An Efficient Approach to Robust SDN Controller Placement for Security

Shu Yang, Laizhong Cui, *Senior Member, IEEE*, Ziteng Chen, Wei Xiao

***Abstract*—Security is one of the critical issues in traditional networks. Software-Defined Networking (SDN) improves the security aspect by separating the control plane and the data plane of networks. To improve the performance of SDN, researchers have designed many advanced controller prototypes and considered the controller placement problem. However, link failures are critical security issues in networks and greatly impact SDN's security. The controller placement problem for link failures is still challenging today. In this paper, we study the SDN controller placement problem for single-link and multi-link failures, respectively. For single-link failures, we develop a heuristic algorithm to address the controller placement problem. For multi-link failures, we introduce the Monte Carlo Simulation to reduce the computational overhead. We conduct experiments with real network topologies, and the simulation results show that the heuristic algorithm can save significantly more time than the optimal algorithm, while achieving good performance.**

***Index Terms*—Software-defined networking, controller placement problem, link failure**

I. Introduction

With the development of communication technology, more and more computing devices are joining and interacting via the Internet. However, security issues such as robustness, reliability, and misbehaving attacks are ongoing challenges in networks. Due to the characteristic of distributed control without reliable authorities or operators, networks are vulnerable to attacks and failures. The routing and forwarding strategies of traditional networks are inflexible, which cannot adapt to the actual network traffic or the real user demands [1]. Therefore, a novel network architecture is necessary to solve the security issues in traditional networks.

Software-Defined Networking (SDN) [2] is an important research topic in both industry and academia. Different from traditional networks, the control plane of SDN is decoupled from the data forwarding plane, such that the data forwarding plane will handle the packets according to the rules from the control plane. In SDN, the operators can easily modify the flow tables in switches, and computing devices can act as controllers, which simplifies the management and the deployment of SDNs. OpenFlow [2] is the most widely used SDN protocol. For OpenFlow-based controllers, there are two interfaces: 1) the northbound interface that provides information for applications; 2) the southbound interface that provides communication with the switches [3]. By deploying OpenFlow-based switches and controllers, we are able to implement a secure SDN that satisfies our demands.

In the security aspect, SDN provides secure and reliable services compared with traditional networks. Firstly, SDN can detect and prevent misbehaviors from adversaries by dropping malicious flows dynamically. Secondly, thanks to the programmable interfaces and the separation of the control plane and the data plane, SDN developers and operators can design their customized strategies and applications to defend against the attacks [4]. Thirdly, the routing and forwarding devices can be added or removed quickly, making SDN more flexible and resilient to device failures.

However, there are some challenging security issues in SDN [5], [6]. For instance, link failures are critical security issues in SDN, as the network operation relies on the link connectivity. Researchers have studied malicious behaviors to attack the links in SDN, such as the Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks [7], [8]. Ahmad *et al.* in [9] also pointed out that link failures can be caused by the lack of SDN security management. To solve the single point of failure in SDN, numerous approaches have been proposed. Among them, the distributed architecture is considered a promising solution [10], [11], [12], where multiple SDN controllers are allowed to be placed in networks. They divide the network into multiple domains, each of which is handled by an individual controller. Compared with traditional centralized networks, the distributed SDN architecture is more resilient to the single point of failure.

In the distributed SDN architecture, deploying multiple controllers properly continues to pose a challenging task. Heller *et al.* in [13] were the first to propose the SDN controller placement problem, showing that different controller placement schemes will affect the SDN performance. In the security aspect, numerous SDN controller placement schemes were proposed to improve security and robustness [14], [15]. However, the SDN controller placement problem with the occurrence of link failures is still challenging. Guo *et al.* in [16] considered the controller placement problem for single-link failures, but ignored the multi-link failures. In a real network environment, especially in large network topology, there is a portion of multi-link failures affecting security [17]. Therefore, the SDN controller placement problem for multi-

This work was supported in part by the National Key Research and Development Plan of China under Grant 2018YFB1800302, in part by the National Natural Science Foundation of China under Grants 61772345 and 61902258, in part by Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grant JCYJ20190808142207420 and the Pearl River Young Scholars funding of Shenzhen University. *(Corresponding author: Laizhong Cui)*

S. Yang, L. Cui, Z. Chen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, PR.China and with the Guangdong Laboratory of Artificial-Intelligence and Cyber-Economics (SZ), Shenzhen University.

W. Xiao is with the Blockcloud Technology, Shenzhen, PR.China.

link failures should be considered carefully.

In this paper, we solve the SDN **c**ontroller **p**lacement problem for **s**ingle-**l**ink **f**ailures (CPSLF) and **m**ulti-**l**ink **f**ailures (CPMLF), respectively. For CPSLF, only one link may fail at a moment, and we develop the **O**ptimal **P**lacement **A**lgorithm (OPA) and the **G**reedy **P**lacement **A**lgorithm (GPA). Since the controller placement problem is shown to be NP-hard [13], solving CPSLF optimally is also NP-Hard, as OPA needs exponential computation time to consider all the possible controller placement schemes with different link failure scenarios. Consequently, we develop the greedy placement algorithm (GPA), where the controllers are greedily and iteratively placed to achieve the lowest propagation latency. As for CPMLF, many more link failures events may happen, which will lead to non-polynomial computation time. Therefore, we introduce the Monte Carlo Simulation to simulate the failure scenarios according to the link failure rates. By conducting the Monte Carlo Simulation for a sufficient number of times, we can emulate those multi-link failure scenarios with high occurrence probabilities. The greedy algorithm will place the controllers iteratively and greedily, and output the controller placement schemes. We conduct experiments with real network topologies including Internet2 [18] and Cernet2 [19]. Experimental results show that the greedy algorithm and the Monte Carlo Simulation are suitable and secure in different link failure situations.

Our contributions in this work are summarized as follows:

- For single-link failures, since solving the optimal controller placement problem is NP-Hard, we develop a heuristic algorithm where the controllers are greedily placed for lower overheads.
- We address the controller placement problem with multi-link failures based on link failure rates. To simulate the multi-link failure scenarios efficiently, we employ the Monte Carlo Simulation with the information of the link failure rates, and develop the greedy algorithm based on the Monte Carlo Simulation.
- We perform evaluations with real network topologies, and the results show that the greedy solution and the Monte Carlo Simulation work well with the controller placement problem in different link failure situations.

The rest of this paper is organized as follows. In Section II, we will introduce the background and related work of security for SDN and the controller placement problem. Motivations and notations of this paper will be given in Section III. Then we will present the controller placement problems for single-link failures in Section IV. Section V will discuss the controller placement problems for multi-link failures. We conduct experiments and present the simulation results in Section VI. Finally, the conclusion and the future work will be discussed in Section VII.

II. Related Work

A. Security for SDN

The emergence of SDN improves network security. SDN enjoys granular security benefits, as the network operators can handle potential misbehaving segments at a granular level. The SDN controller has a centralized view, where network traffic and data packets can be collected and analyzed by operators [20]. Meanwhile, the network devices can be dynamically updated (e.g., add or remove) without getting access to the physical devices, so that stability is guaranteed. Besides, many security enhancements are employed in SDN devices. For example, the Transport Layer Security (*TLS*) is utilized in OpenFlow switches [21], which provides the authentication mechanisms between controllers and switches. Bosshart *et al.* in [22] proposed a new compiler and programming language that guarantees SDN security at the code level. Moreover, thanks to the programmability, researchers can develop their on-demand security expansions (including applications, routing and forwarding policies, etc.) to enhance SDN security further.

However, SDN security is still an ongoing challenge. For example, SDN suffers from DoS and DDoS attacks [7], malicious applications [23], system-level security issues [8], [24], etc. As one of the security issues, link failures also influence the SDN greatly. They can be caused by different network attacks [7], [8], and the lack of the security management [9]. They influence several network performance metrics such as delay, packet loss, throughput, and reliability. Thus, SDN security and efficiency highly rely on countermeasures to the link failures. In the centralized SDN architecture, the single SDN controller is easily affected by the single point of failure caused by link failures, which can lead to network crashes. Researchers have proposed a few countermeasures to link failures. Authors in [10], [11], [12] developed the distributed SDN architecture, where the network is divided into multiple domains, each of which is handled by an individual controller. However, they ignored the controller placement schemes in link failure situations. Link failure detection and recovery methods are proposed in [25], [26], but they are not adaptive to multi-link failures. Therefore, we will study SDN security for link failures, especially for multi-link failures.

