Controller Backup and Replication for Reliable Multi-domain SDN

Junli Mao[1], Lishui Chen[2], Jiacong Li[3], and Yi Ge[4]*
[1,2] The 54th Research Institute of CETC
Shijiazhuang Hebei, 050081, China
[e-mail: maoj1_@126.com, 15031150937@139.com]
[3] China Telecom Corporation Limited Research Institute
Beijing, 102290, China
[e-mail: lijc50@chinaletecom.cn]
[4] State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, 100876, China
[e-mail: geyi@bupt.edu.cn]
*Corresponding author: Yi Ge

*Received June 1, 2020; revised July 30, 2020; accepted November 17, 2020;
published December 31, 2020*

Abstract

Software defined networking (SDN) is considered to be one of the most promising paradigms in the future. To solve the scalability and performance problem that a single and centralized controller suffers from, the distributed multi-controller architecture is adopted, thus forms multi-domain SDN. In a multi-domain SDN network, it is of great importance to ensure a reliable control plane. In this paper, we focus on the reliability problem of multi-domain SDN against controller failure from perspectives of backup controller deployment and controller replication. We firstly propose a placement algorithm for backup controllers, which considers both the reliability and the cost factors. Then a controller replication mechanism based on shared data storage is proposed to solve the inconsistency between the active and standby controllers. We also propose a shared data storage layout method that considers both reliability and performance. Besides, a fault recovery and repair process is designed based on the controller backup and shared data storage mechanism. Simulations show that our approach can recover and repair controller failure. Evaluation results also show that the proposed backup controller placement approach is more effective than other methods.

Keywords: SDN, Backup Controller Placement, Shared Data Storage, Failure Recovery

http://doi.org/10.3837/tiis.2020.12.006 **ISSN : 1976-7277**

1. Introduction

As an evolving technology, software defined networking (SDN) separates the control from data plane. It is proposed to reduce the complicacy of network management and improve the network performance [1-2]. However, the single and centralized controller brings potential problems with scalability and control plane performance. Therefore,some research works have designed multiple controller architectures to avoid this bottleneck. Among them the distributed multi-controller architecture [3-6] has been widely adopted, which forms multi-domain SDN.

In SDN environment, controller plays a significant role. However, controllers are liable to fail. Therefore, it is important to study the reliability problem of multi-domain SDN against controller failure. An active backup approach is usually a natural choice to solve such problem. That is to deploy some backup controllers so that the network can work normally in case of controller failure. Such approach mainly involves two aspects and the following challenges are met:

- First of all, in the aspects of the backup controller deployment, lots of existing work focus on primary controller deployment, which cannot be applied on backup controller deployment directly. Because the number and location of backup controllers should consider the failure probability of master controller and other factors.

- Secondly, in the aspects of the controller replication, most methods are based on the communication between controllers, which might lead to inconsistency between controllers.

In order to obtain the smooth failure recovery and repair, network information needs to be exchanged between the primary and backup controllers.

In the past, a lot of work has been done on backup controller deployment and controller replication. As for the controller deployment problem [7-12], most researches focused on the primary controller deployment, and there have been few studies on the deployment of backup controllers in multi-domain SDN. For controller replication [13-16], most existing methods are based on the communication between controllers, which might lead to inconsistency between controllers. Some approaches [17-18] used data store to avoid interaction between controllers for information exchange between working controllers, but these methods did not mention the detailed design of data store and the concrete steps of failure recovery based on the data store.

Therefore, our work mainly solves two problems: the backup controller deployment problem and the controller replication problem. To settle the first one, we attempt to enhance the reliability of the network through finding the appropriate placement scheme for backup controllers, so as to deploy necessary backup controllers against controller failure. In order to solve the second problem, we try to design a mechanism that can store data to ensure the uniformity of primary and backup controllers without communication between controllers.

The main contributions of this study are as follows:

- In order to improve the flexibility of large networks, a method for multi-domain SDN network sharing backup controller is proposed. In order to optimize the reliability of the control plane with as fewer alternate controllers as possible, we figure out the number of spare controllers according to the probability of failure and calculate the layout of the spare controllers through a multi-objective optimization algorithm named PSO.

- A mechanism that can share data store is proposed to insure the uniformity between primary and backup controllers. With such mechanism, the controller can update the current network information to the data store once the network changes. To improve the performance of network, controllers can get the network information from the data store instead of communicating with other controllers.

- A data store placement approach is proposed to ensure the dependability of the data store and minimize the communication delay. We introduce a hot-standby data store to prevent the primary data store failure and design a greedy algorithm to calculate the location of data stores, minimizing the communication delay to the controllers.

The remainder of this paper is organized as follows. Section 2 reviews the relevant work. Section 3 presents the motivation examples. Section 4 formulates both the backup controller placement problem and the controller replication problem. In Section 5 presents our design and implementation of the proposed mechanisms. In Section 6, we present simulation results. Finally, this paper is concluded in Section 7.

2. Related Work

Many researchers have paid much attention to the controller deployment problem [7-12] and the controller replication problem [13 -18]. The comparison and classification of relevant work are shown in **Table 1**.

Table 1. Comparison of related work features

Category	Highlights	Research
The controller deployment problem	The propagation delay based approaches	[7, 8]
	The network reliability based approaches	[9-12]
The controller replication problem	The active replication approaches	[14,15]
	The passive replication approaches.	[16]
	Sharing database	[17,18]

2.1 Controller Deployment

The controller deployment problem [7-12] could be divided into two categories: one is based on propagation delay methods, and the other is based on network reliability methods. The method based on network reliability proposed a controller placement scheme to enture the network reliability using minimum controllers.

The above researches all focused on the primary controller placement. Different from the above works, other than only focusing on the primary controller placement, we propose a configuration method of shared backup controller SDN network with multiple domains, which improves the reliability of the control network under the premise of meeting the delay requirements.

2.2 Controller Replication

There are two kinds of controller replication methods: active method and passive method [13]. In an active replication scenario, a switch is connected to multiple controllers that process requests. The backup controller can take control of the network at any time as long as it keeps consistent with the primary controller. However, it has some disadvantages. First of all, it is difficult to ensure that all messages are delivered to all controllers in an orderly manner. Secondly, it is not desired that all controllers retain the total network view. Lastly, both primary and backup controllers need to process the request from the switches. It may result in additional overhead.

In order to solve these problems, an active replication strategy is performed in the Ryu controller by Eros et al. [14] to guarantee uniformity among the distributed controllers though the OpenReplica service. Naga Katta et al. [15] introduced Ravana, a fault-tolerant SDN controller platform that processed the control messages. Nevertheless, these approaches were based on the external service or new protocol to optimize the process of active replication. Therefore, it is complicated to realize these approaches and may lead to the computation performance degradation of the controller.

In a passive replication scenario, a switch connects to only a controller that handles the requests and the controller updates other controllers. The author [16] proposed a new mechanism that used CPRecovery component organization to improve SDN resilience. Compared with the active mode, the processing costs of slave controllers are saved. However, it is required that the total slave controllers provide surveillance to the master controller to ensure consistency in the event of a failure.

