IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 7, 2017, accepted January 8, 2018, date of publication January 23, 2018, date of current version February 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2795576

Elastic Switch Migration for Control Plane Load Balancing in SDN

YANG ZHOU[1], KANGFENG ZHENG[1], WEI NI[2], (Senior Member, IEEE), AND REN PING LIU[3], (Senior Member, IEEE)
[1]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]CSIRO, Sydney, NSW 2122, Australia
[3]Global Big Data Technologies Centre, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Yang Zhou (zhouyang@bupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802703 and in part by the National Natural Science Foundation of China under Grant 61602052.

ABSTRACT Software-defined network (SDN) provides a solution for the scalable network framework with decoupled control and data plane. Migrating switches can balance the resource utilization of controllers and improve network performance. Switch migration problem has to date been formulated as a resource utilization maximization problem to address the scalability of the control plane. However, this problem is NP-hard with high-computational complexities and without addressing the security challenges of the control plane. In this paper, we propose a switch migration method, which interprets switch migration as a signature matching problem and is formulated as a 3-D earth mover's distance model to protect strategically important controllers in the network. Considering the scalability, we further propose a heuristic method which is time-efficient and suitable to large-scale networks. Simulation results show that our proposed methods can disguise strategically important controllers by diminishing the difference of traffic load between controllers. Moreover, our proposed methods can significantly relieve the traffic pressure of controllers and prevent saturation attacks.

INDEX TERMS Earth mover's distance, load balancing, reconnaissance, saturation attacks, switch migration.

I. INTRODUCTION

Distributed control plane is a promising technique in software-defined networks (SDN) to achieve reliability and scalability [1], [2]. This type of network usually hosts multiple controllers which are responsible for deciding how packets should be forwarded by switches. As connections between controllers and switches are mostly static, they cannot adapt to the changes of network and thus, controllers may suffer load imbalance and performance degradation with the increase of network scale [3]. The controller placement problem [4] has been widely researched to optimize the performance of network by deciding where and how many controllers to be placed. However, it requires frequently changing the locations of controllers.

On the other hand, it is a key issue to prevent controllers, especially the strategically located important ones, from threats and attacks to enhance the security of network control plane. In SDN, switches send PACKET_IN packets to controllers to request routing information when flow table has no entry to match traffic flows. This event is specially utilized by adversaries to launch attacks to overload or even crash down the controllers [5], [6]. Furthermore, the failure of heavily-loaded controllers may even cause the cascading failures of other controllers [7], [8].

In general, the switch migration problem is an NP-hard constrained node selection problem [9]. It has been formulated to be a resource utilization maximization problem with constraints of CPU, bandwidth, and memory of controllers [10]. The current solutions for this problem are typically heuristic, such as Markov approximation [3], greedy method [11], and so on. These methods focus on improving the performance or scalability of control plane, which hardly claim either optimality or security in any sense. Moreover, strategically located controllers may handle more traffic than others, which makes them easy to be recognized or saturated. There is few work focuses on the security issues to protect these important nodes from attacks.

In this paper, we propose a dynamic load balancing method to migrate switches by altering logical topology of SDN, aiming at improving the scalability and security of control plane. Our key idea is to interpret the switch migration as a signature matching problem which in turn can be formulated as a three-dimensional earth mover's distance (EMD) model. Further considering resilience and computational efficiency, we also develop suboptimal algorithms with polynomial time-complexities. Simulation results show that our algorithms are able to suppress traffic difference among controllers and protect them from reconnaissance and saturation attacks. Time-efficiency of our methods is also evaluated, which confirms that the proposed suboptimal algorithms can be applied to large-scale networks.

The rest of paper is organized as follows. Section II presents related work. Section III describes the system and network model. We also introduce the implementation of system and the definition of EMD in this section. In Section IV, we discuss the detail of designed algorithm. We present the computational efficient heuristics in Section V. Section VI evaluates the experiment results of our algorithm. We conclude the paper in Section VII.

II. RELATED WORK

The controller placement problem was firstly proposed in [4] to decide how many and where controllers to be placed in SDN to achieve scalability. More efforts were spent on solving the controller placement problem [8], [12], [13] and discussing the resilience and scalability of control plane. However, as the important role that controller plays in network, frequently mutation brings network extra burden of performance.

The switch migration problem has been researched to address the issue of the scalability of control plane. Reference [14] firstly proposed a migration protocol for seamless migration of switches among multiple controllers. However, how to select both the switches to migrate and the target controllers was not described in detail. In [15], a dynamic switch migration mechanism was designed for clustered controllers, which guaranteed the sustainability of control plane even when some controller crashed down. The aggregate load of cluster should be collected before migration, which increased the processing time. The switch migration problem was also formulated as network utility maximization problem in [3] with capacity constraints, and solved in a distributed manner at each controller by taking approximation methods. At each instant, the switch and the target controller were all randomly selected. In [10], the switch migration problem was modeled as a centralized resource utilization maximization problem with constraints of CPU, bandwidth, and memory, and solved by non-cooperative game theory method. Reference [11] built a switch migration scheme based on greedy algorithm to maximize the trade-off between migration costs and the load balance rate. However, these schemes have not taken any security challenges of control plane into account.

Referring to the security of control plane, some solutions have been proposed to detect SDN distributed denial-of-service (DDoS) attacks by analyzing traffic patterns, e.g, entropy of destination addresses [5], number of invalid packets per time window [16], distributions of different flow features [17], [18], and so on. And the mitigation of attacks was mostly implemented as traffic migration by modifying flow rules [19]. A load balancing scheme was proposed in [20] to migrate traffic when DDoS occurs. But this method focused on the overloaded links and released traffic burden by simply finding out the shortest path bypassing overloaded links between switches.

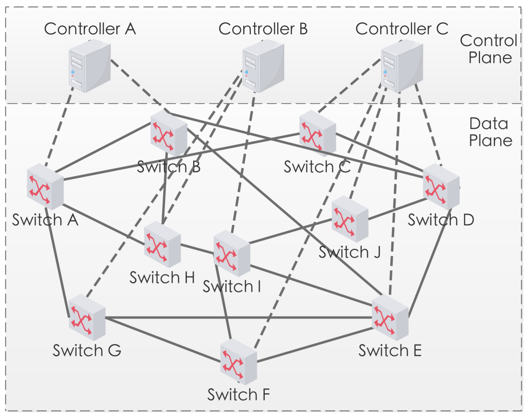

FIGURE 1. An illustration of SDN topology with multiple controllers.

III. SYSTEM MODEL

In this section, we introduce the network model and attack model, and also discuss the method to implement our system in detail. We also describe a fundamental knowledge of the EMD in this section aiming at clearly explaining our proposed 3-dimensional EMD model in Section IV.