B. Controller Placement Problem

Heller *et al.* in [13] proposed the SDN controller placement problem, and pointed out that different controller placement schemes will influence the network performance. For distributed SDN architectures, different placement schemes with multiple controllers greatly impact the network latency, resilience, and tolerance to network failures [27]. Numerous controller placement schemes have been proposed since then to improve SDN security. Hu *et al.* in [28] considered the controller failures, and proposed the controller fault-tolerant placement schemes. Ros *et al.* in [14] introduced a node fault-tolerant controller placement problem and developed a heuristic algorithm with at least five-nines reliability. Hock *et al.* in [15] considered the fault-tolerant controller placement schemes. They provided a Pareto-based optimal placement algorithm *(POCO)* and extended it to large-scale networks. Furthermore, the SDN controller placement problem is studied and solved in cellular networks [29], wireless networks [30], and wireless edge networks [31].

However, the controller placement problem for link failures is still challenging, especially for multi-link failures. Guo

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2994837, IEEE Transactions on Network and Service Management

et al. in [16] discussed the controller placement problem for single-link failures, and developed the heuristic-based controller placement scheme resistant to single-link failures, but ignored the multi-link failure situations. Zhong *et al.* in [32] also considered single-link failures. They managed the failed links and proposed a min-cover-based SDN controller placement scheme with minimal required controllers. Similar single-link failure situations are also considered and solved in [33], [34] and [26]. However, there is a portion of multi-link failures in large network topologies with hundreds of nodes and links. Thus, the SDN controller placement schemes should be resistant to multi-link failures in large networks [17]. Although [35], [36], [37] addressed the SDN controller placement problem for multi-link failures, they cannot adapt to the actual link failure rates in various networks. To enhance the reliability and solve the security issues caused by link failures, we will study the SDN controller placement problem for both single-link and multi-link failures, respectively, which is also a link fault-tolerant controller placement problem.

III. Motivation and Notation

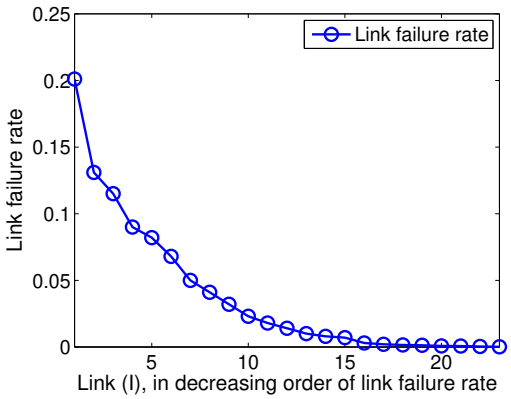


Fig. 1: Per-link failure rate, in descending order.

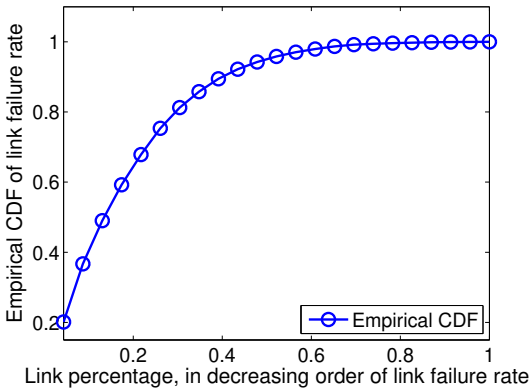


Fig. 2: The CDF of link failure rate.

A. Data trace study

We investigate the real data traces of the links of China Education and Research Network 2 (Cernet2) [19] consisting of 21 nodes and 23 physical links with IPv6 connectivity. We have collected the link failure rates of Cernet2, which are presented in decreasing order in Fig. 1, while the cumulative distribution function (CDF) of link failure rates is shown in Fig. 2. We can see that the failure rate distribution follows the "long tail" feature. About 60% of link failures in Cernet2 are caused by some specific vulnerable links, e.g., 4 out of 23 links. The "long tail" feature can also be observed in other real networks [38]. Therefore, when link failures happen, the exhaustive computation for all the possible multi-link failure scenarios is unnecessary. This inspires us to employ other methods to solve the controller placement problem when link failures happen.

B. Overhead

Generally, for the SDN controller placement problem, there are two kinds of overheads. The first is the computational overhead. For the network topology in a dynamic environment, we should take into account the computational overhead to achieve a good performance in real-time. For single-link failures, the computation cost for an optimal controller placement scheme is $O(|E| \times NFS)$, where $|E|$ is the number of physical links, and *NFS* is the computation cost of traversing all the possible controller placement schemes. The optimal controller placement problem is NP-hard [13], which will bring a high computational overhead to network operators. For multi-link failures, the computation load will be much higher. The second is the control overhead. In the distributed SDN architecture, each controller will manage its domain, and thus the capacity of controllers will affect the network performance. In this paper, we will study the overheads for the SDN controller placement problem, and evaluate the network performance with controllers that have the finite and infinite capacity, respectively.

C. Notation

We introduce the notations as referred to in [16]. The network is denoted by $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of n nodes and $E = \{e_1, e_2, \cdots, e_m\}$ is the set of m physical links between the nodes. We assume that there are k controllers in the network, and they are represented by $\Theta = \{\theta_1, \theta_2, \cdots, \theta_k\}$. Let $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$ denote a controller placement scheme, where $\psi(\theta_i)$ is defined as the location of controller θ_i. Given k controllers, there will be a finite set of $\binom{n}{k}$ controller placement schemes, and we denote the set as $\mathcal{P}$.

For brevity, we use c_i to infer whether link e_i works or fails, which is denoted as:

$$c_i = \begin{cases} 1, & \text{if } e_i \text{ fails.} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Meanwhile, we denote the set of network state as $\mathcal{G} = \{g_1, g_2, \cdots, g_l\}$, where $l = 2^m$. We also use ω_g^e to represent the state of link e in g, which can be defined as $g = \{\omega_g^{e_1}, \omega_g^{e_2}, \cdots, \omega_g^{e_m}\}$.

Moreover, we use F to denote different link failure situations, and the cardinality of F represents the number of simultaneously failed links in the network. For example, $|F| = 1$ indicates a single-link failure situation, and $|F| \geq 2$ indicates the multi-link failure situations. We assume that no more than K links fail simultaneously, and thus the cardinality of F satisfies $1 \leq |F| \leq K$.

Additionally, $d_{i,j}$ is defined as the shortest path latency between node i and node j. For a certain placement scheme $P \in \mathcal{P}$, the node-to-controller worst-case latency $L_{wl}(P)$ and the average-case propagation latency $L_{al}(P)$ are determined as (2) and (3):

$$L_{wl}(P) = \max_{i \in V} \min_{j \in P} d_{i,j} \tag{2}$$

$$L_{al}(P) = \frac{1}{n} \sum_{i \in V} \min_{j \in P} d_{i,j} \tag{3}$$

We assume that each switch connects to the closest controller. For simplicity, we will use the worst-case latency to measure the propagation latency, with our objective being to minimize the worst-case latency in different link failure situations.

IV. Controller Placement Problem for Single-link Failures

In this section we will study the controller placement problem for single-link failures (CPSLF). More specifically, it is the link failure situation $|F| = 1$ where only one link may fail at a moment. The notations for CPSLF are listed in Table I.

A. Problem

For CPSLF, as every link may fail at some point, there are m different link failure scenarios, and we define them as $\Pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$, where $\pi_i \in \Pi (1 \leq i \leq m)$ is the failure scenario when $c_i = 1$. Moreover, we define the failure rate of link e_i (i.e., the occurrence probability of failure scenario π_i) as:

$$\Delta_i = \frac{t_i}{t_{duration}} \tag{4}$$

where t_i is the duration of failure scenario π_i in a given period $t_{duration}$. For Cernet2, t_i follows the failure rate shown in Fig. 1. We assume Δ_i is independent of Δ_j $(1 \leq i \neq j \leq m)$, and that their failure duration periods do not overlap.

