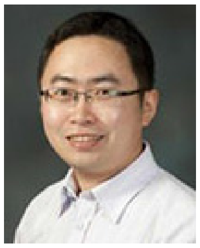


Jie Cui was born in Henan, China, in 1980. He received the Ph.D. degree from the University of Science and Technology of China in 2012. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. He has over 80 scientific publications in reputable journals, such as the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, the IEEE INTERNET OF THINGS JOURNAL, *Information Sciences*, the *Journal of Parallel and Distributed Computing*, as well as academic books and international conferences. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.

Qinghe Lu received the B.E. degree in computer science from Anhui University, Hefei, China, in 2016, where he is currently pursuing the M.E. degree with the School of Computer Science and Technology. His research focuses on software-defined networking.

Hong Zhong was born in Anhui, China, in 1965. She received the Ph.D. degree in computer science from the University of Science and Technology of China in 2005. She is currently a Professor and a Ph.D. Supervisor with the School of Computer Science and Technology, Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking. She has over 120 scientific publications in reputable journals, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE TRANSACTIONS ON BIG DATA, the IEEE INTERNET OF THINGS JOURNAL, *Information Sciences*, the *Journal of Parallel and Distributed Computing*, as well as academic books, and international conferences.

Miaomiao Tian was born in Anhui, China, in 1987. He received the Ph.D. degree in computer science from the University of Science and Technology of China in 2014. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. His research interests include cryptography and information security.



Lu Liu received the Ph.D. degree from the University of Surrey. He is currently the Head of the School of Electronics, Computing and Mathematics and a Professor of distributed computing with the University of Derby, and an Adjunct Professor with Jiangsu University. He has over 170 scientific publications in reputable journals, academic books, and international conferences. He has secured many research projects which are supported by U.K. research councils, BIS, Innovate U.K., British Council, and leading U.K. industries. He is a fellow of British Computer Society.