Besides the above two replication approaches, Fábio Botelho et al. [17-18] implemented a prototype with a sharing database to facilitate the replication. However, they did not explain how to design and deploy the sharing database and how to recover from failure based on the database.

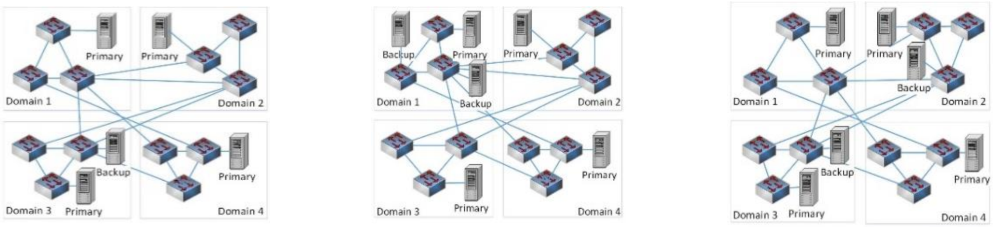
Different from these related works, a shared data storage mechanism is proposed to exchange network information between prime and backup controllers. It can avoid the inconsistence of controllers largely. We not only design the data structure of the data store, propose a data store deployment approach to ensure the reliability and performance of the information exchange, but also design the failure recovery and repair process. In addition to the above researches on control plane reliability, there are also some research [19] on the reliability of data plane based on SDN centralized control. The reliability of data plane is not the scope of the paper.

3. Motivating Examples

3.1 Backup Controller Deployment Problem

Fig. 1 presents an instance of an SDN network including four domains, where each domain has a primary controller and three switches connecting with each other. (a), (b), and (c) are three specific backup controller deployment cases. As shown in the figure, there is one backup controller in case (a), and there are two backup controllers in case (b) and case (c).

Obviously, if the probability that multiple primary controllers fail at the same time is small, one backup controller is enough in this scenario, as shown in (a).



(a) (b) (c)

Fig. 1. Examples demonstrating backup controller placement problem

However, if the failure probability of the primary controller is high, more backup controllers will be needed. So it is necessary to decide the number of backup controllers. In (b) and (c), there are both two controllers, but the locations of the controllers are different. It can be seen that (c) is more reliable than (b), because if a regional failure occurs in domain 1, all the backup controllers will crash at the same time. Therefore, we need to properly design the number and locations of backup controllers. Besides the above physical isolation factor, there are still more factors need to be considered, such as the reliability of the control network based on the deployment of backup controllers, and the delay between the switches and the backup controllers.

3.2 Controller Replication Problem

We discuss controller replication problem from two aspects: the active replication and the passive replication. In active mode, both the primary and backup controllers process the request from switches at the same time, which may cause these two controllers inconsistent. As shown in **Fig. 2**, there is one primary controller, one backup controller and two disjoint paths, <S1, S2> and <S1, S3, S2> in the network. The bandwidth of each path is 10Mbps. Suppose there are two data flows need to be sent from switch S1 to switch S2. One of them needs 10Mbps bandwidth and the other needs 5Mbps. Assuming these two flow requests arrive at these two controllers in different order owing to network delays, the primary assigns the 10Mbps flow to the path <S1, S2>, and the 5Mbps flow to the path <S1, S3, S2>. However, the backup controller assigns the 10Mbps flow to the path <S1, S3, S2> and the 5Mbps flow to the path <S1, S2>. Now, considering the primary controller crashes and the backup becomes primary controller, if a new flow with 5Mbps arrives, the new master assigns the flow to path <S1, S2>, which it believes has 5Mbps available. Nevertheless, this blocks the path that is already fully utilized, as the information of the network in new primary controller diverges from its actual state.

In passive mode, the request from switches is processed by only the primary controller, then the primary controller informs the result to the backup controller. However, there still exists the possibility of inconsistency. As **Fig. 2** shows, the primary controller receives requests from switches and processes them, then it updates information to the backup controller. Now, assume that path <S1, S2> fails, switch S1 sends this information to the primary controller. Then the primary crashes before it sends the update to the backup. When the backup becomes master, it thinks the path <S1, S2> available, which diverges from its actual state.

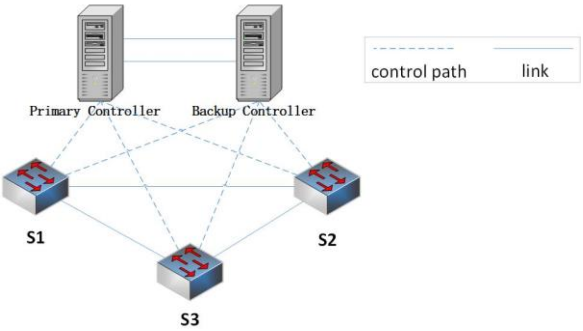


Fig. 2. Examples demonstrating controller replication problem

In order to solve the problem of the consistency between the controllers as mentioned above, we employ a shared data storage to store the whole network status in real time. With such mechanism, the primary and backup controller do not need to communicate with each other to exchange the information. This approach can avoid the inconsistency problem between primary and backup controllers in active and passive replication. Obviously, the data store plays an important role in such mechanism, so the location of the data store will also influence the performance and the reliability of the network.

In order to ensure the reliability of the data store, we propose to deploy two data stores in the network, one is the primary, and the other is the backup. The backup one is a hot standby to ensure the network can operate normally when the primary one crashes. To improve the performance control network at the same time, we also optimize the location of the backup data store.

4. Problem Formulation

Based on the above motivation, in this section, we formulate our problem as a sharing backup controller deployment problem and a controller replication problem. For the input of the problem, we can use the following tuple $I=\{G(N,L);C;D;fault_C;delay_G\}$. The physical network of SDN is formulated as an undirected weighted graph $G=(N,L)$. N represents the set of switches, L indicates the set of the links connecting switches. C denotes the set of primary controllers of each domain, each element of the set is presented by a switch with which the primary controller is connected to the network. D is the set of domains. Additionally, the probability of controller failure is represented by $fault_C$. $delay_G$ indicates the matrix of the latency of each two adjacent nodes.

This paper is based on the following assumptions:

- The controllers and switches are connected with in-band mode. And the data store and controllers are also connected in in-band mode.
- Each domain has only one primary controller.
- The latency between two switches in the same domain is smaller than the latency between two switches from different domains.

4.1 Sharing Backup Controller Deployment Problem

In general, more backup controllers in a multi-domain SDN may make the network more reliable. However, the reliability is not increased by simply adding redundant backup controllers, which will lead to the increase in cost at the same time. Therefore, we need to place as less backup controllers as possible to guarantee the reliability of the entire network. Besides, in multi-domain SDN environment, the transmission delay between switches and their controllers must meet a response time requirement, otherwise the network cannot work.