A. NETWORK MODEL

Consider an SDN network with multiple controllers. Fig. 1 gives an illustration in detail, where there are 10 switches and 3 controllers in the network. Suppose that the physical locations of controllers are duplicate with switches, but functionally logically independent of each other [21]. As explained in OpenFlow protocol v1.4.0 [22], controllers may have three different kinds of roles, i.e., master, equal and slave. For a network domain, there is only one master controller, which facilitates automated network management. In our model, a switch can only be controlled by one master controller, and a controller can control more than one switch. At the same time, there are many alternative slave controllers for switches. A slave controller will be selected as a new master if the original master controller fails.

Let $G(\mathcal{N}, \mathcal{E})$ represent the network topology, where $\mathcal{N}$ collects the vertexes (i.e., switches), $\mathcal{E}$ collects edges (i.e., connections between switches). The size of network is N, i.e., $|\mathcal{N}| = N$, where $|\cdot|$ represents for cardinality. $\mathcal{P}$ represents the set of controllers. $|\mathcal{P}| = P$.

Many factors may impact the performance of controllers, e.g., CPU, memory and throughput. Compared with the rest, the processing of PACKET_IN events is generally regarded as the significant part of controller load [3], [8], [23]. There are also other types of packets that are sent from the switches to the controller, such as topology discover messages. In our paper, we assume that loads of controllers are equal to the total traffic for receiving all packets sent from switches. Each switch is responsible for receiving traffic requests from end-hosts that connect to it, and forwarding traffic as flow entries indicate. When the flow table in the switch has no entry to match traffic flow, the switch sends PACKET_IN packets to the corresponding master controller to request the forwarding strategies.

B. ATTACK MODEL

We suppose that strategically important located controllers are those which control more switches and deal with more traffic. In Fig. 1, controller C is a strategically important located controller. In practice, these controllers usually play important roles in a network, and become easier to be targeted at attacks. As a result, how to protect these controllers from attacks is an important issue.

Two kinds of attacks are considered in our paper. One is reconnaissance attack, where adversaries can monitor or eavesdrop traffic information of control plane. By this means, they can recognize the strategically important located controllers. Concerning the monitoring activities in practice, we define the historically accumulative traffic of controllers as the sum of traffic during monitoring intervals. A bigger difference of historically accumulative traffic between controllers means a larger gap of traffic loads they deal with, which makes strategically important ones easier to be identified. Consider the worst case, i.e., adversaries have already recognized strategically important controllers. In this case, our method aims at converting those strategically important controllers into lightly important ones, to make the ones adversaries have already recognized no more trivial, thus reduce the impacts on reconnaissance attacks. In practice, adversaries can launch reconnaissance attacks by using a compromised network tapping application in SDN. By this means, they can eavesdrop the communication information between the controllers, especially the clear text ones. For example, adversaries can intercept the sFlow [24] datagrams containing traffic statistics, which are sent by using the unencrypted UDP packets, to monitor the network.

The other attack we consider is saturation attack of control plane. As defined by SDN, controllers are responsible for maintaining the forwarding strategies of the network, and the switches should request them by sending PACKET_IN packets when they have no flow entry to match. This event is especially utilized to adversarially saturate controllers in SDN. The attackers can usually achieve this goal by spoofing packet source or destination IP addresses, i.e., forging large amounts of packets that the switches forwarding table can't match [6].

C. SYSTEM IMPLEMENTATION

In general, as our algorithms are designed as logically centralized ones, they can reside in every controller in an SDN environment.

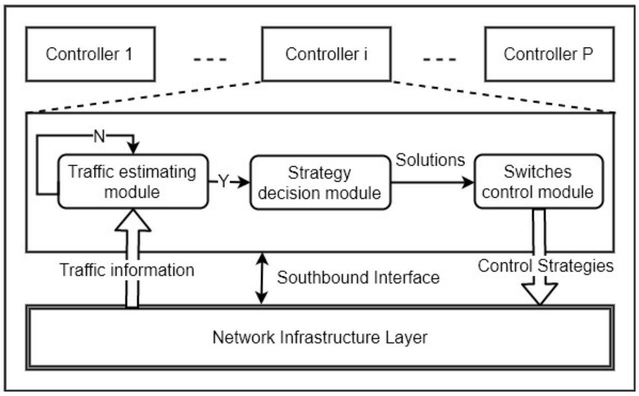

FIGURE 2. Overview of system infrastructure.

We design that each controller has a global view of the network, making independent decisions. The controllers should maintain a traffic estimation module, which calculates the historical traffic load during estimating intervals. Our algorithms are triggered after at least one controller's historical load is over a threshold that network administrator set before the algorithms run, or on demand of administrators in the network. After obtaining the solutions by running our algorithms, controllers reassign the switches according to the results. To avoid network conflicts, the controllers only migrate the switches that are controlled by themselves to other controllers. The process of reassignment can be implemented as modifying roles of controllers [14]. After migration, the controllers continue to repeat these steps for another round of estimating and reassignment.

We explain the whole process in detail as Fig. 2 shown.

D. EARTH MOVER'S DISTANCE

EMD [25] is a well-known algorithm that is widely used to measure the difference of two images [26]. The computation of EMD is based on linear transportation problem, with an objective of minimizing transmission cost. The cost is defined as the amount of earth transported by the distance.

In image applications, pixels are quantified as coordinates for convenience of calculations. Suppose that figure P is composed of m clusters, and $P = \{(p_1, \omega_{p_1}), \cdots, (p_m, \omega_{p_m})\}$ is the signature of P. p_i represents cluster i, ω_{p_i} is the weight of cluster i. $Q = \{(q_1, \omega_{q_1}), \cdots, (q_n, \omega_{q_n})\}$ is the signature of figure Q having n clusters. Moreover, d_{ij} is the ground distance between p_i and q_j, which can be defined as the Euclidean distance, or other distance measures. The objective is to find out the optimal flow $F = [f_{ij}]$, of which f_{ij} denotes the flow from p_i to q_j, making sure that the overall moving cost is minimized, as can be formulated as,

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{1}$$

The moving process should satisfy following constraints,

$$f_{ij} \geq 0, \quad 1 \leq i \leq m, \; 1 \leq j \leq n \tag{2a}$$

$$\sum_{j=1}^{n} f_{ij} \leq \omega_{p_i}, \quad 1 \leq i \leq m \tag{2b}$$

$$\sum_{i=1}^{m} f_{ij} \leq \omega_{q_j}, \quad 1 \leq j \leq n \tag{2c}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}f_{ij} = \min\left(\sum_{i=1}^{m}\omega_{p_i}, \sum_{j=1}^{n}\omega_{q_j}\right) \tag{2d}$$

where (2a) restricts that the flows can only move from P to Q and not vice versa. (2b) restrains that the amount of earth that can move out from p_i should not exceed its weight ω_{p_i}. Also q_j cannot receive more earth than its weight ω_{q_j}, as shown in (2c). (2d) forces to move the amount of earth as much as possible.