Given a network topology G in a single-link failure situation $|F| = 1$, the problem is to search for the controller placement scheme $P \in \mathcal{P}$ that minimizes the worst-case propagation latency $L_{wl}(P)$. Formally,

$$\min L_{wl}(P) \tag{5}$$

$$\text{s.t.} \quad |F| = 1. \tag{6}$$

TABLE I: Notation list for CPSLF

Notation	Meaning
V	set of nodes
E	set of edges
n	number of nodes
m	number of physical links
e_i	i^{th} physical link
π_i	failure scenario when e_i fails
c_i	status of link e_i
Π	set of failure scenarios
Δ_i	failure rate of link e_i
θ_i	i^{th} controller
Θ	set of controllers
$\psi(\theta_i)$	location of θ_i
$d_{i,j}$	shortest path latency between node i and node j
$L_{wl}(P)$	node-to-controller worst-case propagation latency of placement scheme P
F	link failure situation

B. Optimal Placement Algorithm for CPSLF

To search the optimal placement scheme with the lowest worst-case latency, we develop the Optimal Placement Algorithm (OPA) for CPSLF. The main idea of OPA is to traverse all placement schemes for every link failure scenario. For a certain scheme $P_i \in \mathcal{P}$, OPA will compute the current worst-case propagation latency $L_{wl}^{\pi_r}(P_i)$ for failure scenario π_r, and multiply its corresponding occurrence probability. Then it will take the sum and get the worst-case latency of P_i. Formally, the worst-case propagation latency of controller placement scheme P_i is computed by (7):

$$L_{wl}(P_i) = \sum_{\pi_r \in \Pi} L_{wl}^{\pi_r}(P_i) \times \Delta_r \tag{7}$$

After finishing the computation process of $\binom{n}{k}$ placement schemes, OPA will select the optimal one $P \in \mathcal{P}$ with the lowest worst-case latency.

Details of OPA for CPSLF are given in Algorithm 1. From lines 2 to 8, the $\binom{n}{k}$ possible controller placement schemes will be considered, and OPA will select the optimal one among them. Lines 3 to 6 are the latency computation process of a scheme P_i. For each failure scenario, as the network topology information will change, the shortest path and the latency information should be updated correspondingly. Therefore, in line 4, the algorithm will recalculate the shortest paths between the nodes for the failure scenario π_j. The algorithm will compute the worst-case latency and multiply the corresponding occurrence probability. Finally, the algorithm will choose the optimal one P among them.

It can be obtained that the time complexity (also the computational overhead) of OPA is bounded by $O(m \times \binom{n}{k})$, which will cost non-polynomial runtime.

C. Greedy Placement Algorithm for CPSLF

Given a network topology, OPA will perform computation for the exhaustive search of all the possible placement schemes. Although OPA guarantees that it will find the optimal placement scheme, the computational overhead and the resource requirement will increase rapidly with the size of the search space.

Algorithm 1: Optimal Placement Algorithm for CP-SLF

Input: $G = (V, E), k$
Output: $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$

```
1  begin
2      for i ← 1 to (n choose k) do
3          for j ← 1 to m do
4              Recalculate the shortest paths in link failure
                scenario π_j.
5              Compute the worst-case latency of P_i and
                multiply the Δ_j.
6          end
7          Add up the worst-case latency of P_i.
8      end
9      Choose the optimal placement scheme with the
        lowest worst-case latency and assign to P.
10 end
```

To reduce the computational overhead, we design the greedy placement algorithm (GPA) for CPSLF. The main idea of GPA is to place controllers greedily and iteratively. In each round, GPA will greedily place a controller on a specific node to minimize the worst-case latency. After k iterations, k controllers will be greedily placed, and we will output the final placement scheme P.

Details of GPA for CPSLF are given in Algorithm 2. The input of GPA is the network $G = (V, E)$ and the number of controllers k. The output is the placement scheme $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$. In GPA, S is the set of the controllers that have already been added to the network. At the beginning of GPA, the corresponding variables need to be initialized. Lines 8 to 18 are the greedy computation process. Firstly, in line 10, the algorithm will recalculate the shortest path latency between the nodes for the link failure scenario π_j. Secondly, for controller θ_i, it will have $|V - S|$ different alternatives. GPA will place θ_i on a node to minimize the worst-case latency based on S, and the computation process is similar to OPA. After finding the optimal place of θ_i, we update $\psi(\theta_i)$, add $\psi(\theta_i)$ to the placement scheme P, and add θ_i to the set S. Finally, we will output the controller placement scheme P.

Theorem 1. *The time complexity of GPA for CPSLF is* $O(mnk)$.

Proof. Lines 9 to 13 are the greedy computation process, whose time complexity is bounded by $O(mn)$. Since GPA needs to place k controllers, the time complexity of Algorithm 2 is $O(mnk)$. □

In [13], Heller *et al.* proved that solving the controller placement problem is equivalent to solving the k-center problem, which is NP-Hard. We will show that GPA has a competitive ratio of 2.

Theorem 2. *The greedy algorithm for the controller placement problem has a competitive ratio of 2.*

Algorithm 2: Greedy Placement Algorithm for CPSLF

Input: $G = (V, E), k$
Output: $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$

```
1  begin
2      for i ← 1 to k do
3          ψ(θ_i) ← ∅;
4      end
5      S ← ∅;
6      i ← 1;
7      j ← 1;
8      while i ≤ k do
9          while j ≤ m do
10             Recalculate the shortest paths in link failure
                scenario π_j.
11             The algorithm will place θ_i on n nodes,
                respectively, and compute the
                corresponding worst-case latency and
                multiply the Δ_j.
12             j ← j + 1;
13         end
14         Choose the one with the lowest worst-case
            latency and update ψ(θ_i);
15         P ← P + ψ(θ_i);
16         S ← S ⋃ {θ_j};
17         i ← i + 1;
18     end
19 end
```

Proof. We assume that OPA and GPA are the worst-case latency performance computed by OPA and GPA, respectively. Suppose P^* and P are the controller placement schemes computed by OPA and GPA, respectively. For $v \in P^*$, we denote $N(v)$ as the node set in its domain. For simplicity, we let $d_{u,P} = \min_{v \in P} d_{u,v}$ for a node $u \in V$.

Case 1: One node in P is selected from each domain of P^*, i.e., $\forall v \in P^*, N(v) \cap P \neq \emptyset$. In the domain of $v \in P^*$, we let $w \in P \cap N(v)$, where w is the controller selected by GPA. Therefore, for a node $u \in V$, by the triangle inequality, we have $GPA = d_{u,P} \leq d_{u,w} \leq d_{u,v} + d_{v,w} \leq 2 \cdot OPT$.

Case 2: At least two nodes in P are selected within the same domain, i.e., $\exists v' \in P^*$, $|N(v') \cap P| \geq 2$. Suppose i and j are the nodes computed by GPA iteratively, and $i, j \in N(v') \cap P$, where i is selected before j. Let P' be the controller set before selecting j. Therefore, by the triangle inequality, for a node $u \in V$, we have $GPA = d_{u,P} \leq d_{u,P'} \leq d_{j,P'} \leq d_{j,i} \leq d_{j,v'} + d_{v',i} \leq 2 \cdot OPT$. □

In practice, GPA can achieve sub-optimal latency performance, which will be demonstrated in Section VI. Meanwhile, GPA can significantly reduce the search space, especially for large network topology. For example, when $k = 5$, $n = 34$, $m = 42$, OPA's search space is $42 \times \binom{34}{5} = 11,686,752$, while GPA has only $5 \times 42 = 210$, which is much smaller than OPA. In large-scale networks, GPA's superiority on search space is more obvious. Hence, our proposed GPA is a time-efficient approach for CPSLF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2994837, IEEE Transactions on Network and Service Management

TABLE II: Notation list for CPMLF

Notation	Meaning
K	maximal value of failed links
t	number of simultaneously failed links
E_t	failure scenario when $c_{i_1} = c_{i_2} = \cdots = c_{i_p} = 1$
Π	set of failure scenarios
$\Delta_{i_1 i_2 \cdots i_p}$	occurrence probability of E_t

V. Controller Placement Problem for Multi-link Failures

In this section we will study the controller placement problem for multi-link failures (CPMLF). More specifically, it is the link failure situation $2 \leq |F| \leq K$ where at least 2 links fail simultaneously. We define E_t $(2 \leq t \leq K)$ as the multi-link failure event where $e_{i_1}, e_{i_2}, \cdots, e_{i_p}$ $(i_1 \neq i_2 \neq \cdots \neq i_p)$ fail simultaneously while other links work. The notations for CPMLF are listed in Table II.

A. Problem

In E_t, we define $\pi_{i_1 i_2 \cdots i_p} \in \Pi$ as the link failure scenario when $c_{i_1} = c_{i_2} = \cdots = c_{i_p} = 1$ and $i_1 \neq i_2 \neq \cdots \neq i_p$. Furthermore, $\Delta_{i_1 i_2 \cdots i_p}$ is defined as the occurrence probability of the failure scenario $\pi_{i_1 i_2 \cdots i_p}$ when $c_{i_1} = c_{i_2} = \cdots = c_{i_p} = 1$, which is computed by:

$$\Delta_{i_1 i_2 \cdots i_p} = \frac{t_{i_1 i_2 \cdots i_p}}{t_{duration}} \tag{8}$$

where $t_{i_1 i_2 \cdots i_p}$ is the occurrence duration of $\pi_{i_1 i_2 \cdots i_p}$ in a given period $t_{duration}$, and they also follow the failure rate for Cernet2 shown in Fig. 1 and Fig. 2. In this paper, we assume the links are independent, and thus the occurrence probability $\Delta_{i_1 i_2 \cdots i_p}$ is computed by:

$$\Delta_{i_1 i_2 \cdots i_p} = \Delta_{i_1} \times \Delta_{i_2} \times \cdots \times \Delta_{i_p} \tag{9}$$

where $\Delta_{i_1}, \Delta_{i_2}, \cdots, \Delta_{i_p}$ are failure rates of link $e_{i_1}, e_{i_2}, \cdots, e_{i_p}$, respectively.