Without loss of generality, given a multi-domain SDN network and the set of master controllers, the sharing backup controller deployment problem is to find appropriate number and locations of backup controllers, while satisfying the follows: (i) to ensure the whole network reliability with minimum backup controllers; (ii) to satisfy the required delay.

The output of the problem is represented by the tuple $O=\{N^{BC};num\}$, which stands for the backup controller deployment scheme. N^{BC} is a subset of N, each element of which represents a switch through which a backup controller is connected to the network. num stands for the number of backup controllers.

4.2 Controller Replication Problem

As mentioned above, we plan to design a shared data storage to solve the controller consistency problem. We need to design the data structure of the network information stored in data store. Since the data store plays an important role in the network, it is necessary to guarantee the reliability of the network when the data store crashes. Therefore, we deploy a hot-standby data store. Considering the performance of the network, we need to place these two data stores at appropriate locations to ensure that the latency from all the controllers to both primary and backup data store is minimum. Without loss of generality, given the input tuple as mentioned above, the data store deployment problem is to find proper locations of two data stores. That is to say, we need to find a subset of N, denoted as N^{DS}, on which the two data stores are placed, and the latency from all the controllers to data stores are minimum.

A summary of the input and output symbols of the above problem formulation is shown in **Table 2**.

Table 2. The summary of the input and output symbols

Symbol	Definition
N	Set of network nodes (forwarding devices).
L	Set of links.
C	Set of primary controllers.
D	Set of domains.
$fault_C$	Probability of controller failure.
$delay_G$	Matrix of latency between each two adjacent nodes.
N^{DS}	Set of nodes where a data store is placed.
N^{BC}	Set of nodes where a backup controller is placed.
S_{ij}	The switch *j* in the domain *i*.
num	The number of backup controllers.

5. Design and Implementation

5.1 Sharing Backup Controller Deployment Approach

In this part, we first define a metric to measure the reliability of the control plane. Then, we refine the sharing backup controller deployment problem by analyzing the objectives and constraints. In addition, we put forward an algorithm to find the deployment scheme of backup controllers based on particle swarm optimization (PSO) algorithm.

5.1.1 The reliability metric

In this paper, we evaluate the reliability of the control plane based on the reliabilities of all the controller-switch pairs. So we need to measure the reliability of two nodes (from controller to switch) at first. We define a metric called *reliability factor* to indicate the reliability of a controller-switch in a network. The summary of the symbols used in the definition of the reliability factor is shown in **Table 3**.

Table 3. The summary of the reliability factor symbols

Symbol	Definition
n_{uv}	The number of paths between node *u* and *v*
n_{uvl}	The link amount of all paths between node *u* and *v*
$\overline{len_{uv}}$	The average length of paths between *u* and *v*
len_{uvi}	The length of path i between *u* and *v*
dif_{uv}	The difference in length of all paths between node *u* and *v*
cor_{uv}	The average correlation of all paths between node *u* and *v*
rf_{uv}	The reliability factor between node *u* and *v*
P_{uvi}	The path i between *u* and *v*
$X_{l,P_{uvi}}$	Whether link l belongs to path i between *u* and *v*

Zhang Yet et al. [11] have proposed that the reliability of two nodes can be represented by the average length of paths, the number of paths and the maximum rate of overlapping link. However, through analysis, we observed that the above factors are not enough. Therefore, the average correlation of all of the paths between node pairs is defined as the following:

$$cor_{uv} = \frac{\sum_{l \in L} \sum_{i=1}^{n_{uv}} X_{l,P_{uvi}}}{n_{uvl}} / n_{uv} \tag{1}$$

where $X_{l,P_{uvi}}$ denotes whether the link l is on the path i between node u and v, and n_{uv} devotes the number of links of all paths between node u and v. For example, in **Fig. 3**, the cor_{uv} of (a) is 4/7, and the cor_{uv} of (b) is 4/6. We can find that the reliability increases as the average correlation decreases.

Although the average length, the maximum rate of overlapping link, the number of paths, and average correlation are equal, the reliabilities of the node pairs may still be different. As shown in (a) and (c) of **Fig. 3**, the number of paths is 2, the average length is 4, the average correlation coefficient is 4/7, and the maximum overlap rate is 1. However, (c) is more reliable than (a). Since the difference between the length of the two paths of (c) is greater than that of (a). Based on it, the length difference of all paths is defined as:

$$dif_{uv} = \frac{\frac{\sum_{i=1}^{n_{uv}} (len_{uvi} - \overline{len_{uv}})^2}{\overline{len_{uv}}}}{n_{uv}} \tag{2}$$

where $\overline{len_{uv}}$ presents the average length of paths between u and v as (3) shows.

$$\overline{len_{uv}} = \frac{\sum_{i=1}^{n_{uv}} len_{uvi}}{n_{uv}} \tag{3}$$

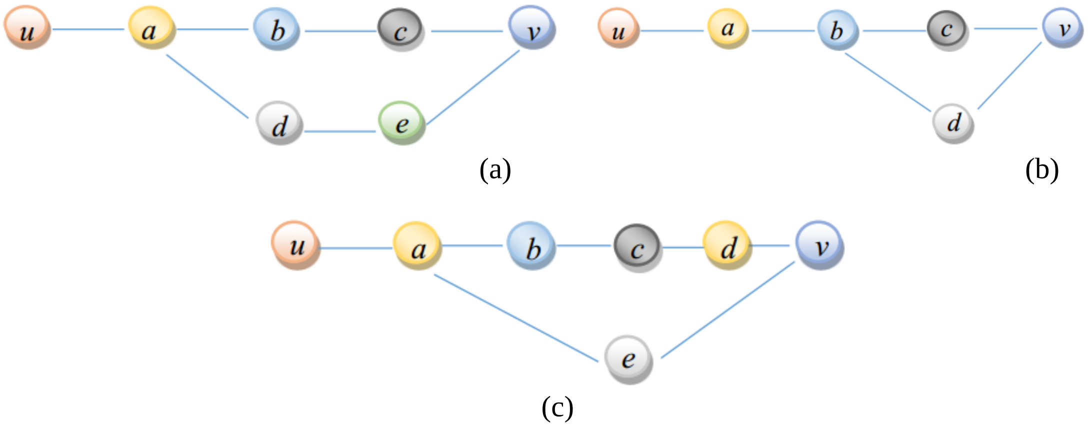


Fig. 3. Different paths case between node u and v

Based on the above metrics, *reliability factor* indicates the reliability of the two nodes in a network, as defined in equation (4).

$$rf_{uv} = \frac{n_{uv}}{\overline{len_{uv}} * cor_{uv} * (1 - dif_{uv})} \tag{4}$$

(4) is used to evaluate the reliability between a controller-switch pair. Based on this, we define the reliability of the control plane. That is the sum of the reliability factor of each controller-switch pair, as defined by (5), S_{xy} represents the location where a backup controller is placed.

$$R = \sum_{v \in N} rf_{S_{xy},v} \tag{5}$$

5.1.2 Formulation Refinement

Shared backup controller is shared in multiple domains. Therefore, the number of backup controllers is less than the number of domains so that unnecessary cost could be saved. To solve the sharing backup controller placement problem, We need to consider three issues: how many backup controllers do we need to deploy; how to allocate the backup controller to domains, and where to place them.