In this context, the EMD is defined as the overall moving work normalized by the total amount of earth moved, as specified by,

$$\text{EMD(P, Q)} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{3}$$

IV. OPTIMAL SWITCH MIGRATION MODEL

In this section, we propose to interpret switch migration problem as a signature matching problem to obtain optimal solutions, and extend the traditional EMD algorithm into a three-dimensional model to measure and reduce the difference of traffic load between controllers.

The historically accumulative traffic of each controller is visualized as a signature, and the switches are migrated to conform the signature to the one with even historical loads across all controllers. We also interpret that a controller has higher exposure probability if it deals with more traffic during historically monitoring intervals, and is also easier to be saturated because the large amount of requests it receives from switches. Thus the main goal of our algorithms is to equally distribute traffic loads between controllers without exceeding the capacities of them. By this means, historically important controllers can be disguised and protected, delaying or even preventing potential attacks.

A. OPTIMAL SWITCH MIGRATION MODEL FOR DELAY-TOLERANT TRAFFIC

First consider delay-tolerant traffic. We denote v_i' and $\bar{v}'$ as historically accumulative traffic of each controller i and their average, respectively. Similarly, denote v_i and $\bar{v}$ as currently accumulative traffic of controller i and the average. $\bar{v} = \frac{1}{P}\sum_{i=1}^{P} v_i, i \in \mathcal{P}$.

The proposed algorithm attempts to conform the pictorial signature of the accumulative traffic of controllers to a uniform one with even traffic of $\bar{v}$ at every node by reassigning switches. To do this, we construct two signatures, i.e., two one-dimensional distributions: $\{v_i - v_i', \forall i \in \mathcal{P}\}$, $\{[\bar{v} - v_j']^+, \forall j \in \mathcal{P}\}$. The first signature $\{v_i - v_i'\}$ provides the traffic volumes that controller i supplies, i.e., the traffic that controller i will receive in this monitoring interval. The second signature $\{[\bar{v} - v_j']^+\}$ denotes the demand of each controller to achieve evenly distributed traffic among all controllers.

The EMD is then described to measure the difference between these two signatures, as given by

$$\epsilon^* = \arg\min_{\epsilon(\tau)} \tau \tag{4a}$$

$$\epsilon(\tau) = \left(\frac{\sum_{i=1}^{\mathcal{P}}\sum_{j=1}^{\mathcal{P}}\sum_{k=1}^{\mathcal{N}} c_{ij}\theta_{ik}t_k x_{ijk}}{\min\left(\sum_{i=1}^{\mathcal{P}}(v_i - v_i'), \sum_{j=1}^{\mathcal{P}}\left[\bar{v} - v_j'\right]^+\right) - \tau}\right) \tag{4b}$$

where $\left[\cdot\right]^+ = \max(\cdot, 0)$; $x_{ijk} = 1$ if switch k is migrated from controller i to j, or $x_{ijk} = 0$, otherwise; $\theta_{ik} = 1$ when switch k is currently controlled by controller i, or $\theta_{ik} = 0$, otherwise. $\theta_{ik}t_k$ denotes the traffic controller i receives from switch k. c_{ij} denotes the cost of migrating the switches from controller i to j.

Keep in mind our goal to minimize the difference of accumulative traffic between controllers. Meanwhile, the EMD is inherently defined to minimize the difference. Therefore, we define the cost of migrating as the normalized variance of controllers accumulative traffic between the average, as follows

$$c_{ij} = \sqrt{\left(\frac{1}{2}((v_i - \bar{v})^2 + (v_j - \bar{v})^2)\right)}. \tag{5}$$

Our EMD has a different form to the conventional definition mentioned in Section III-D. An auxiliary variable τ is defined to indicate the gap between the total traffic load that is expected to migrate and the total traffic that can migrate, whereas there is no such gap for the conventional EMD. This is due to the fact that the proposed switch migration is discrete and these two loads do not equate in most cases.

Given v_j', $\bar{v}'$, v_j and $\bar{v}$, we can formulate a binary linear programming problem to minimize the EMD of the two signatures, as given by

$$\min\left(\sum_{i=1}^{\mathcal{P}}\sum_{j=1}^{\mathcal{P}}\sum_{k=1}^{\mathcal{N}} c_{ij}\theta_{ik}t_k x_{ijk}\right) \tag{6a}$$

$$s.t. \sum_{j=1}^{\mathcal{P}} \theta_{ik}x_{ijk} \leq 1, \quad \text{for any } k; \tag{6b}$$

$$\sum_{j=1}^{\mathcal{P}}\sum_{k=1}^{\mathcal{N}} \theta_{ik}t_k x_{ijk} \leq v_i - v_i', \quad \text{for any } i \in \mathcal{P}; \tag{6c}$$

$$\sum_{i=1}^{\mathcal{P}}\sum_{k=1}^{\mathcal{N}} \theta_{ik}t_k x_{ijk} \leq \left[\bar{v} - v_j'\right]^+, \quad \text{for any } j \in \mathcal{P}; \tag{6d}$$

$$\min\left(\sum_{i=1}^{\mathcal{P}}(v_i - v_i'), \sum_{j=1}^{\mathcal{P}}\left[\bar{v} - v_j'\right]^+\right) - \sum_{i=1}^{\mathcal{P}}\sum_{j=1}^{\mathcal{P}}\sum_{k=1}^{\mathcal{N}} \theta_{ik}t_k x_{ijk} \leq \tau; \tag{6e}$$

$$x_{ijk} \in \{0, 1\} \tag{6f}$$

Here, constraint (6b) restricts a switch can only be migrated from a controller to one controller; (6c) and (6d) restrict the amounts of traffic that can be supplied and demanded by the first and the second signatures, respectively; (6e) specifies the maximum amount of traffic that can be migrated; and (6f) specifies the variables to be binary.

Given τ, problem (6) is an integer linear program, and can be optimally solved using a branch and bound/cut algorithm [27]. A bisection method can be taken to recursively search for the minimum feasible value of τ. When τ is too small, (6) can become infeasible. When τ is large, (6e) becomes inactive and (6) becomes always feasible. To this end, (6) is an on-off function of τ preserving monotonicity, and can be readily solved bisectionally.

