Computers, Materials & Continua CMC, vol.64, no.1, pp.665-679, 2020

Single Failure Routing Protection Algorithm in the Hybrid SDN Network

Haijun Geng[1], Jiangyuan Yao[2,*] and Yangyang Zhang[3]

Abstract: Loop free alternate (LFA) is a routing protection scheme that is currently deployed in commercial routers. However, LFA cannot handle all single network component failure scenarios in traditional networks. As Internet service providers have begun to deploy software defined network (SDN) technology, the Internet will be in a hybrid SDN network where traditional and SDN devices coexist for a long time. Therefore, this study aims to deploy the LFA scheme in hybrid SDN network architecture to handle all possible single network component failure scenarios. First, the deployment of LFA scheme in a hybrid SDN network is described as a 0-1 integer linear programming (ILP) problem. Then, two greedy algorithms, namely, greedy algorithm for LFA based on hybrid SDN (GALFAHSDN) and improved greedy algorithm for LFA based on hybrid SDN (IGALFAHSDN), are proposed to solve the proposed problem. Finally, both algorithms are tested in the simulation environment and the real platform. Experiment results show that GALFAHSDN and IGALFAHSDN can cope with all single network component failure scenarios when only a small number of nodes are upgraded to SDN nodes. The path stretch of the two algorithms is less than 1.36.

Keywords: Multipath routing, network availability, routing protection algorithm, network failure, hybrid SDN network.

1 Introduction

SDN is a newly emerging network architecture, which is characterized by decoupling the functions of control and forwarding planes [Wang, Deng, Ren et al. (2020); Qiu, Zhao, Wang et al. (2019)]. The former consists of one or more SDN centralized controllers responsible for path selection and routing decision [Das, Sridharan and Gurusamy (2020); Zhang, Cui and Wang (2018)]. The latter is composed of SDN switches responsible for forwarding the packets in the network. SDN refers to the idea of software programming. The centralized SDN controller communicates with the SDN switches through the standardized OpenFlow protocol [Astuto, Mendonca, Nguyen et al. (2014); Shi, Li, Xie et

[1] School of Software Engineering, Shanxi University, Taiyuan, 030006, China.

[2] School of Computer Science and Cyberspace Security, Hainan University, Haikou, 570228, China.

[3] College of Engineering Northeastern University, Boston, 02115, USA.

[*] Corresponding Author: Jiangyuan Yao. Email: yaojy@hainanu.edu.cn.

Received: 27 January 2020; Accepted: 13 April 2020.

CMC.doi:10.32604/cmc.2020.09912 www.techscience.com/journal/cmc

al. (2020)]. The SDN centralized controller has a logical view of the whole network controlling the forwarding path of all packets in the network [Zhang, Cheng and Lin (2017)]. The controller sends the routing decision information to the SDN switches through OpenFlow protocol [Rubio, Galis, Astorga et al. (2011); Zheng, Xu, Zhu et al. (2019)]. Compared with the traditional network architecture, the SDN network has many advantages, such as flexible control of network traffic and easy to implement network management and security policies [Scott-Hayward, Natarajan and Sezer (2016)]. Therefore, the performance of the network can be greatly improved when the SDN technology is deployed in the existing traditional network [Sezer, Scott-Hayward, Chouhan et al. (2013)]. However, upgrading all current network devices to SDN devices is impossible. On the one hand, deploying SDN devices needs considerable economic costs, such as human and material resources. On the other hand, deploying SDN devices may lead to network interruption and further seriously affect the user's experience. In academia and industry, SDN technology is widely used to upgrade the traditional network and gradually replace the traditional equipment in the network. The network where traditional and SDN devices coexist is regarded as the hybrid SDN network [Vissicchio, Vanbever, Cittadini et al. (2017)]. A hybrid SDN network mainly includes SDN controller, SDN switch, and traditional router [Xu, Li, Huang et al. (2017)], as shown in Fig. 1. The SDN controller exchanges information with SDN switch through the OpenFlow protocol. SDN switch can work in two modes; one is to interact with the traditional router through traditional routing protocols, and the other is to interact with SDN switch and controller through SDN protocols [Xu, Huang, Chen et al. (2018)]. However, the traditional router only supports the traditional routing protocol and cannot communicate with the SDN controller. Existing studies on a hybrid SDN network are mainly focused on network performance and management. However, studies on fast re-routing in a hybrid SDN network [Salsano, Ventre, Lombardo et al. (2016)]. Routing availability and energy efficient routing are two key problems in the network. This paper mainly studies routing availability. Therefore, this study concentrates on how to implement fast re-routing technology in a hybrid SDN network. This technology can immediately respond to network failures and reduce the packet loss caused by such failures.