There are three objectives need to be achieved in the sharing backup placement problem. The first is to minimize the cost, that is, to minimize the number of backup controllers. The objective is formulated as (6), where $x_{ij} \in \{0,1\}$ indicates whether backup controller is connected to switch S_{ij}.

$$F_1 = \min \sum_{1<i<D, j \in N} x_{ij} \tag{6}$$

The second goal is to minimize the total delay, as shown in equation (7), where S_{xy} devotes the location where a backup controller is placed and S_{ij} means the location of a switch *j* in domain *i*.

$$F_2 = \min \sum_{1<i<D\&i\neq x,\, j\in N} delay(S_{ij}, S_{xy}) \tag{7}$$

The last objective is to maximize the reliability of the control plane, which can be represented as (8), where S_{xy} stands for the location where a backup controller is placed.

$$F_3 = \max \sum_{v\in N} rf_{S_{xy},v} \tag{8}$$

The first three constraints (9) - (11) guarantee the backup controller be placed appropriately. (9) ensures that a backup controller is not located in the same position as a primary controller node.

$$S_{ij} \notin C, \forall x_{ij} = 1 \tag{9}$$

(10) is to guarantee the physical isolation, where $y_{ij,d} \in \{0,1\}$ indicates whether the backup controller connected to the switch S_{ij} is responsible for domain $d \in D$. The physical isolation means if the backup controller *C* is responsible for domain *A*, it cannot be deployed in the domain. This constraint ensures that the primary and the backup controller for one domain are not deployed in the same domain.

$$i \neq d, \forall y_{ij,d} = 1, x_{ij} = 1 \tag{10}$$

(11) ensures that all backup controllers can cover all the domains. |D| represents the number of domains.

$$\sum_{d\in D} y_{ij,d} \geq |D| \tag{11}$$

Constraint (15) ensures that the number of backup controllers is in a certain range. Thus the amount of backup controllers is sufficient to guarantee the reliability of the control plane. num means the number of backup controllers. If a possible fault occurs, it is expressed as $F_i = 1$, otherwise $F_i = 0$. Obviously, the random variable F_i is with the Bernoulli distribution. (12) represents the amount of failures. (13) represents the probability of more than k' controllers fail concurrently. (14) means the probability of controllers concurrently fail is less than a certain value.

$$|F| = \sum F_i \tag{12}$$

$$p(|F| = k') = (\begin{matrix}|C| \\ k\end{matrix}) f_C^{k'} (1 - f_C)^{|C|-k'} \tag{13}$$

$$p(|F|>k')<\varepsilon \tag{14}$$

$$k' \leq num \leq |C| \tag{15}$$

5.1.3 Algorithm

An algorithm based on particle swarm optimization (PSO) [20] is designed. The algorithm abstracts the backup controllers into particles, and the solution to the problem is represented by the position of particle. Since there are three goals in this problem, we can take them as one fitness function as (16) shows. It can be easily seen that the lower the value of this function is, the better the result will be. The coefficient α , β and γ represent the different importance of three objectives and it can be modified according to different situations. Additionally, the coefficients must satisfy equation (17).

$$F = \alpha F_1 + \beta F_2 + \gamma \frac{1}{F_3} \tag{16}$$

$$\alpha + \beta + \gamma = 1 \tag{17}$$

In our model, there are |C| domains in the network and $|S_i|$ switches in domain *i*. In order to avoid all backup controller failure at the same time caused by regional failure, we put at most one backup controller in each domain. Therefore, we use |C|-bit integer to denote the solution to this problem, and each bit range from 0 to $|S_i|$-1 . 0 means there is no backup controller in the domain, the other number represents the possible position of backup controller except the location of primary controller.

In PSO algorithm, we need to update the speed of the particle, then update the position based on the new speed. In order to consider more situations in limited number of iterations, we update the speed of particles randomly. In *UpdateSpeed()* function of our algorithm, we generate |C|-bit integer as speed, and each bit ranges between $-(|S_i|-1)$ and $+(|S_i|-1)$ randomly. Then we get the new position by adding the last position and the new speed. If the result is out of the range 0 to $|S_i|-1$, then make the bit equal to 0 or $|S_i|-1$. Besides, if the amount of backup controllers showed by the new position is less than the result calculated by function (14), then we need to generate the speed and compute the new position again. **Table 4** presents the pseudo code of the algorithm to calculate the location of the backup controller.

Table 4. Algorithm to compute the sharing backup controller placement scheme

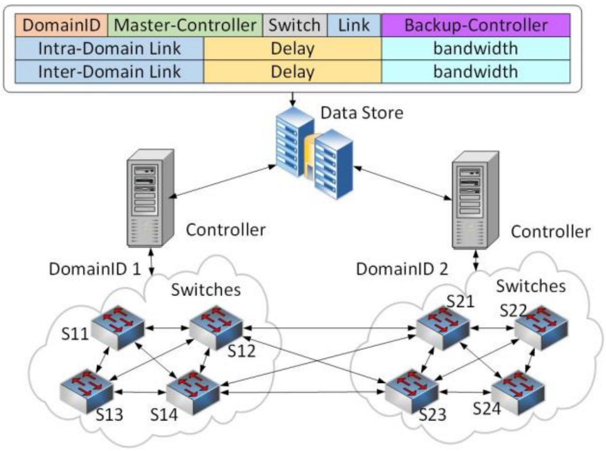
```
Algorithm 1 Algorithm to compute the sharing backup
controller placement scheme
Input: $G=(N, L)$, $|C|$
Output: $gBest$
Procedure:
loop=0;
(X,V)=Random(PopSize);
Fitness = F(X);
for i in X
    pBest[i] = $X_i$
end for
gBest = min{ pBest[i]};
while (loop < MaxLoop) do
    for 1 to PopSize
        V = UpdateSpeed(X,V);
        X = UpdatePosition(X,V);
        if F( $X_i$ ) < F(pBest[i])
            pBest[i] = $X_i$ ;
        if F(gBest) > F(pBest[i])
            gBest = pBest[i];
    end for
    loop++;
end while
```

5.2 Sharing Data Store Based Replication Approach

5.2.1 Architecture Design

Based on the assumptions of this paper, there are several domains in a network and there is one *primary* controller in each domain. A controller replication method based on data storage is proposed. The data store based multi-domain SDN architecture is illustrated in **Fig. 4**.

The data structure of the data store is designed as shown in **Fig. 4**. The data store mainly records three kinds of information: (*i*) The topology information of each domain, including the information of switches, master controller, links between every two switches, and the backup controllers of each domain. (*ii*) The information of intra-domain links, involving the information of the delay and bandwidth of the links between each two switches in each domain. (*iii*) The information of inter-domain links, recording the delay and bandwidth of inter-domain links.