Algorithm 1 Optimal Switch Migration Model for Delay-Tolerant Traffic

```
Input: $G(\mathcal{N}, \mathcal{E})$; A; the set of controllers $\mathcal{P}$; the time instant
    $T$; the accumulative traffic of each controller node $v_i'$; the
    current traffic of each switch $t_k$; and the initial $\theta_{ik}$ $(i \in$
    $\mathcal{P};\ k \in \mathcal{N})$.
Output: $\theta_{ik},\ \forall i \in \mathcal{P},\ k \in \mathcal{N}$
1: repeat
2:     Let $v_i = v_i' + \sum_{k \in \mathcal{N}} \theta_{ik} t_k$.
3:     calculate $\bar{v} = \frac{1}{P} \sum_{i=1}^{\mathcal{P}} v_i$, and $c_{ij}$ using (5);
4:     substitute $v_i$, $\bar{v}$, $v_i'$ $(i \in \mathcal{P})$, $\bar{v}'$, $c_{ij}$ and $\theta_{ik}$ into the EMD
       problem (6), and solve the problem optimally using the
       binary branch and bound/cut method;
5:     update
           $\theta_{ik} \leftarrow \sum_{i \in \mathcal{P}} \theta_{ik} x_{ijk}$;
           $v_i \leftarrow v_i' + \sum_{k \in \mathcal{N}} \theta_{ik} t_k$;
           $(i \in \mathcal{P};\ k \in \mathcal{N})$;
6: until next time instant.
```

Algorithm 1 describes in detail our algorithm. As described, the kernel of the algorithm is the EMD problem (6) solved by using the binary branch and bound/cut method in Step 4. The optimal solution for (6) is used to update θ_{ik}; or in other words, to find the new assignments of the switches, as depicted in Step 5. By repeating these steps, the switches controlled by any heavily loaded controllers move to those lightly loaded. The convergence of the iterations can be guaranteed, since $\bar{v}$ does not decrease during the iterations while it is also obviously upper bounded.

B. OPTIMAL SWITCH MIGRATION MODEL WITH BOUND CONSTRAINTS

We also consider our model to the delay-bounded traffic scenario. There are many constraints in real network when concerning the migration of switches. In this case, we describe the demands of attaching constraints to our method and their applicable scenes.

(i) Capacity Constraint
To prevent the controllers from saturation attacks, we propose to add (7) into (6), which restrains the capacity of controller i to be less than its upper bound C_i.

$$c_i - \sum_{j=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k x_{ijk} + \sum_{i'=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{i'k} t_k x_{i'ik} \leq C_i \qquad (7)$$

Here, $\sum_{j=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k x_{ijk}$ is the total traffic that controller i moves away, $\sum_{i'=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{i'k} t_k x_{i'ik}$ is the traffic migrated to i. c_i represents the current traffic load of controller i (not the accumulative traffic). In this way, the loads of controllers are tightly restricted under their upper bound of capacities.
Note that problem (6) may have no feasible solutions when controllers cannot adjust their loads within the limits of capacities, which results in overloading some controllers. To avoid this, we can further reroute the traffic of heavily-loaded controllers to middleboxes, which are responsible for analyzing and detecting malicious traffic, or just drop it. However, addressing this problem is beyond the scope of our paper.

(ii) Bandwidth Constraint
The bandwidth of southbound is usually the bottleneck of OpenFlow which is the most popular protocol in SDN. In this case, we restrict the total bandwidth of controller i's southbound to be less than its upper bound B_i, as given by,

$$\sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k + \sum_{i'=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{i'k} t_k x_{i'ik} - \sum_{j=1}^{\mathcal{P}} \sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k x_{ijk} \leq B_i, \quad \text{for any } i \in \mathcal{P}; \qquad (8)$$

(iii) Latency Constraint
The latency, in which we are particularly interested, is the latency for the switches to request new flow rules to the controller. A higher latency usually results in the switches failing to install flow rules in time, which impacts the efficiency and availability of the network. Therefore, considering the quality of service (QoS) of the network, we add the constraints of propagation latency into (6) as follows.

$$d_{jk} x_{ijk} \leq L_j \quad \text{for any } j, k. \qquad (9)$$

Here, L_j is the maximum propagation latency of controller j. d_{jk} represents the actual propagation delay of the path from controller j to switch k. In our paper, we formulate d_{jk} as the sum of the propagation delay of each link along the actual path from j to k. In practice, controllers determine the path from j to k, given the current view of the topology. We simplify it by using the traditional routing protocols, e.g., OSPF, to implement the process of discovering the shortest path between j and k in simulations.
The method to solve this QoS constrained model is similar to the method we described.

V. HEURISTIC SWITCH MIGRATION MODEL

In this section, we propose a heuristic model with reduced computational complexity and time-efficiency. This model is a simplified version of Algorithm 1 and decouples the three-dimensional switch migration model with two concatenated sub-problems, namely, traffic allocation and switch selection.

The traffic allocation is formulated as a signature matching problem similar as the EMD model, which is a linear programming transportation problem. This problem is designed to compute the amount of traffic to migrate, with minimized cost. The switch selection is achieved by using the aforementioned binary branch and bound/cut algorithm after we solve the traffic allocation problem, but with a substantially smaller number of variables, i.e., less than N. As a result, the complexity can be significantly reduced to solve the transportation problem, facilitating applications to large-scale networks.

We begin with optimizing variable y_{ij}, $i \in \mathcal{N}, j \in \mathcal{N}$, which represents the traffic to move from controller i to a destination, i.e., controller j. Following the EMD criterion, a linear programming problem can be formulated to determine the total traffic that needs to migrate from the heavily-loaded controllers, as given by

$$\min \left(\sum_{i=1}^{\mathcal{P}} \sum_{j=1}^{\mathcal{P}} c_{ij} y_{ij} \right) \tag{10a}$$

$$s.t. \sum_{j=1}^{\mathcal{P}} y_{ij} \le v_i - v_i', \quad \text{for any } i \in \mathcal{P}; \tag{10b}$$

$$\sum_{i=1}^{\mathcal{P}} y_{ij} \le [\bar{v} - v_j']^+, \quad \text{for any } j \in \mathcal{P}; \tag{10c}$$

$$\min \left(\sum_{i=1}^{\mathcal{P}} (v_i - v_i') - \sum_{j=1}^{\mathcal{P}} [\bar{v} - v_j']^+ \right) - \sum_{i=1}^{\mathcal{P}} \sum_{j=1}^{\mathcal{P}} y_{ij} \le \tau; \tag{10d}$$

which can be solved by using the Simplex method. Here, (10b) restricts the traffic controller i moves out should not exceed the amount it provides, i.e., $v_i - v_i'$. (10c) limits the total traffic controller j receiving from other controllers is lower than the difference between $\bar{v}$ and v_j'.