For CPMLF, given a network topology G in multi-link failure situation $2 \leq |F| \leq K$, the problem is to search for the controller placement scheme $P \in \mathcal{P}$ which satisfies the link failure situation F and minimizes the worst-case propagation latency $L_{wl}(P)$. Formally,

$$\min L_{wl}(P) \tag{10}$$

$$\text{s.t.} \quad 2 \leq |F| \leq K. \tag{11}$$

B. Optimal Placement Algorithm for CPMLF

To search the optimal controller placement scheme, we develop the Optimal Placement Algorithm (OPA) for CPMLF. The main idea of OPA for CPMLF is similar to CPSLF. All the possible controller placement schemes and multi-link failure scenarios will be considered, and OPA will choose the optimal scheme with the lowest worst-case latency. For CPMLF, we assume the maximum number of simultaneously failed links is K. Therefore, the number of all the possible multi-link failure scenarios is $\sum_{t=2}^{K} \binom{m}{t}$.

Details of OPA are given in Algorithm 3. Similarly, in line 2, the optimal algorithm needs to traverse $M = \binom{n}{k}$ different placement schemes. Lines 3 to 8 are the latency computation process. Firstly, the algorithm will recalculate the shortest paths between the nodes as well. For a placement scheme, OPA needs to compute the worst-case latency and multiply the occurrence probability of the corresponding failure scenario. After considering $\sum_{t=2}^{K} \binom{m}{t}$ possible multi-link failure scenarios, we will add up all the worst-case latency of placement scheme P_i according to (7). Finally, in line 11, the algorithm will output the optimal scheme P with the lowest worst-case latency.

Algorithm 3: Optimal Placement Algorithm for CPMLF

Input: $G = (V, E), k, K$
Output: $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$

```
1  begin
2      for i ← 1 to (n k) do
3          for t ← 2 to K do
4              for j ← 1 to (m t) do
5                  Recalculate the shortest paths in link
                     failure scenario π_{i1 i2 ... ip}.
6                  OPA will compute the worst-case
                     latency and multiply the Δ_{i1 i2 ... ip}.
7              end
8          end
9          Add up all the worst-case latency of P_i.
10     end
11     Choose the placement scheme P with the lowest
         worst-case latency.
12 end
```

Solving CPMLF optimally is NP-Hard because OPA needs to compute two combinatorial numbers in line 2 and line 4. In large-scale networks, the search space of OPA will increase exponentially. Hence, we need to develop a greedy solution to address CPMLF in polynomial time.

C. Greedy Placement Algorithm based on Monte Carlo Simulation

Different from CPSLF, besides the $\binom{n}{k}$ placement schemes, there are many more possible multi-link failure scenarios for CPMLF. For example, given a network with 42 physical links, if 3 and 5 links fail simultaneously, there will be $\binom{42}{3} = 11,480$ and $\binom{42}{5} = 850,668$ multi-link failure possibilities, respectively. Besides, the number of the controller placement schemes that need to be considered in the optimal algorithms is overwhelming. For example, given a network topology with 34 nodes, when the numbers of controllers are 5 and 10, there will be $\binom{34}{5} = 278,256$ and $\binom{34}{10} = 131,128,140$ possible controller placement schemes, which will cost much computational effort. As real networks usually reside in dynamic environment with changes (e.g., topology, number of users, device failures, etc.), the computation time and the resources are the limiting factors.

As we mentioned in Section III, each link will fail with a specific probability according to the statistical data. In multi-link failure situations, the network contains many link failure events, each of which will happen according to the occurrence probabilities and the link failure rates. Therefore, we will employ the Monte Carlo Simulation to simulate the multi-link failure scenarios to reduce the computational overhead.

1) Monte Carlo Simulation: Monte Carlo Simulation is widely used in optimization, numerical integration, and generating draws according to a probability distribution. The system contains different events, each of which has a probabilistic and stochastic interpretation. When we perform the Monte Carlo Simulation for a large enough number, N times, the simulation results will converge to the probability density, which will be very close to the real situations. Monte Carlo Simulation is effective to simulate problems consisting of complicated events, especially those that cannot be solved by other approaches.

Monte Carlo Simulation adopts repeated random sampling to obtain the statistical properties of the events. Random numbers are deployed to simulate complicated processes and multi-dimensional problems without analytical solutions and deterministic probabilities. In the Monte Carlo Simulation, random numbers with the deterministic distribution and high quality are critical in the simulation processes. To guarantee the high quality of the random numbers, we will use the random number generator provided by Matlab.

In multi-link failure situations, we will use the Monte Carlo Simulation to simulate different multi-link failure events in networks. We will collect and analyze the failure rate information of each link in advance, then perform the Monte Carlo Simulation for a sufficient number of times, so that those multi-link failure events with higher occurrence probabilities will be selected to compute the latency. In this case, the computation for the exhaustive search of multi-link failure events can be relaxed, which will improve the computation efficiency.

2) Greedy Placement Algorithm Based on Monte Carlo Simulation: In this subsection, we develop the Greedy Placement Algorithm (GPA) based on the Monte Carlo Simulation for multi-link failures. The main idea of our algorithms is, for every controller, the algorithm will emulate the multi-link failure scenarios for N times by the Monte Carlo Simulation according to the link failure rates, and compute the worst-case latency of the given placement scheme. Then the algorithm will select the placement scheme with the lowest latency, and place the controller on the specific node. After iterating k times, the k controllers will be greedily placed, such that the worst-case latency will be minimized greedily.

Details of GPA and the Monte Carlo Simulation process are presented in Algorithm 5 and Algorithm 4. In Algorithm 5, the variables need to be initialized at the beginning. P represents the location list of the controllers, which is also the output of the algorithm. $latencyList$ represents the list of N propagation latency results computed by the Monte Carlo Simulation, and S stores the controllers that have been placed in the network. Lines 5 to 13 are the greedy computation process, which is the same as the single-link failure situation. Each controller will be greedily placed on a node to minimize the worst-case latency iteratively. In line 6, the Monte Carlo Simulation is employed to compute the worst-case latency.

Algorithm 4 is the Monte Carlo Simulation process, which simulates the multi-link failure scenarios for N times, and outputs the latency results to Algorithm 5. It recalculates the shortest path latency first. Then in the h^{th} simulation process, the propagation latency $L_i^h(\theta_i = v_j)(1 \leq j \leq n, 1 \leq h \leq N)$ is computed for placing θ_i on node v_j. After N times, the average latency $L_i(\theta_i = v_j)$ can be obtained for placing θ_i on node v_j, and appended to $latencyList$ by (12):

$$L_i(\theta_i = v_j) = \frac{1}{N} \sum_{h=1}^{N} L_i^h(\theta_i = v_j) \tag{12}$$

After Algorithm 5 receives the latency list, we let $L_i(\theta_i = v_\pi) = \min(latencyList)$, where v_π means the node to place θ_i. Finally, the information on the placement list P and the controller set S is updated, and $latencyList$ is initialized for the next iteration. After k iterations, k controllers will be placed greedily, and we can get the placement scheme P.

Algorithm 4: MonteCarloSimulation(i, m, N)

```
begin
    for j ← 1 to m do
        Recalculate the shortest paths in this link
         failure scenario.
        for h ← 1 to N do
            Conduct the Monte Carlo Simulation and
             compute the worst-case latency
             L_i^h(θ_i = v_j) based on S;
        end
        Compute the average worst-case latency
         L_i(θ_i = v_j);
        latencyList ← latencyList + L_i(θ_i = v_j);
    end
    return latencyList;
end
```

Theorem 3. *The time complexity of GPA based on the Monte Carlo Simulation for CPMLF is* $O(knN)$.

Proof. In Algorithm 5, we need to repeat the process for k times from lines 5 to 13. In Algorithm 4, we need to perform the simulation for nN times. Thus, the time complexity of GPA for CPMLF is $O(knN)$. □

In [39], the authors noted that, when performing the Monte Carlo Simulation for a sufficient number of times, the events with specific probabilities can be simulated with high accuracy. Therefore, the multi-link failure scenarios can be simulated by the Monte Carlo Simulation precisely, and the latency performance of the algorithms also achieves at most twice as bad as the optimal algorithm, i.e., $GPA \leq 2 \cdot OPA$. Meanwhile, the size of the search space declines considerably. Hence, considering the limitation of real networks, operators can resort to GPA for less computational overhead.