DomainID	Master-Controller	Switch	Link	Backup-Controller
Intra-Domain Link		Delay		bandwidth
Inter-Domain Link		Delay		bandwidth

Fig. 4. Data store based multi-domain SDN

Table 5-7 present the data instances of data store information based on the example illustrated in **Fig. 4**, where there are two domains in the network. **Table 5** gives the example data of domain topology information. In this paper, we assume that the controllers and switches are connected in in-band mode. We take the switches that the controller is connected to indicate the location of the controller. **Table 6** gives the example data of the intra-domain link information. **Table 7** gives the inter-domain link information.

Table 5. The topology information of each domain

Domain ID	Master Controller	Switch	Link	Backup Controller
1	S_{12}	{S_{11}, S_{12}, S_{13}, S_{14}}	< S_{11}, S_{12}>< S_{11}, S_{13}> < S_{11}, S_{14}>< S_{12}, S_{13}> < S_{12}, S_{14}>< S_{13}, S_{14}>	{S_{22}}
2	S_{21}	{S_{21}, S_{22}, S_{23}, S_{24}}	< S_{21}, S_{22}>< S_{21}, S_{23}> < S_{21}, S_{24}>< S_{22}, S_{23}> < S_{22}, S_{24}>< S_{23}, S_{24}>	{S_{14}}

Table 6. The inter-domain link information

Intra-Domain Link	Delay (ms)	Bandwidth
< S_{11}, S_{12}>	0.08359092266	908.029326
< S_{11}, S_{13}>	0.22824170439	284.387241
< S_{11}, S_{14}>	0.10084454131	226.981221
< S_{12}, S_{13}>	0.06578912309	493.621828
< S_{12}, S_{14}>	0.09177560092	763.039444
< S_{13}, S_{14}>	0.12534200728	884.167920
< S_{21}, S_{22}>	0.09393250206	1004.29369
< S_{21}, S_{23}>	0.14077037152	661.833826
< S_{21}, S_{24}>	0.08504249152	117.815603
< S_{22}, S_{23}>	0.07383756005	831.273390
< S_{22}, S_{24}>	0.08012484523	497.556873
< S_{23}, S_{24}>	0.12679823859	585.667089

Table 7. The information of inter-domain link

Inter-Domain Link	Delay (ms)	Bandwidth
< S_{12}, S_{21}>	0.58359345266	1908.02326
< S_{12}, S_{23}>	0.42824346439	1284.38241
< S_{14}, S_{21}>	0.50084678631	1526.98121
< S_{14}, S_{23}>	0.66578342689	1493.62828

5.3 Sharing Data Store Placement Approach

In order to optimize the communication performance between primary controllers and the data stores, we propose a sharing data store placement method based on latency. Taking the matrix $delay_G$ of delay between each two nodes as input, we need to find out two nodes, the delays from which to the primary controllers are the minimum and the second minimum.

Considering the *Floyd* algorithm is a multi-source shortest path algorithm to calculate the shortest path between each point pair, given a network $G=$ （N, L） and $delay_G$, we use

Floyd algorithm (**Algorithm 2** in **Table 8**) to work out the minimum latency matrix $LowestDelay_G$.

Table 8. Algorithm to construct the minimum latency matrix

```
Algorithm 2 Algorithm to construct the minimum latency matrix
Input: $G = (N, L)$, $delay_G$
Output: $LowestDelay_G$
Begin:
$LowestDelay_G \leftarrow delay_G$
for $k$ in $N$
  for $i$ in $N$
    for $j$ in $N$
      if ( $LowestDelay[i, j] > LowestDelay[i, k] + LowestDelay[k, j]$ )
        $LowestDelay[i, j] \leftarrow LowestDelay[i, k] + LowestDelay[k, j]$
      end if
    end for
  end for
end for
END
```

Based on the minimum latency matrix, we can compute the sum of the latencies from one node to the others. Then we can find the two nodes which have the minimum and the second minimum latency to deploy the sharing data stores.

Based on the controller backup and the sharing data store mechanism proposed above, we implement both failure recovery and failure repair process. We use role change mechanism of controller [21-22] defined by OpenFlow protocol for master-backup switch of controller during failure recovery and repair process.

5.4 Complexity

For Algorithm 1, the time complexity is at the level of $O(_{num} \cdot D \cdot T)$, where $_{num}$ is the number of backup controllers, D is the dimension of space, T is the total number of iterations. Algorithm 2 is a dynamic programming algorithm and the time complexity is at the level of $O(N^3)$, where N is the total number of nodes.

6. Simulation Results and Analysis

6.1 Experimental Environment

We evaluate our placement algorithms through simulations. We use the network topology generator Brite [23] to generate multi-domain network topologies with the scale from 1 to 11 domains, and with 10 nodes in each domain. It refers to the topology size of the European backbone network [24]. We set the average delay of the intra-domain link as 0.09 ms, and the average delay of the inter-domain link as 0.5ms. We set the controller failure probability range as [0, 25%], which is based on the fault data of network equipment and network function software [25-29]. We evaluated the performance of the sharing backup controller placement approach and the sharing data store based approach respectively. For the former, we first evaluated our method with different coefficients, and then performed the performance evaluation from aspects of the amount of the backup controller and the average reliability of the network in different network scales and different failure probabilities of the controllers. Finally, we compare the proposed method (SBC) with the method BED, which assigns one backup controller for each domain, and the solver LINGO. And in the latter approach based on the sharing data store, we evaluated it from the perspective of communication delay.

6.2 Evaluation Metrics

The following performance metrics are used to evaluate our approaches:

• The number of the backup controllers: The number of backup controllers that are needed to ensure the reliability of the network.

Controller-switch delay: We use the average controller-switch delay to represent the average value of delays of all backup controllers, as defined in (18), where S_{xy} represents the location a backup controller is placed, and *num* stands for the number of backup controllers.

$$ADCS = \sum_{1<i<D\&i\neq x,\, j\in N} Delay(S_{ij}, S_{xy}) / num \tag{18}$$

• The reliability of the control plane: The sum of the reliability factor of each controller-switch pair, as defined in (5).

• The average reliability of the control plane: The average reliability factor of the control plane, as defined in (19), where S_{xy} represents the location a backup controller is placed, and *num* stands for the number of backup controllers. This metric reflects the network reliability provided by per backup controller.

$$AR = \sum_{v\in N} rf_{S_{xy},v} / num \tag{19}$$

• The Controller-data store delay: we use two metrics to measure the delay between controllers and the data store: the average controller-data store delay and the worst controller-data store delay [30]. The average delay is the average value of all delay from controllers to the primary data store, as defined in (20), where S_{ij} represents the location of controllers, S_{xy} represents the location a backup controller is placed, and the *pairs_CD* stands the number of primary controller-primary data store pairs. And the worst delay is the delay with the largest value.

$$ADCD = \sum_{S_{ij}\in C} delay(S_{ij}, S_{DS}) / pairs\_CD \tag{20}$$

6.3 The Feasibility of the Proposed Approach

We verify whether the network can recover from a controller failure based on the sharing data store approach with no packet loss. First we configure two controllers to take control of two domains respectively, and make one controller down to simulate a controller failure. **Fig. 5** presents the results of three experiments, before the master controller failed, the average delay is about 26ms. After failure occurred, the switches tried to connect with the backup controller and the backup controller got the information of the network state from the sharing data store. Therefore, the delay of receiving packets increased to about 750ms. At last, the average delay came back to about 26ms. Despite there is a rise of delay during the process of controller switching, the ICMP sequence number is consecutive. It suggests there is no packet loss.