The set of solutions is denoted by $\left\{ y_{ij}^*, \text{ for any } i, j \in \mathcal{P} \right\}$. A bisection search can also be taken to identify the minimum value of τ to preserve the feasibility of (10).

As mentioned in Section III-A, before the switch migration, the traffic of each controller is composed by the number of requests sending by the switches controlled by the controllers, i.e., $v_i - v_i' = \sum_{k,\theta_{ik}=1} \theta_{ik} t_k$. Given y_{ij}^*, we can proceed to select switch k that is controlled by controller i, i.e., $\theta_{ik} = 1$, to redirect it from controller i to j. We formulate a binary linear program, as given by

$$\max \left(\sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k z_{ijk} \right) \tag{11a}$$

$$s.t. \sum_{k=1}^{\mathcal{K}} \theta_{ik} t_k z_{ijk} \le y_{ij}^*; \tag{11b}$$

$$z_{ijk} \in \{0, 1\} \tag{11c}$$

which can be readily solved using the binary branch and bound/cut.

We proceed to offload k controlled by controller with $\max_{i \in \mathcal{N}} v_i$. (11) is used to move out the switch first to the controller with $\min_{j \in \mathcal{P}} v_j$, and then to the other controller in increasing order of v_j. This repeats for the other nodes in decreasing order of v_i.

Algorithm 2 summarizes the proposed computationally efficient heuristic approach. Steps 3 and 4 execute the traffic allocation, following the EMD criterion and using (10). Steps 5 to 12 conduct the switch selection, using (11). These two parts concatenate to reduce the complexity.

Algorithm 2 Heuristic Model for Delay-Tolerant Traffic

```
Input: $G(\mathcal{N}, \mathcal{E})$; A; the set of controllers $\mathcal{P}$; the time instant
    $T$; the accumulative traffic of each controller node $v_i'$; the
    current traffic of each switch $t_k$; and the initial $\theta_{ik}$ ($i \in$
    $\mathcal{P}$; $k \in \mathcal{N}$).
Output: $\theta_{ik}$, $\forall i \in \mathcal{P}, k \in \mathcal{N}$
1:  repeat
2:    Run steps 2 to 3 of Algorithm 1
3:    Substitute $v_i$, $\bar{v}$, $v_i'$ ($i \in \mathcal{P}$), $\bar{v}'$, $c_{ij}$ and $\theta_{ik}$ into (10), and
      solve the problem using the Simplex Method;
4:    Arrange $\{v_i, \forall i \in \mathcal{N}\}$ in decreasing order so that
      $v_{\pi(1)} \ge v_{\pi(2)} \ge \cdots \ge v_{\pi(N)}$.
5:    for $i = 1, \cdots, N$ do
6:      Let $\mathcal{Q} = \cup_{k=1}^{K} \{j, \text{ for which, } \theta_{jk} = 1\}$.
7:      repeat
8:        Let $j^* = \min_{j \in \mathcal{P}} v_j$;
9:        Offload traffic flows from node $\pi(i)$ to $j^*$ by
          solving (11) for $z_{\pi(i)j^*k}$, given $y_{ij}^*$;
10:       Update
            $\mathcal{Q} \leftarrow \mathcal{Q} \setminus j^*$;
            $v_{\pi(i)} \leftarrow v_{\pi(i)} - \sum_{k=1}^{K} \sum_j z_{\pi(i)j^*k} t_k$;
            $v_j \leftarrow v_j + \sum_{k=1}^{K} z_{\pi(i)j^*k} t_k$, $\forall j \in \mathcal{P}$;
            $\theta_{\pi(i)k} \leftarrow \sum_{k \in \mathcal{K}} \theta_{\pi(i)k} z_{\pi(i)j^*k}$;
            $\theta_{j^*k} \leftarrow \sum_{k \in \mathcal{P}} \theta_{j^*k} z_{\pi(i)j^*k}$.
11:     until no switch can be further offloaded from node
        $\pi(i)$, or $\mathcal{Q} = \emptyset$.
12:   end for
13: until next instant.
```

Similar to Algorithm 1, we can also incorporate constraints when applying this heuristic method.

(i) Capacity Constraint
Similar to (7), (12) shows the case with a capacity constraint, which is attached to (10) to prevent controllers from saturation attacks.

$$c_i - \sum_{j=1}^{\mathcal{P}} y_{ij} + \sum_{i'=1}^{\mathcal{P}} y_{i'i} \le C_i, \quad \text{for any } i \in \mathcal{P} \tag{12}$$

where c_i denotes the current traffic load of controller i. $\sum_{j=1}^{\mathcal{P}} y_{ij}$ denotes the amount of traffic moving out from controller i, and $\sum_{i'=1}^{\mathcal{P}} y_{i'i}$ represents the total traffic that controller i receives. It limits the total traffic load of controller i should be smaller than its upper bound.

(ii) Bandwidth Constraint
Considering the bandwidth of southbound, we proceed to add constraint after (10) when dealing with the constraint of southbound bandwidth, as given by,

$$v_i - v_i' - \sum_{j=1}^{\mathcal{P}} y_{ij} + \sum_{i'=1}^{\mathcal{P}} y_{i'i} \le B_i, \quad \text{for any } i \in \mathcal{P} \tag{13}$$

(ii) Latency Constraint
We can also proceed to add latency constraint, i.e., (9), to (11), to ensure the QoS of network. And this problem can also be solved by binary branch and bound/cut.

VI. SIMULATION AND EVALUATION

In this section, we simulate our proposed algorithms in Section IV and Section V, and evaluate the effectivenesses of these methods with different metrics.

A. EXPERIMENT SETUP

Random k-regular graphs was generated as network topologies, by using Python package Networkx [28], where the degrees of nodes are k. Without loss of generality, we assume that the traffic sent by the switches are random and uniformly distributed at each time interval. Other traffic distributions, such as Poisson distribution, can also be implemented, since the proposed algorithm is general and applicable to different types of traffic conditions. In our simulation, we set the amount of traffic forwarded by the switches to the controller, as loads of controller, as mentioned in Section III-A.

Simulations are run in a computer with 32G RAM and 6 cores of Intel Xeon CPU. We use IBM-CPLEX optimizer [29] to solve linear programming problems developed in Section IV and Section V.