Algorithm 5: Greedy Placement Algorithm Based on the Monte Carlo Simulation for CPMLF

Input: $G = (V, E), k, N$
Output: $P = \{\psi(\theta_1), \psi(\theta_2), \cdots, \psi(\theta_k)\}$

```
begin
    P ← ∅;
    S ← ∅;
    latencyList ← ∅;
    for i ← 1 to k do
        latencyList = MonteCarloSimulation(i, m, N);
        L_i(θ_i = v_π) = min(latencyList);
        place θ_i on node v_π;
        ψ(θ_i) ← v_π;
        P ← P + ψ(θ_i);
        S ← S ⋃ {θ_i};
        latencyList ← ∅;
    end
end
```

VI. Performance Evaluation

A. Simulation Setup

We evaluate the algorithms for single-link and multi-link failures with real network topologies, including Internet Topology Zoo [40], Internet2 [18] and Cernet2 [19].

1) **Topology**: Internet Topology Zoo is a topology repository that contains 261 different commercial network topologies. Developers can download the topology graphs and conduct the simulations on their PCs. Internet2 [18] is a commercial and nationwide backbone network in the United States, which contains 34 nodes and 42 high-speed links connecting the major cities in the USA. Cernet2 [19] is also a real network topology consisting of 21 nodes and 23 links with IPv6 connectivity, which is the largest pure-IPv6 backbone network. It connects more than 200 universities and institutions of the major cities in China. Internet2 and Cernet2 are large and mature network topologies in practice, and they are also widely adopted to simulate network environments.
2) **Failure Rate**: Link failure rates of Topology Zoo and Internet2 are generated with Weibull distribution. The probability density function (PDF) for the distribution is $p(x) = \frac{\alpha}{\lambda}(\frac{x}{\lambda})^{\alpha-1}e^{-(x/\lambda)^{\alpha}}$, where α represents the shape parameter, and λ represents the scale parameter, i.e., the characteristic life of an individual link. A smaller value of α will increase the varying extent of the failure rates, and a larger value of λ will decrease the failure rate of an individual link. For a real network topology, only a small set of links may fail frequently. Thus, the failure rate distribution in descending order follows the "long tail" feature, according to the realworld statistical data. To generate the "long tail" graph and meet the characteristics of links, we refer to [14] and set the default parameters of the Weibull distribution as $\alpha = 0.9$ and $\lambda = 15526$. To evaluate the performance of our proposed approaches in a worse scenario, we will increase the link failure rates compared with the default values, and prove that our placement schemes are efficient to cope with link failures in SDN. As for Cernet2, the link failure rates are obtained based on the real collected data shown in Section III.

We conduct the experiments in Matlab. The inputs of both algorithms are the information and the distance matrices of nodes. The computation processes are explained in Section IV and Section V. We will report the controller placement schemes computed by OPA and GPA with different numbers of controllers. We will evaluate the efficiency-related network metrics, such as the worst-case propagation latency, runtime, and cost-benefit ratios of the algorithms. We will also change the link failure rates by varying the values of α and λ of the distribution, and evaluate the performance of our approaches in different link failure situations. Additionally, to prove the robustness of our algorithms, we will estimate the *probability of network survival*, which is a critical security-related metric that measures the probability of our approaches surviving the link failure situations with different numbers of controllers.

In our experiments, the number of controllers k varies from 1 to 7. We point out that the value range of k makes sense, because in reality we want a small number of controllers to minimize the controller communication overhead. At the same time, a limited k can avoid excessive search space, so that we can focus on the relationship between the computational overhead and the network performance. By default, these controllers have the infinite capacity. We will also study the placement schemes computed by OPA and GPA with controllers that have the finite capacity, where the algorithms only output the feasible placement schemes that satisfy the capacity constraints. More specifically, for a controller placement scheme, the algorithms will check the capacity constraints of the controllers. If the placement scheme with the lowest worst-case latency is unsatisfied, it will be discarded, and the algorithms will proceed to search for other solutions. The search will stop when one feasible scheme that meets the capacity demands and achieves relatively best-latency is found. Moreover, to simulate enough multi-link failure scenarios, we will conduct the Monte Carlo Simulation for $N = 1,000,000$ times. In this case, the size of the search space and the maximal value of failed links are determined, such that OPA can also be handled in our computing devices.

Since the simulation results of single-link failures are similar to our previous work [16], we omit the simulation results of these and focus on the multi-link failure situations. The single-link failure results can be seen in [16].

B. Simulation Results of Multi-link Failure

1) Location of Controllers with infinite capacity: Fig. 3-Fig. 10 illustrate the controller placement schemes for multi-link failures. In these figures, the circle nodes denote the switches and the square nodes denote the controllers. If the switches are assigned to a specific controller, they are in the same color. For Internet2, in Fig. 3 and Fig. 7, the placement schemes computed by OPA at $k = 3$ are node 4, 8, 15, where node 4 and 15 are included in the placement scheme computed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2994837, IEEE Transactions on Network and Service Management

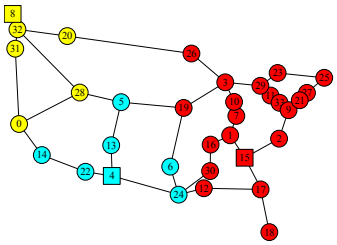


Fig. 3: Placement of Internet2 computed by OPA at $k = 3$.

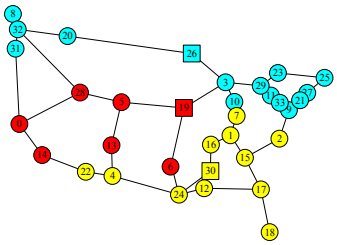


Fig. 4: Placement of Internet2 computed by GPA at $k = 3$.

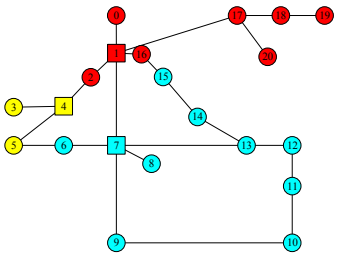


Fig. 5: Placement of Cernet2 computed by OPA at $k = 3$.

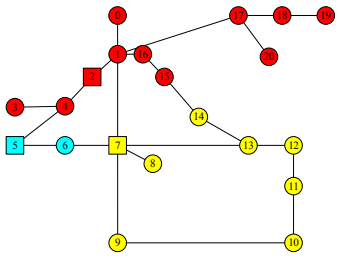


Fig. 6: Placement of Cernet2 computed by GPA at $k = 3$.

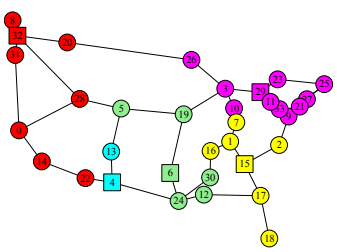


Fig. 7: Placement of Internet2 computed by OPA at $k = 5$.

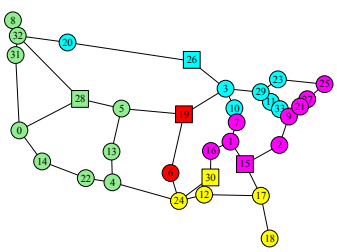


Fig. 8: Placement of Internet2 computed by GPA at $k = 5$.

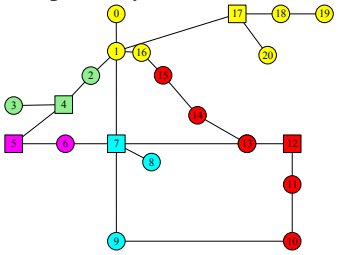


Fig. 9: Placement of Cernet2 computed by OPA at $k = 5$.

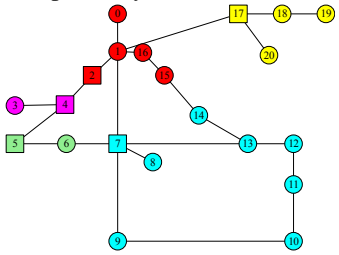


Fig. 10: Placement of Cernet2 computed by GPA at $k = 5$.

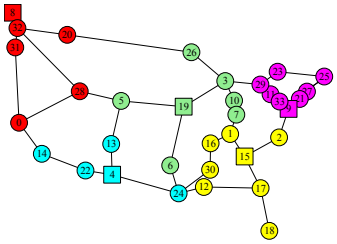


Fig. 11: Placement of Internet2 computed by OPA with 5 controllers with finite capacity.