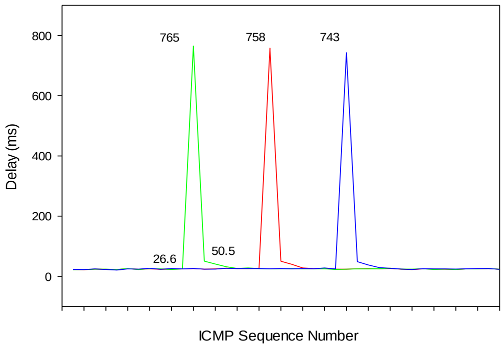


Fig. 5. ICMP packet delay through controller failure and recovery.

6.4 Performance Evaluation of the Sharing Backup Controller Placement Approach

6.4.1 Performance Evaluation in Comparison with Different Coefficients

In this part, we evaluate our approach with different coefficients in formula (16). In order to observe the influence of different coefficients on the performance of the approach, we set the coefficients α , β , and γ range from 0 to 1 while satisfying the constraint specified by formula (17) to observe the results of F_1 , F_2 , F_3 and F . For space limitation, **Table 9** only shows the results of four typical conditions. It can be seen that the value of F is minimum when α =0.2, β =0.3, and γ =0.5. It also can be found the value of F_1 is minimum when the number of backup controllers is the most important (α =1). Similarly the value of F_2 is minimum when the delay is the most important and the value of F_3 is maximum when the reliability of the network is the most significant. Therefore, in the following experiment, we set α =0.2, β =0.3, and γ =0.5.

Table 9. The value of F with different coefficients

F	F_1	F_2	F_3	α	β	γ
0.265	2	5.04	0.92	0.2	0.3	0.5
0.47	1	5.49	0.89	1	0	0
0.34	2	4.68	0.91	0	1	0
0.36	3	6.04	0.94	0	0	1

6.4.2 Performance Evaluation under Different Network Scales and Different Failure Probabilities

Firstly, we evaluate the sharing backup controller placement approach from aspects of the amount of the backup controller and the average reliability of the network in different network scales and different failure probabilities of the controllers. The results are presented in **Fig. 6**, **Fig. 7** and **Fig. 8**. As shown in **Fig. 6**, with the increase of the number of domain, the amount of needed backup controllers will also increase. We can also find that more backup controllers are needed as the failure probability of the primary controller becomes larger. Because larger failure probability will probably lead to more concurrent failures. Therefore, more backup controllers will be needed for failure recovery.

Fig. 7 shows the average controller-switch delay with different network scales and failure probabilities. As illustrated in the figure, with the increase of the number of domain, the average delay increases at the same time. This is because as the number of domains increases, more switches are needed to connect to the backup controllers, and the number of backup controllers grows less rapidly than the number of backup controller-switch pairs. While with the rise of the failure probability of the primary controller, the average delay decreases. This is because the higher the failure probability of the primary controller is, the more backup controllers are needed, resulting in the decrease of the average delay.

Additionally, **Fig. 8** presents the average reliability with different network scales and faulty probabilities. As shown in the figure, as the amount of domains increases, the average reliability of the control plane becomes smaller. At the same time, with the increase of failure probability of the master controller, the average reliability gets smaller too. Under the usual failure probability conditions [5%, 15%], the average reliability of the control plane can reach more than 84%, and under stressful failure rates, the average reliability can also reach about 70%.

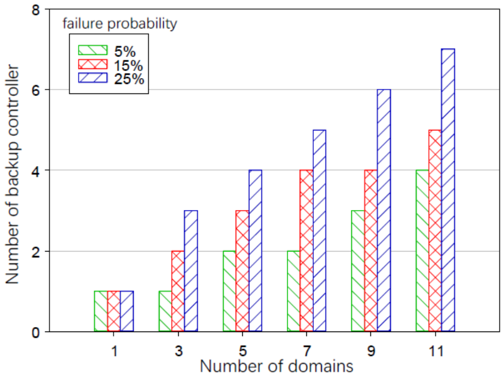


Fig. 6. The number of backup controllers with different network scales and failure probabilities

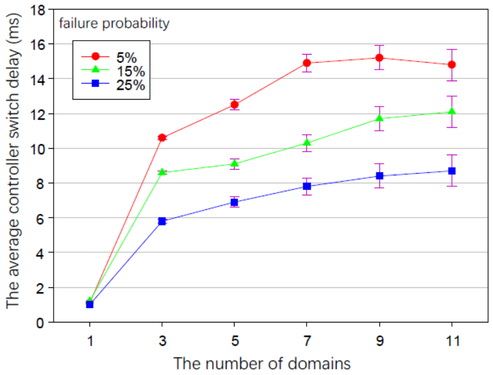


Fig. 7. The average controller-switch delay with different network scales and failure probabilities.

The 95% confidence intervals are also shown

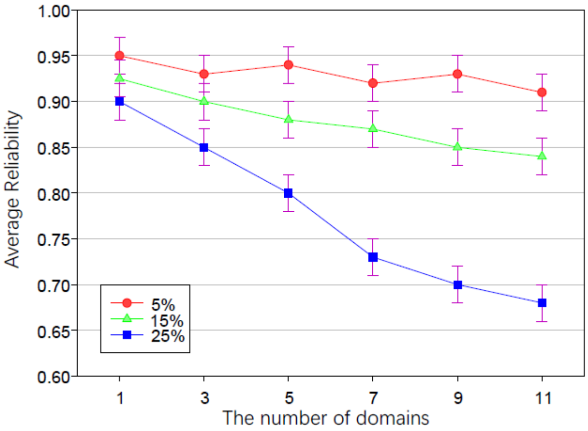


Fig. 8. The average reliability with different network scales and failure probabilities

6.4.3 Performance Evaluation in Comparison with Other Approaches

We compare our sharing backup controller placement algorithm (SBC) with the method BED, which assigns one backup controller for each domain, and the solver LINGO [31]. The detailed introduction and strategy of three approaches is shown in **Table 10.** The networks with 1, 3, 5, 7, 9 and 11 domains are randomly generated. There are 10 nodes in each domain. We set the failure probability of the master controller is set as 0.15. We evaluate the number of the controllers, the reliability of control plane, and the average reliability of three approaches.