B. EVALUATION METRICS

We assume that adversaries can identify strategically important controllers by monitoring historically accumulative traffic of controllers. In this case, the difference between loads of controllers is utilized by adversaries to recognize strategically important ones. Thus, we introduce a coefficient of variation (CV) as a metric to evaluate the effectiveness of algorithms against reconnaissance or eavesdropping attacks, which is defined as follows:

$$c_v = \frac{\sigma}{\mu}, \tag{14}$$

where σ is the standard deviation and μ is the mean. CV is also known as relative standard deviation (RSD), which is a measure of the dispersion of distributions. A large value of CV indicates a big difference between loads of the controllers, which makes adversaries easy to recognize strategically important controllers.

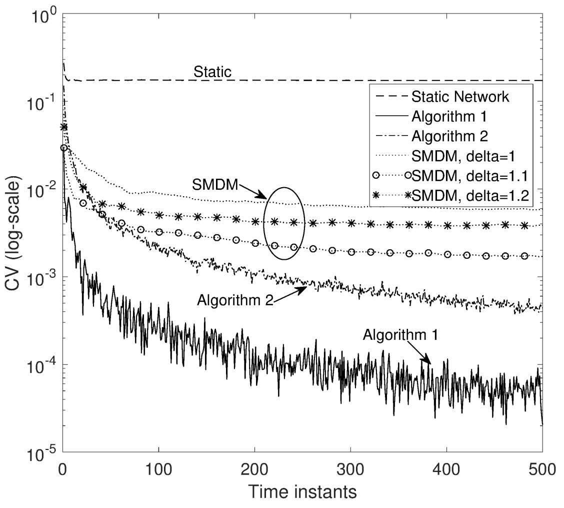

FIGURE 3. Simulation results of network with 20 nodes and 3 controllers. 50 rounds of iterations.

Saturation attack is also considered, which is launched by generating large number of PACKET_IN requests to controllers. To evaluate the effectiveness of our methods to this attack, we measure the load of controllers at any time instant. When the load of controller is over its upper bound of capacity, it will have higher risk of being compromised by adversaries.

C. EXPERIMENT RESULTS

For comparison purpose, we simulate our proposed methods, i.e., Algorithm 1 and Algorithm 2, together with other algorithms, including the case of static network and the switch migration-based decision-making algorithm (SMDM) proposed in [11]. The model of static network is as such that which the switch assignments do not change during the whole process of simulation. Thus the results of static method depend on the initial assignment of network. As to the SMDM method, a variable δ is defined as the trigger of the switch migration, i.e., the threshold of load diversity. We study the impacts of δ on the final solutions by implementing SMDM with different δ.

Firstly, we demonstrate the advantages of our proposed methods on balancing load.

Fig. 3 compares the CV of accumulative traffic between the algorithms in a 20-node network with 3 controllers. Each line in Fig. 3 represents the average of 50 rounds of iterations, and at each iteration the inputs of all algorithms are the same. Each round of simulation has 500 times of traffic requests, which can also be interpreted as 500 intervals of new traffic. As the y-axis of Fig. 3 is set as log-scale, it amplifies the differences between CVs, and the lines of Algorithm 1 and Algorithm 2 are jagged. Fig. 3 shows that Algorithm 1 gains the minimum CV and performs best. We also see that the SMDM method with $\delta = 1.1$ performs best among three cases of SMDM. That is because δ represents the upper bound of load diversity between controllers, and the load diversity is

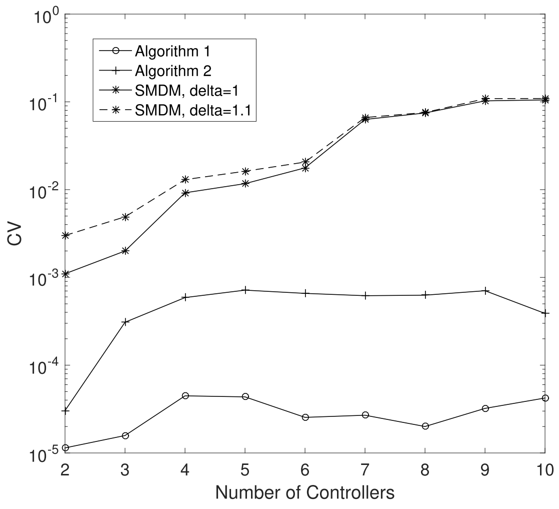

FIGURE 4. Simulation results of 20-node network with different number of controllers.

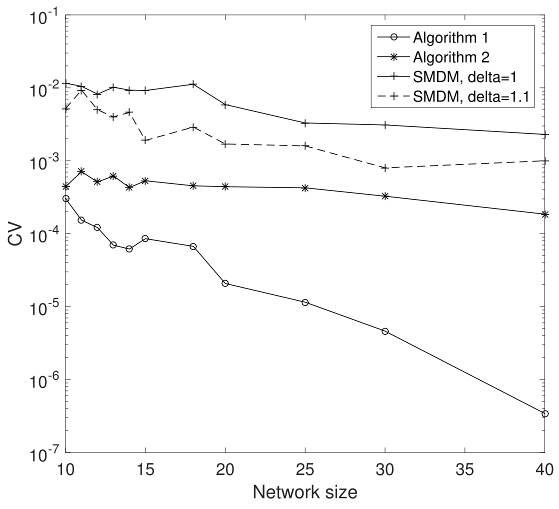

FIGURE 5. Simulation results of different sizes of network. The number of controllers is 3.

computed as the ratio of two controllers loads, i.e., $\frac{v_i}{v_j}$. When δ is equal to 1, some pairs of controllers with small difference are chosen as the outmigrating controllers and immigrating controllers, respectively. This leads to a suboptimal case that after migration, j is the outmigrating controller and i is immigrating controller. And the distribution of controllers' loads is still imbalanced. However, a bigger δ does not guarantee a better result, such as the case of $\delta = 1.2$ plotted in Fig. 3. This is due to the fact that when δ is too big, some controllers are not selected as the controllers to migrate as their load diversity is within the limits of δ.

To analyze the factors that may affect the performance of the algorithms, we simulate the algorithms under different network environments as Fig. 4 and Fig. 5 show. In these simulations, the inputs of all methods at any time instant keep unchanged.