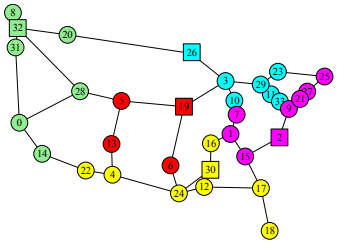


Fig. 12: Placement of Internet2 computed by GPA with 5 controllers with finite capacity.

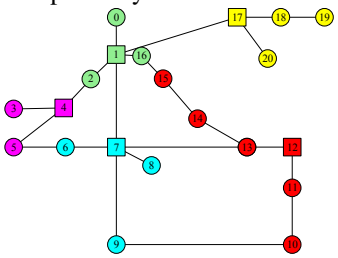


Fig. 13: Placement of Internet2 computed by OPA with 5 controllers with finite capacity.

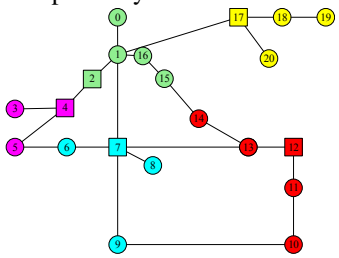


Fig. 14: Placement of Internet2 computed by GPA with 5 controllers with finite capacity.

by OPA at $k = 5$. Meanwhile, in Fig. 7 and Fig. 8, when $k = 5$, the placement scheme computed by OPA also shares node 15 with GPA. It means that node 15 is a critical node for Internet2. Moreover, in Fig. 4 and Fig. 8, node 19, 26 and 30 computed by GPA at $k = 3$ are inclusive at $k = 5$ due to the strategy of GPA. While in Fig. 3 and Fig. 7, node 8 is not included at $k = 5$. As for Cernet2, in Fig. 5 and Fig. 9, OPA at $k = 3$ shares node 4 and node 7 with $k = 5$. Meanwhile, given the same value of k, the placement schemes computed by OPA and GPA are different in the single-link failure situations and the multi-link failure situations. For example, only node 29 is shared at $k = 5$ in OPA for Internet2, and only node 4, 7, 17 are shared at $k = 5$ in OPA for Cernet2. This means the additional multi-link failure events will influence the decision making of the algorithms.

We also evaluate the controller placement schemes of Internet2 computed by GPA when $k = 5$, with higher link failure rates by adjusting the values of α and λ, e.g., $\alpha = 0.7$ and $\lambda = 2000$. The results are shown in Fig. 21. Compared with Fig. 8, node 2 is selected to place the controller instead of node 15. It shows that our greedy approach can output an appropriate placement scheme in different link failure situations.

2) Location of Controllers with finite capacity: We investigate the placement schemes with 5 controllers, each of which has the finite capacity to handle a specific number of nodes in its domain. The results are presented in Fig. 11-Fig. 14. For simplicity, we assume the k controllers have the same maximum capacity, and their aggregate capacity is not less than the total number of nodes in the network, i.e., the capacity of each controller is $\lceil \frac{n}{k} \rceil$. For example, for Internet2 and Cernet2, when $k = 5$, the capacity of each controller is 7 and 5, respectively. In Fig. 11 and Fig. 12, for Internet2, the placement schemes with 5 controllers that have the finite capacity are different from the placement schemes of 5 controllers with infinite capacity shown in Fig. 7 and Fig. 8. This is because, with the controllers that have the finite capacity, OPA and GPA need to consider the capacity constraints of controllers, where the number of nodes in each domain cannot be greater than the controller capacity. Consequently, the computed placement schemes with controllers that have the infinite capacity cannot satisfy the capacity constraints of the controllers with finite capacity, and thus the algorithms have to search for other placement schemes that satisfy the capacity constraints. Correspondingly, the latency performance with controllers that have the finite capacity is worse than that of controllers with infinite capacity. For OPA and GPA, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2994837, IEEE Transactions on Network and Service Management

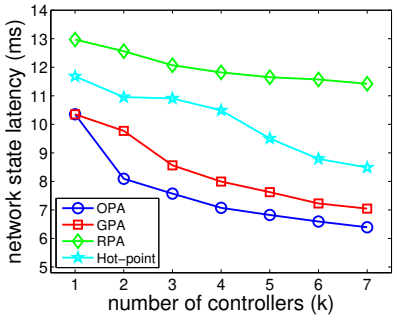


Fig. 15: Worst-case latency of Internet2.

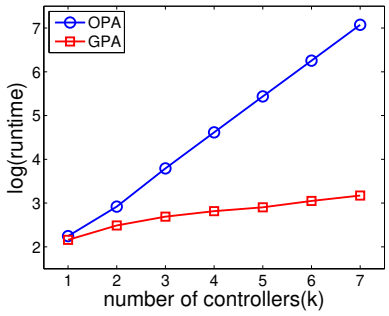


Fig. 16: Runtime of Internet2.

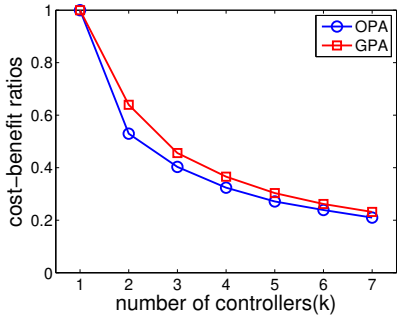


Fig. 17: Cost-benefit ratios of Internet2.

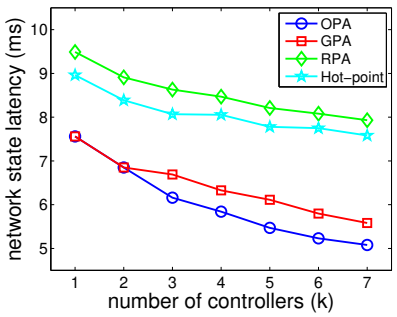


Fig. 18: Worst-case latency of Cernet2.

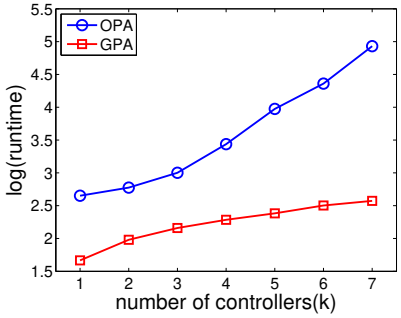


Fig. 19: Runtime of Cernet2.

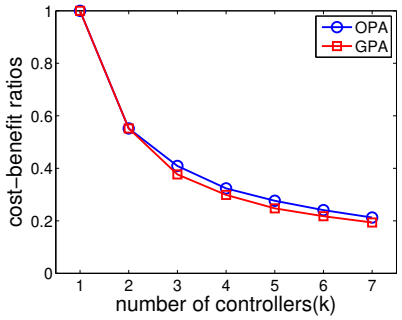


Fig. 20: Cost-benefit ratios of Cernet2.

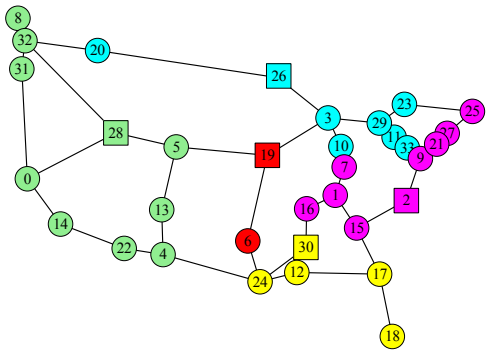


Fig. 21: Placement of Internet2 in different link failure rate.

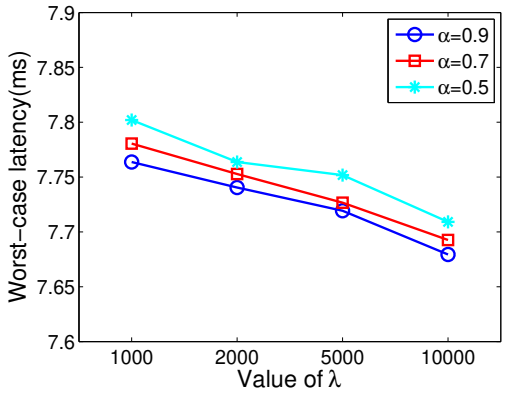


Fig. 22: Worst-case latency in different values of parameters.

latency values with controllers that have the finite capacity are 6.94*ms* and 7.53*ms*, respectively, which are 8.61% and 6.96% larger than that of controllers with infinite capacity. The similar results of Cernet2 can be obtained in Fig. 13 and Fig. 14.