Table 10. Compared approaches

Scheme	Instruction and Strategy
SBC	Several domains share one backup controller, and calculate the number of backup controller based on the failure probability, then deploy the backup controller on node with considering delay and reliability of each domain. Our proposed method.
LINGO	A solver tool for solving optimization problems. Import three objective functions and constraints, and then get solution.
BED	Each domain has a backup controller, and deploy the backup controller on second best node with considering delay and reliability of each domain.

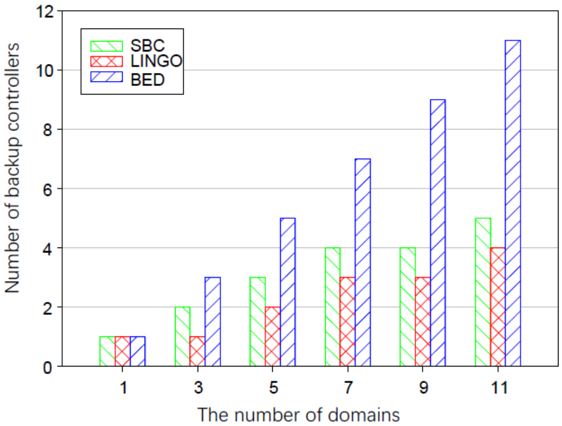


Fig. 9. The number of backup controllers with different number of domains

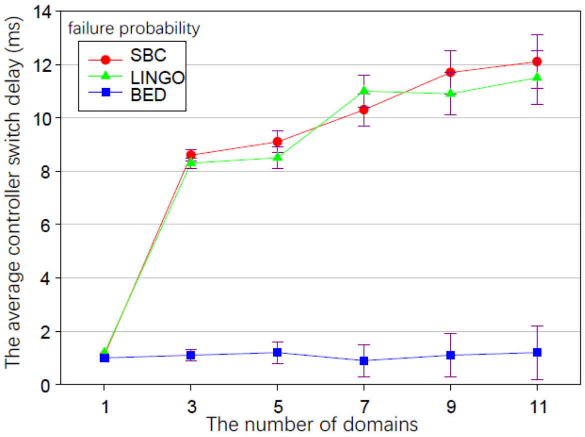


Fig. 10. The average delay of with different number of domains.

Fig. 9 presents the amount of backup controllers. As shown in the figure, the number of backup controllers of our approach SBC and solver LINGO is much less than that of BED with the increase of the number of domains. And compared with SBC, LINGO works out the solutions that require slightly less backup controllers.

As can be seen from **Fig. 10**, as the number of domains increases, the average delays of the three methods also increase at different rates. Since BED requires the largest number of backup controllers, the average delay is the lowest. The delay generated by SBC and the solver LINGO is relatively close.

As for the average reliability of the control plane, namely the reliability provided by per backup controller, as illustrated in **Fig. 11**, we can find that the results of SBC and LINGO are very close and better than that of BED.

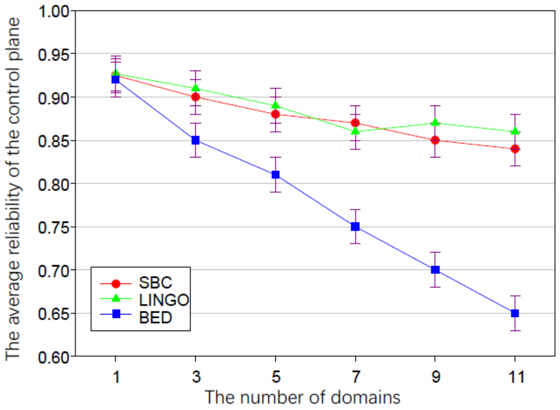


Fig. 11. The average reliability of the control plane in different approaches The 95% confidence intervals are also shown.

At last, we also compare the computation time of our approach and solver LINGO. As shown in **Fig. 12**, our algorithm uses less time than LINGO when the scale of the network gets larger.

Based on the above observations, we can conclude that the proposed method SBC can achieve as good performance as the solver LINGO, both of which are superior to the BED method. At the same time, SBC's computation time is much less than LINGO.

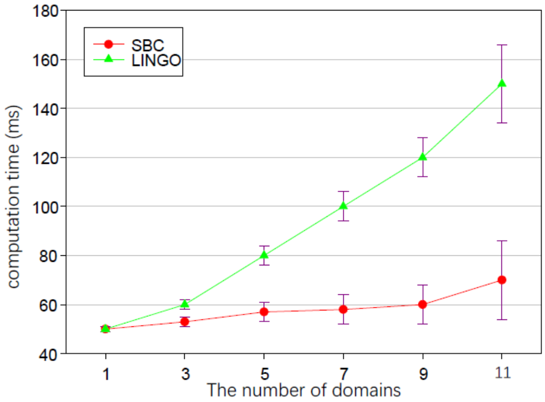


Fig. 12. The computation time of different approaches The 95% confidence intervals are also shown

6.5 Performance Evaluation of the Sharing Data Store Based Approach

We evaluate the data store placement approach from the perspective of communication delay. **Fig. 13** shows the average controller-data store delay and the worst controller-data store delay in different network scales. We can find that both the average delay and worst delay increases as the number of domains gets larger, but all these delays are in an acceptable level (average delay ≤ 50ms, worst delay ≤ 200ms), which will not influence the performance of the network.

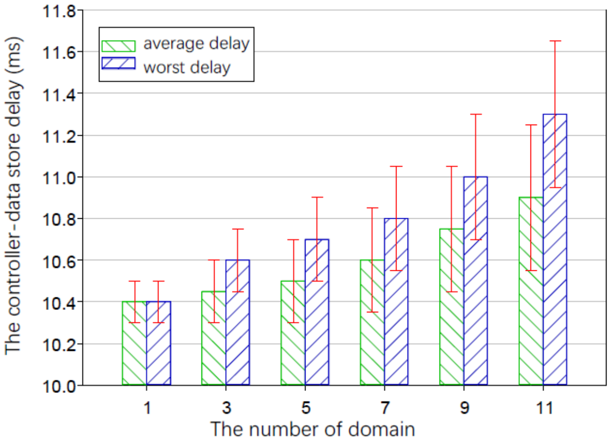


Fig. 13. The average controller-data store delay and the worst controller-data store delay in different network scales. The 95% confidence intervals are also shown.

7. Conclusion

In this paper, we focused on the reliability problem of multiple-domain SDN against controller failure from perspectives of backup controller deployment and controller replication. We firstly propose a placement algorithm for backup controllers based on PSO algorithm, which considers both the reliability and the cost factors. Then we propose a mechanism based on shared data storage for controller replication, which could solve the inconsistency problem between primary and backup controllers. To ensure the reliability and the performance of the control plane, we also proposed an approach for data store placement, which considers both reliability and performance. Besides, based on controller backup and the sharing data store mechanism, we design the processes of failure recovery and repair. We validated our approach through a small-scale testbed. Besides, simulation results also showed that, on one hand, compared with other methods, the proposed backup controller placement approach can improve the reliability of control plane with less resource cost and computation time. On the other hand, our data store based approach can achieve a good delay performance. In the future, we will research load balancing problem based on sharing data store architecture.