Fig. 4 shows the results with different numbers of controllers in a network. In this figure, each CV is obtained after

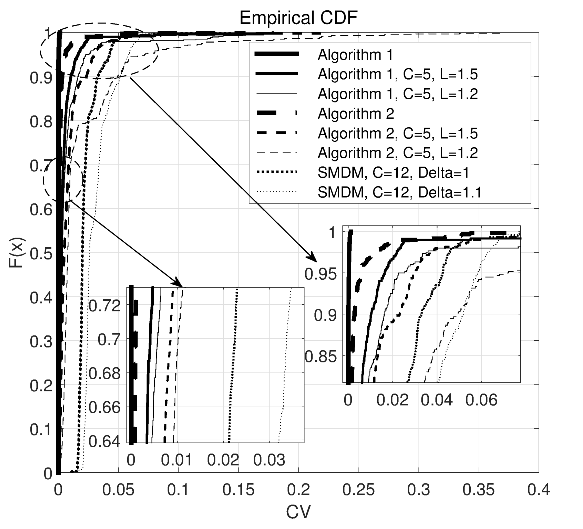

FIGURE 6. CDF of CV of different algorithms, with constraints of capacity and latency.

abundant iterations, i.e., after the algorithms achieve convergence. The y-axis of Fig. 4 is plotted at *log*-scale because the values of SMDM are much greater than that of our proposed methods. We can conclude from this figure that the number of controllers has little influence on the final state of the network in our proposed methods. That is because, after sufficient iterations, the final result of our algorithm depends on the network topology and the locations of controllers. Moreover, the results of the proposed algorithms are typically in the order of the power minus four and minus five. They are so small that the differences between these points are visually indistinguishable. At the same time, we can see that our algorithms obtain a more balanced state than SMDM, regardless of the number of controllers. Fig. 5 plots CVs with different sizes of network. All of these networks have 3 controllers and the topologies are generated as 4-regular graphs. It shows that with the increase of network size, Algorithm 1 has lower CV. And the CV of Algorithm 1 is critically reduced when network size is getting larger.

By this means, we can conclude that our algorithms can make it difficult for adversaries to identify strategically important controllers, as the algorithms obtain the lower CV and the accumulative loads of controllers are similar with each other.

We evaluate the average computational time of the algorithms at each iteration by running the algorithms repeatedly. Table 1 shows the detail of our evaluation results. We can conclude from this table that Algorithm 2 is far more time-efficient against the growth of network size and the number of controllers, which makes it suitable to be applied to large-scale networks.

As mentioned in Sections IV and V, our proposed methods can also deal with the situation with network constraints. Fig. 6 shows the CDF of CV of the algorithms restrained by different constraints. We represent 'C' as the upper bound

TABLE 1. Average computational time (*s*) for different algorithms.

Algorithms	Number of controllers	Network size($\|\mathcal{N}\|$)				
	$\|\mathcal{P}\|$	20	30	40	50	100
Algorithm 1	3	3.8284	78.6888	3351.8	4105.2	\
	5	139.4414	4412.7	\	\	\
	10	156.3347	6806.5	\	\	\
Algorithm 2	3	0.1451	0.2757	0.3606	0.3979	0.4041
	5	0.0932	0.1801	0.3058	0.4478	0.5507
	10	0.1141	0.1296	0.1661	0.2343	0.6868
SMDM method	3	0.0129	0.0206	0.0213	0.0209	0.0252
	5	0.0728	0.0848	0.0945	0.1013	0.0824
	10	0.3577	0.7123	1.2015	1.5041	1.8887

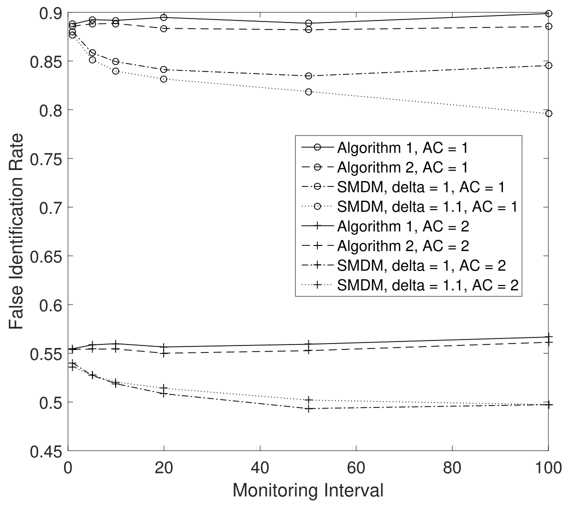

FIGURE 7. Rate of false identification for different algorithms.

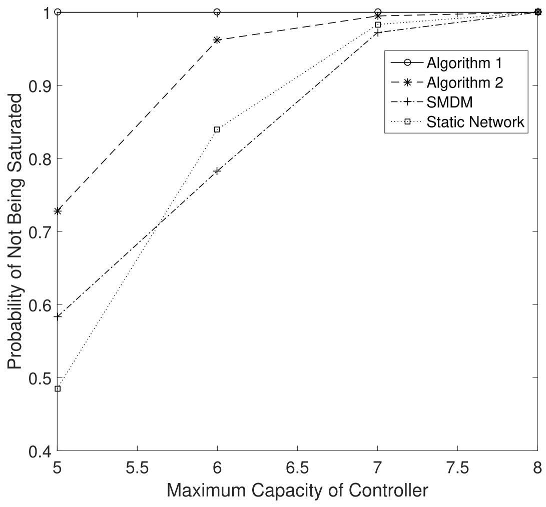

FIGURE 8. Probability of not being saturated for controllers.

of capacity, and ‘L’ as the maximum value of propagation latency. We amplify two parts of Fig. 6 to clearly display the results. Apparently, with looser limits of capacity and latency, the algorithms can obtain better results. And under the same constraint, Algorithm 1 obtains lower CV than other algorithms. This is because the well-designed of Algorithm 1, which always gets the optimal solutions within the limit of constraints.

To evaluate the effectiveness of our methods against reconnaissance and saturation attacks, we further evaluate our methods with different attack models. Firstly, we consider our methods against reconnaissance attacks. As mentioned in Section III-B, we assume the worst-case reconnaissance attack, i.e., the adversaries have already identified all strategically important controllers by monitoring their traffic. In this case, our methods can decrease the difference between the controllers, thereby disguising the strategically important controllers.

Particularly, we evaluate our methods using false identification rate, which defines the number of controllers that are falsely identified by adversaries over the total number of controllers. We also take the capability of attackers into consideration, i.e., “AC” in Fig. 7. We set the x-axis of this figure as the monitoring interval, i.e., the number of time instants that the adversaries keep monitoring the network. The y-axis is set as the mean of false identification rates during multiple monitoring intervals. In this figure, the different values of attack capability mean the different numbers of controllers an adversary can recognize during one monitoring interval. From Fig. 7, we see that with the growth of attack capability, adversaries can identify controllers more accurately. Specifically, for algorithm 1, after 20 monitoring intervals, adversaries can be confused and falsely identify controllers at the rate of 0.8947, when $AC = 1$. When the attack capability increases to two, the false identification rate falls to 0.5565. Given the capability of attackers, our proposed methods can effectively confuse the adversaries and decrease the number of controllers they identify. For example, when attackers can recognize two controllers at a time, after 50 monitoring intervals, the average false identification rates of Algorithms 1 and 2 are 0.5593 and 0.5527, respectively. As to the SMDM method, the false identification rates go to 0.4933 and 0.5020 when δ is 1 and 1.1.