3) Network State Latency: Fig. 15 and Fig. 18 show the results of the worst-case latency for Internet2 and Cernet2. For comparison, we developed the random placement algorithm (RPA) and the Hot-point algorithm for multi-link failures. Regardless of different multi-link failure scenarios, RPA will place the k controllers randomly without optimizing the performance metrics. As for the Hot-point algorithm, it will calculate the closeness centrality of each node, and place the k controllers on the nodes with the highest centrality values that are used to measure the robustness of networks, e.g., the resilience to link failures [41]. RPA is widely used in the controller placement problem [28], [42], and the Hot-point algorithm based on the closeness centrality is also employed for the robust SDN controller placement problem to enable the failover with link failures [43].

In Fig. 15 and Fig. 18, as the number of controllers grows, the worst-case latencies of OPA and GPA decrease, and OPA always performs better than GPA. Compared with CPSLF, the latency gap between OPA and GPA is larger in multi-link failure situations. For Internet2, when $k = 5$, OPA performs only 4.4% better than GPA for single-link failures, but performs 10.7% better than GPA for multi-link failures. Therefore, the multi-link failure scenarios will influence the performance of GPA to some extent. However, GPA still outperforms the RPA and Hot-point algorithm considerably. We can see the latency deviations of GPA, RPA and Hot-point algorithms are getting larger as k grows. For Internet2, when $k = 1$ and $k = 7$, GPA performs 25.1% and 62.2% better than RPA, and also outperforms the Hot-point algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2994837, IEEE Transactions on Network and Service Management

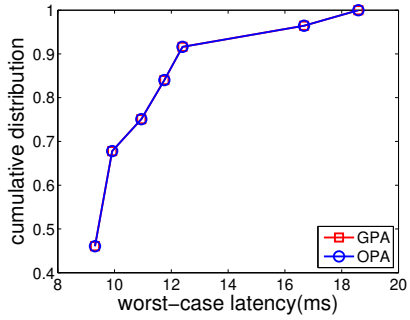


Fig. 23: Worst-case latency distribution of Internet2 at $k = 1$.

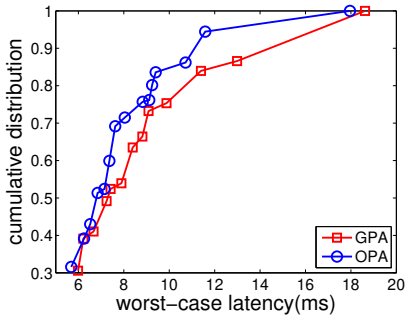


Fig. 24: Worst-case latency distribution of Internet2 at $k = 3$.

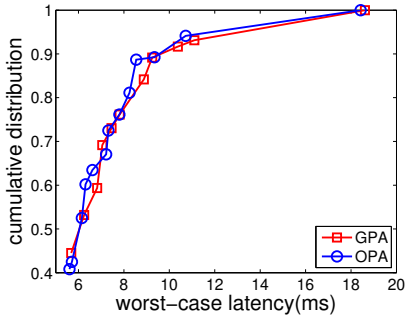


Fig. 25: Worst-case latency distribution of Internet2 at $k = 5$.

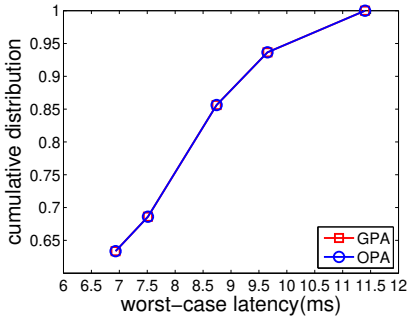


Fig. 26: Worst-case latency distribution of Cernet2 at $k = 1$.

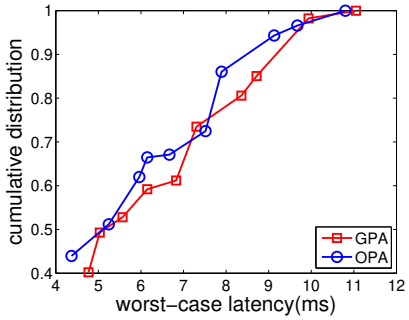


Fig. 27: Worst-case latency distribution of Cernet2 at $k = 3$.

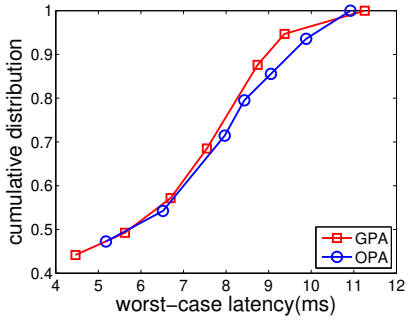


Fig. 28: Worst-case latency distribution of Cernet2 at $k = 5$.

Compared with RPA and Hot-point algorithm, GPA is an efficient method to solve the controller placement problem for multi-link failures. Similar results of Cernet2 can be obtained in Fig. 18.

Furthermore, we evaluate the worst-case latency of Internet2, computed by GPA with different values of link failure rates when $k = 5$. The results are presented in Fig. 22. We increase the link failure rates by decreasing the values of λ and α, and thus the link failure rates are higher than the default values of Internet2. We notice that the worst-case latency is higher than that with the default parameters. For example, when $\lambda = 10000$ and $\alpha = 0.9$, the worst-case latency is $7.679ms$, which is higher than the performance with default parameters, i.e., $7.625ms$. However, our approach still achieves good network performance even in a worse scenario. When $\lambda = 1000$ and $\alpha = 0.5$, our approach achieves $7.802ms$, which is only 2.32% higher than the performance with default parameters. The results show our approaches are efficient and practical with different link failure rates.

4) Worst-case Latency Distribution: Fig. 23-Fig. 25 show the latency distribution results for Internet2. Fig. 26-Fig. 28 show the results for Cernet2. Similar to CPSLF, for CPMLF, as the value of k increases, the distribution gap between OPA and GPA becomes smaller. Moreover, the maximum latency for multi-link failures gets larger, especially the latency computed by GPA. For example, when $k = 3$, the maximum latency computed by GPA for single-link failures is $16.2ms$, while the maximum latency for multi-link failures is $18.62ms$. It shows that the multi-link failure situation will slightly affect the performance of GPA, but it still achieves good performance.

5) Runtime: Fig. 16 and Fig. 19 present the results of runtime logarithm for Internet2 and Cernet2. Obviously, GPA runs much faster than OPA. As k grows larger, the runtime of OPA soars rapidly, since OPA needs exponential time and GPA needs polynomial time. For Internet2, when $k = 1$, OPA needs $556.12s$, which is 3.86 times more than GPA. When $k = 5$, OPA needs $273726.95s$, which is $1,361$ times more than GPA. For Cernet2, the similar results are achieved as well. Compared with CPSLF, due to the excessive multi-link failure events, the computational overhead is about 4 times larger than single-link failures. The results also confirm that we cannot search for an optimal placement scheme in practice, and GPA is an efficient and necessary solution.

6) Cost-Benefit Ratio: We show the cost-benefit ratio performance depicted in Fig. 17 and Fig. 20. We define cost-benefit ratio as $\frac{latency_1}{latency_k} \times \frac{1}{k}$. A value of 1.0 implies a proportional reduction, where k controllers reduce the latency to $1/k$ of the one-controller latency. The higher ratio value, the better performance it will reach. For Internet2, when $k = 1$, reducing the network latency to half of that needs 3 and 2 controllers for OPA and GPA, respectively. For Cernet2, it both needs 3 controllers to reduce to half of that for the two algorithms.

7) Survival Probability: Fig. 29 presents the probability of network survival with different values of controllers for Internet2 and Cernet2. The probability of network survival is the probability of a controller placement scheme surviving multi-link failures, meaning that all nodes are handled and connected by the controllers. A higher probability of network survival shows that our approaches are more resistant to link

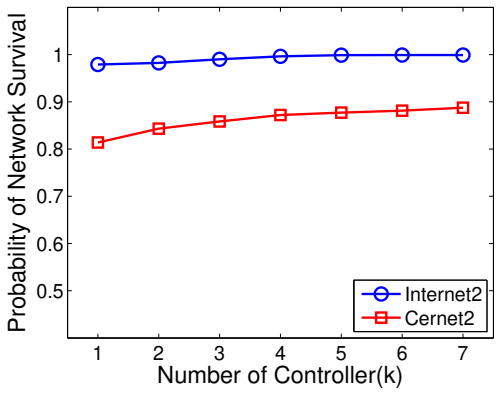


Fig. 29: Survival Probability of Networks.

failures. For Internet2, when $k = 1$, the network can survive 97.89% of link failure situations, and when $k = 7$, it can survive 99.91% of the link failure situations, showing that more controllers will help improve network security. When $k = 1$, Internet2 can still survive the overwhelming majority of multi-link failure situations, which implies that our approaches are helpful, and security of the controller placement schemes computed by our algorithms are guaranteed even in a worse scenario. As for Cernet2, the results show a similar trend with Internet2. We notice that the probability of network survival of Cernet2 is lower than Internet2, because the link failure rates of Cernet2 are higher than those of Internet2. Nevertheless, our approaches for Cernet2 are still protective for most link failure situations.