References

[1] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114-119, Feb. 2013. Article (CrossRef Link)

[2] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, “Interactive monitoring, visualization, and configuration of OpenFlow-based SDN,” in *Proc. of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 207-215, May 2015. Article (CrossRef Link)

[3] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, “Multi-controller Based Software-Defined Networking: A Survey,” *IEEE Access*, vol. 6, pp. 159a80-15996, Mar. 2018, Article (CrossRef Link)

[4] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Loutievski, M. Zhu, R. Ramanathan, Y. Iwata and H. Inoue, “Onix: A Distributed Control Platform for Large-scale Production Networks,” in *Proc. of the 9th USENIX OSDI*, pp. 351-364, Oct. 2010. Article (CrossRef Link)

[5] A. Tootoonchian and Y. ganjali, “HyperFlow: A Distributed Control Plane for OpenFlow,” *USENIX INM/WREN*, 2010. Article (CrossRef Link)

[6] P. Lin, S. Wolff, Y. Wang, A. Xu, and Z. Chen, “A west-east bridge based SDN inter-domain testbed,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 190-197, Feb. 2015. Article (CrossRef Link)

[7] Y. Jimenez, C. Cervello-Pastor and A. J. Garcia, “On the controller placement for designing distributed SDN control layer,” in *Proc. of Networking Conference*, IFIP. Trondheim, pp. 1-9, June 2014. Article (CrossRef Link)

[8] Fan Y , Ouyang T, “ Reliability-Aware Controller Placements in Software Defined Networks,” in *Proc. of 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems*, pp. 2133-2140, 2019. Article (CrossRef Link)

[9] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, “On reliability-optimized controller placement for Software-Defined Networks,” *Communications*, vol. 11, no. 2, pp. 38-54, Feb. 2014. Article (CrossRef Link)

[10] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, “Survivor: An enhanced controller placement strategy for improving SDN survivability,” in *Proc. of 2014 IEEE Global Communications Conference*, pp. 1909-1915, Dec. 2014. Article (CrossRef Link)

[11] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in *Proc. of 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pp. 1-6, 2011. Article (CrossRef Link)

[12] Y. Wang, Q. Zhong, X. Qiu, and W. Li, “Resource Allocation for Reliable Communication Between Controllers and Switches in SDN,” *Journal of Network & Systems Management*, pp. 966-992, 2018. Article (CrossRef Link)

[13] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, “Resilience of SDNs based on active and passive replication mechanisms,” in *Proc. of IEEE Global Communications Conference*, pp. 188-2193, 2013. Article (CrossRef Link)

[14] E. S. Spalla, D. R. Mafioletti, A. B. Liberato, G. Ewald, C. E. Rothenberg, L. Camargos, R. S. Villace, and M. artinello, “AR2C2: Actively replicated controllers for SDN resilient control plane,” in *Proc. of NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 189-196, 2016. Article (CrossRef Link)

[15] N. Katta, H. Zhang, M. Freedman, and J. Rexford, “Ravana: Controller fault-tolerance in software-defined networking,” in *Proc. of The Symposium on SDN Research (SOSR)*, no. 4, pp. 1-12, 2015. Article (CrossRef Link)

[16] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, “A replication component for resilient OpenFlow-based networking,” in *Proc. of Network Operations and Management Symposium*, pp. 933-939, 2012. Article (CrossRef Link)

[17] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, “On the Feasibility of a Consistent and Fault-Tolerant Data Store for SDNs,” *Second European Workshop on Software Defined Networks, IEEE Computer Society*, pp. 38-43, 2013. Article (CrossRef Link)

[18] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira, “On the Design of Practical Fault-Tolerant SDN Controllers,” in *Proc. of European Workshop on Software Defined Networks*, pp. 73-78, 2014. Article (CrossRef Link)

[19] M. M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, and M. Martinello, “Software defined service function chaining with failure consideration for fog computing,” *Concurrency and Computation Practice and Experience*, 2019. Article (CrossRef Link)

[20] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. of ICNN'95 - International Conference on Neural Networks*, vol.4, pp. 1942-1948, 1995. Article (CrossRef Link)

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

[22] O. N. Foundation. (2011) Openflow switch specification version 1.2.0 (wire protocol 0x04), [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdnresources/

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proc. of Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 346-353, 2001. Article (CrossRef Link)

[24] S. Orlowski, R. Wessaly, M.Pioro, and A. Tomaszewski, "SNDlib 1.0-Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276-286, 2019. Article (CrossRef Link)

[25] P. Gill, N. Jain, and N. Nagappan, “Understanding network failures in data centers: measurement, analysis, and implications,” in *Proc. of the ACM SIGCOMM 2011*, pp. 350-361, 2011. Article (CrossRef Link)

[26] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme," in *Proc. of 2017 IEEE International Conference on Communications*, pp. 1-6, 2017. Article (CrossRef Link)

[27] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-Aware Backup Allocation for A Chain of Virtual Network Functions," in *Proc. of IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1918-1926, 2019. Article (CrossRef Link)

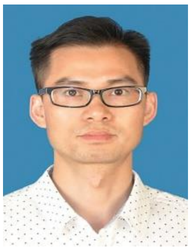
[28] J. Kong, I. Kim, X. Wang, Q. Zhang, H.C. Cankaya, and W. Xie, T. Ikeuchi and J. P. Jue, "Guaranteed-Availability Network Function Virtualization with Network Protection and VNF Replication," in *Proc. of GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1-6, 2017. Article (CrossRef Link)

[29] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of Service Function Chains with on-site backups," in *Proc. of 2017 IEEE/ACM 25th International Symposium on Quality of Service* (IWQoS), pp. 1-10, 2017. Article (CrossRef Link)

[30] B. Heller, R. Sherwood, and N. McKeown, “The controller placement problem,” *Acm Sigcomm Computer Communication Review*, vol. 42, no. 4, pp. 7-12, 2013. Article (CrossRef Link)

[31] LINDO Systems, Inc. [Online]. Available: http://www.lindo.com/ [Accessed: 14-Dec-2020]

Junli Mao received Master,s degree in Electronic and Communication Engineering from Xian Electronics Science and Technology University, China in 2006, she is currently working as Senior Engineer in the 54th research institute of China Electronics Technology Group Corporation. Her research interests include communication network operation support system, network management, SDN management and control.



Lishui Chen received the B.Sc. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2005 and 2011. He is a senior engineer in Science and Technology on Communication Networks Laboratory, Shijiazhuang, China. His main research interests include communications network system, Software Defined Network, information system and cloud computing.

Jiacong Li received the bachelor's degree from Beijing University of Posts and Telecommunications in 2015. She is currently a M.S. candidate at Beijing University of Posts and Telecommunications. Her research interest focuses on Software-Defined Networking (SDN) and cloud computing.

Yi Ge received the bachelor's degree from Beijing Jiaotong University in 2019. She is currently a M.S. candidate at Beijing University of Posts and Telecommunications. Her research interest focuses on the management of future network.