Furthermore, we evaluate the effectiveness of our methods against saturation attacks, as shown in Fig. 8. In this simulation, we assume that the adversaries keep injecting flows that the switches cannot match according to their flow tables into network to saturate controllers. We define the probability that the controllers are not saturated during simulation. The inputs of all simulations in Fig. 8 remain the same. We see that with the increase of controller capacity, the probability of the controllers not being saturated is increasing. It is worth noting that Algorithm 1 always keeps the controllers under the restriction of their capacities, that is to say that the controllers cannot be saturated when we exploit this algorithm.

VII. CONCLUSION

In this paper, we propose an elastic switch migration model, which is based on three-dimensional earth mover's distance algorithm, to protect strategically important controllers from reconnaissance and saturation attacks in SDN. This scheme ensures optimal allocations of switches and minimal difference between controllers. A heuristic method is also designed to reduce the computational complexities, enhancing the scalability of switch migration for large-scale networks. Simulation results show that our proposed methods can significantly disguise important controllers against reconnaissance or eavesdropping by reducing the differences between controllers. Saturation attacks can also be prevented, as the switches are always migrated from highly-loaded controllers to lightly-loaded controllers and within the limits of capacities of controllers.

REFERENCES

[1] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. OSDI*, vol. 10. 2010, pp. 1–6.

[2] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[3] 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.

[4] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. ACM*, 2012, pp. 7–12.

[5] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against sdn controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2015, pp. 77–81.

[6] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[7] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–2.

[8] 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.

[9] W. Liang, X. Gao, F. Wu, G. Clien, and W. Wei, "Balancing traffic load for devolved controllers in data center networks," in *Proc. Global Commun. Conf. (GLOBECOM)*, 2014, pp. 2258–2263.

[10] G. Cheng, H. Chen, H. Hu, and J. Lan, "Dynamic switch migration towards a scalable SDN control plane," *Int. J. Commun. Syst.*, vol. 29, no. 9, pp. 1482–1499, 2016.

[11] 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.

[12] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 1, pp. 4–17, Mar. 2015.

[13] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, Jan. 2015.

[14] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Oct. 2013.

[15] 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)*, 2014, pp. 171–177.

[16] N. I. G. Dharma, M. F. Muthohar, J. D. A. Prayuda, K. Priagung, and D. Choi, "Time-based DDoS detection and mitigation for SDN controller," in *Proc. 17th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Aug. 2015, pp. 550–553.

[17] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *Proc. Int. Conf. IEEE Inf. Netw. (ICOIN)*, Aug. 2015, pp. 309–311.

[18] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Comput. Netw.*, vol. 81, pp. 308–319, Mar. 2015.

[19] Y. E. Oktian, S. Lee, and H. Lee, "Mitigating denial of service (DoS) attacks in OpenFlow networks," in *Proc. Int. Conf. IEEE Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2014, pp. 325–330.

[20] M. Belyaev and S. Gaivoronski, "Towards load balancing in SDN networks during DDoS-attacks," in *Proc. 1st Int. Sci. Technol. Conf. (Modern Netw. Technol.)*, 2014, pp. 1–6.

[21] S. Jain *et al.*, "B4: Experience with a globally deployed software defined wan," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[22] *OpenFlow Switch Specification Version 1.4.0*, Open Networking Foundation, Oct. 2013. [Online]. Available: https://www.opennetworking.org/

[23] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," *Hot-ICE*, vol. 12, pp. 1–6, Apr. 2012.

[24] P. Phaal, S. Panchen, and N. McKee, *Inmon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*, document RFC 3176, IETF, 2001.

[25] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[26] D. Zhang and G. Lu, "Evaluation of similarity measurement for image retrieval," in *Proc. Int. Conf. Neural Netw. Signal Process.*, vol. 2. 2003, pp. 928–931.

[27] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook Appl. Optim.*, vol. 2, pp. 65–77, Jan. 2002.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python Sci. Conf. (SciPy)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[29] I. ILOG Enligne. (2012). *Cplex Optimizer*. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer

YANG ZHOU received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012, where she is currently pursuing the Ph.D. degree with the School of Cyberspace Security. She visited CSIRO, Australia, in 2015. Her main research interests lie in software-defined networking, optimization, and cybersecurity defense.

KANGFENG ZHENG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications in 2006. He is currently an Associate Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include network security and network data analysis.

WEI NI (M'09–SM'15) received the B.E. and Ph.D. degrees in electronics engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. He was a Research Scientist and a Deputy Project Manager with the Bell Labs R&I Center, Alcatel/Alcatel-Lucent, from 2005 to 2008, and a Senior Researcher with Devices R&D, Nokia from 2008 to 2009. He is currently a Senior Scientist, a Team Leader, and a Project Leader with CSIRO, Australia. He also holds adjunct positions with the University of New South Wales, Macquarie University, and the University of Technology Sydney. His research interests include optimization, game theory, graph theory, and their applications to network and security.

Dr. Ni served as a Program Committee Member for CHINACOM 2014 and a TPC Member for the IEEE ICC'14 Workshop on body area networks, ICCC'15, EICE'14, and WCNC'10. He served as the Student Travel Grant Chair for WPMC 2014, the PHY Track Co-Chair for the IEEE VTC-Spring 2016, and the Publication Chair for BodyNet 2015. He has been serving as the Track Chair for VTC-Spring 2017 and the Secretary for the IEEE NSW VTS Chapter since 2015. He has been serving as an Editor for the *Hindawi Journal of Engineering* since 2012.

REN PING LIU (M'09–SM'14) was a Principal Scientist of CSIRO, where he leads wireless networking research activities. He joined the University of Technology Sydney as a Professor of networking technologies with the School of Computing and Communications in 2016. He specializes in protocol design and modeling, and has delivered networking solutions to a number of government agencies and industry customers. He has over 100 research publications, and has supervised over 30 Ph.D. students. His research interests include Markov analysis and QoS scheduling in WLAN, VANET, IoT, LTE, 5G, SDN, and network security.

He received the B.E. (Hons.) and M.E. degrees from the Beijing University of Posts and Telecommunications, China, and the Ph.D. degree from the University of Newcastle, Australia. He is the Founding Chair of the IEEE NSW VTS Chapter. He served as a TPC Chair for BodyNets2015, ISCIT2015, and WPMC2014, and as an OC Co-Chair for VTC2017-Spring, BodyNets2014, ICUWB2013, ISCIT2012, and SenSys2007. He served on the Technical Program Committee in a number of the IEEE conferences.