VII. Conclusion and Future Work

In this paper, we considered the SDN controller placement problem for single-link failures (CPSLF) and multi-link failures (CPMLF). For CPSLF, we developed the Greedy Placement Algorithm (GPA) to reduce the computational overhead. For CPMLF, we pointed out there are many more link failure scenarios for multi-link failures. To reduce the computational overhead, we combined the Monte Carlo Simulation with GPA to simulate the multi-link failure scenarios according to link failure rates, and placed the controllers greedily and iteratively. We evaluated our approaches with real network topologies. Experimental results showed that GPA and the Monte Carlo Simulation are efficient and reliable methods to solve the controller placement problem for single-link and multi-link failures.

In the future, we will consider the failure rates of nodes and controllers as well, and propose comprehensive solutions for the controller placement problem with network failures. We will also consider a larger scale failure analysis, and provide a more precise prediction mechanism for those vulnerable links, nodes, and controllers. Furthermore, we will study the proper number of controllers in different network topologies.

References

[1] M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer, "Flexibility in softwarized networks: Classifications and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2600–2636, Thirdquarter 2019.

[2] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Flexible traffic splitting in openflow networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 407–420, Sep. 2016.

[3] L. Li, W. Chou, W. Zhou, and M. Luo, "Design patterns and extensibility of rest api for networking applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 154–167, March 2016.

[4] S. Hong, R. Baykov, L. Xu, S. Nadimpalli, and G. Gu, "Towards sdn-defined programmable byod (bring your own device) security," in *Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS'16)*, Feb 2016.

[5] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, Dec 2018.

[6] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful sdn data planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701–1725, Thirdquarter 2017.

[7] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, July 2018.

[8] Q. Li, Y. Chen, P. P. C. Lee, M. Xu, and K. Ren, "Security policy violations in sdn data plane," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1715–1727, Aug 2018.

[9] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, Fourthquarter 2015.

[10] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, ser. INM/WREN'10. USA: USENIX Association, 2010, p. 3.

[11] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. USA: USENIX Association, 2010, pp. 351–364.

[12] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 1–6.

[13] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 473–478, Sep. 2012.

[14] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 31–36.

[15] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, March 2015.

[16] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards controller placement for robust software-defined networks," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Dec 2015, pp. 1–8.

[17] T. Elhourani, A. Gopalan, S. Ramasubramanian, T. Elhourani, A. Gopalan, and S. Ramasubramanian, "Ip fast rerouting for multi-link failures," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 3014–3025, Oct. 2016.

[18] "Internet2 open science, scholarship and services exchange," https://www.internet2.edu/products-services/advanced-networking/layer-2-services/.

[19] "China education and research network 2 (cernet2)," http://www.cernet2.edu.cn/.

[20] M. Caria, A. Jukan, and M. Hoffmann, "Sdn partitioning: A centralized control plane for distributed routing protocols," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 381–393, Sep. 2016.

[21] W. Li, W. Meng, and L. F. Kwok, "A survey on openflow-based software defined networks: Security challenges and countermeasures," *Journal of Network and Computer Applications*, vol. 68, pp. 126–139, 2016.

[22] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker,

"P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[23] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high-performance network operating system," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 78–89.

[24] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee, "Dynamic packet forwarding verification in sdn," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 915–929, Nov 2019.

[25] S. S. W. Lee, K. Li, K. Y. Chan, G. Lai, and Y. C. Chung, "Software-based fast failure recovery for resilient openflow networks," in *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, Oct 2015, pp. 194–200.

[26] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient sdn control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Sep. 2016, pp. 253–259.

[27] A. Alshamrani, S. Guha, S. Pisharody, A. Chowdhary, and D. Huang, "Fault tolerant controller placement in distributed sdn environments," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[28] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 672–675.

[29] M. J. Abdel-Rahman, E. A. Mazied, K. Teague, A. B. MacKenzie, and S. F. Midkiff, "Robust controller placement and assignment in software-defined cellular networks," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, pp. 1–9.

[30] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, and M. El-Nainay, "On stochastic controller placement in software-defined wireless networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017, pp. 1–6.

[31] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella, and L. Tassiulas, "Sdn controller placement with delay-overhead balancing in wireless edge networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1446–1459, Dec 2018.

[32] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in sdn," in *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2016, pp. 481–487.

[33] B. P. R. Killi and S. V. Rao, "Link failure aware capacitated controller placement in software defined networks," in *2018 International Conference on Information Networking (ICOIN)*, Jan 2018, pp. 292–297.

[34] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined wans," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, Sep. 2018.

[35] S. Petale and J. Thangaraj, "Failure-based controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 503–516, March 2020.

[36] M. Ashrafi, N. Correia, and A.-T. Faroq, "A scalable and reliable model for the placement of controllers in sdn networks," in *International Conference on Broadband Communications, Networks and Systems*. Springer, 2018, pp. 72–82.

[37] L. Li, N. Du, H. Liu, R. Zhang, and C. Yan, "Towards robust controller placement in software-defined networks against links failure," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 216–223.

[38] G. Aceto, A. Botta, P. Marchetta, V. Persico, and A. Pescapé, "A comprehensive survey on internet outages," *Journal of Network and Computer Applications*, vol. 113, pp. 36–63, 2018.

[39] G. Rubino and B. Tuffin, *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

[40] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct 2011.

[41] D. F. Rueda, E. Calle, and J. L. Marzo, "Robustness comparison of 15 real telecommunication networks: Structural and centrality measurements," *Journal of Network and Systems Management*, vol. 25, no. 2, pp. 269–289, 2017.

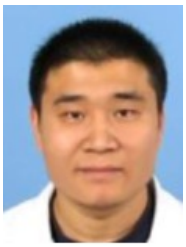
[42] Z. Su and M. Hamdi, "Mdcp: Measurement-aware distributed controller placement for software defined networks," in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2015, pp. 380–387.

[43] D. Santos, A. de Sousa, and C. M. Machuca, "Combined control and data plane robustness of sdn networks against malicious node attacks," in *2018 14th International Conference on Network and Service Management (CNSM)*, Nov 2018, pp. 54–62.

Shu Yang received the B.Sc. degree from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree from Tsinghua University, Beijing. He is currently an associate researcher with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interest includes network architecture, edge computing and high performance router.

Laizhong Cui is currently a Professor in the College of Computer Science and Software Engineering at Shenzhen University, China. He received the B.S. degree from Jilin University, Changchun, China, in 2007 and Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. His research interests include Future Internet Architecture and protocols, Edge Computing, Multimedia Systems and Applications, Blockchain, Internet of Things, Cloud and Big Data Computing, Software-Defined Network, Social Network, Computational Intelligence and Machine Learning. He led more than 10 scientific research projects, including National Key Research and Development Plan of China, National Natural Science Foundation of China, Guangdong Natural Science Foundation of China and Shenzhen Basic Research Plan. He has published more than 70 papers, including IEEE Transactions on Multimedia, IEEE IoT Journal, IEEE Transactions on Industrial Informatics, IEEE Transactions on Vehicular Technology, IEEE Transactions on Network and Service Management, ACM Transactions on Internet Technology, IEEE Transactions on Computational Biology and Bioinformatics and IEEE Network. He serves as an Associate Editor or a Member of Editorial Board for several international journals, including International Journal of Machine Learning and Cybernetics, International Journal of Bio-Inspired Computation, Ad Hoc and Sensor Wireless Networks and Journal of Central South University. He is a Senior Member of the IEEE, and a Senior Member of the CCF.

Ziteng Chen received his B.Sc. degree from Shenzhen University, Shenzhen, China, in 2018. He is pursing his M.Sc. degree in Shenzhen University. His research interest includes software-defined networking and blockchain.



Wei Xiao received his Doctor degree in computer science and technology in Tsinghua University, China. His research interests include computer vision and multi-modal information fusion. He has published several articles in IEEE International Conference on Multi-sensor Fusion and Integration for Intelligent Systems and ACM multimedia Conference (MM).