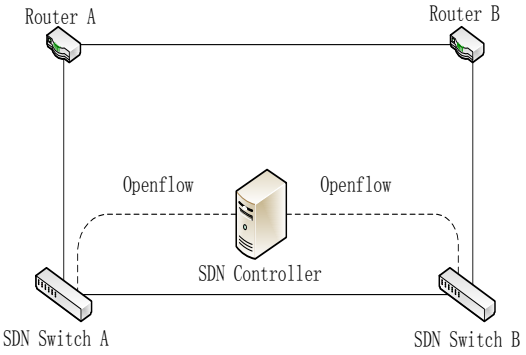

Figure 1: Hybrid SDN network architecture

The structure of the paper is organized as follows: Section 2 describes the related works. Section 3 introduces the network model and describes the problem which will be solved in this study. Section 4 proposes two algorithms showing how to deploy LFA scheme in a

hybrid SDN network. Section 5 carries out experiment simulation in different network topologies and summarizes the experiment results. Finally, Section 6 concludes.

2 Related works

A considerable number of studies have shown that 70% of the failures in the network are single-link failures, and the remaining 30% are single-node and concurrent failures [Amin, Amin and Shah (2016); Foerster, Pignolet, Schmid et al. (2018); Zheng, Xu, Zhu et al. (2019)]. Hence, this study focuses on how to deal with the single-link failure in the network. Routing protection scheme is widely employed in academia and industry to deal with frequently occurring network failures [Geng, Shi, Wang et al. (2018)]. Equal cost multipath routing (ECMP) is the earliest and simplest route protection scheme employed in the industry. However, numerous studies have shown that ECMP cannot provide a high network failure protection ratio [Geng, Shi, Wang et al. (2017); Yang, Xu and Li (2018)]. To overcome this, the internet engineering task force (IETF) puts forward the framework of fast re-routing. From this framework, studies have proposed LFA, Not-via, backup tunnel, and so on. Among all routing protection schemes, LFA has been paid close attention to the industry because of its simplicity and has been deployed and supported by router manufacturers, such as Cisco, Juniper, and Huawei. LFA is simple and easy to deploy. However, LFA has a fatal disadvantage, that is, it cannot protect all possible single network failure scenarios in the network. To overcome this, this study analyzes the problem of LFA failure coverage by using the theoretical knowledge of graph theory in the literature. In addition, this study increases the LFA failure protection ratio by adjusting the link weight in the network. However, the proposed method still cannot guarantee to deal with all single failure scenarios. Therefore, the relationship between LFA's failure protection ration and network topology is theoretically analyzed in detail, and LFA can cope with all single failure scenarios by adding links to the network. Braun et al. [Braun and Menth (2016)] studied how to deploy LFA in SDN network, so as to deal with all possible single failure scenarios. However, all schemes are based on the traditional network architecture or SDN network, which cannot be directly applied to the hybrid SDN network.

Therefore, this study analyzes how to deploy LFA in a hybrid SDN network and ensures that LFA can deal with all possible single failure scenarios in the network. This study proposes the following: First, all source-destination pairs that are not protected by the LFA are calculated and protected by deploying SDN nodes in the network, so that the solution can cope with all single network failure scenarios.

Fig. 2 depicts the key idea of this study. As shown in the figure, the shortest path from source s to destination d is $(s, a, \ldots, d)$, and the SDN node is c. If no failure in the network occurs and a packet is forwarded from the source s to the destination d, then the forwarding path of the packet is $(s, a, \ldots, d)$. When the link (s, a) fails, the source s will first forward the packet to the SDN node c, and then, the SDN node c forwards the packet to the destination node d.

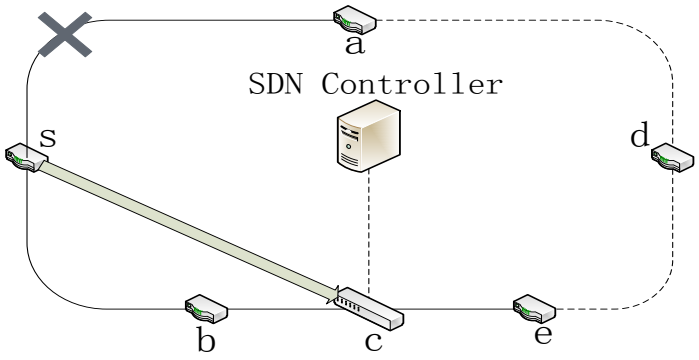

Figure 2: Key idea of the study

3 Network model and problem description

3.1 Network model

A network can be modeled as a graph $G = (V, E)$, where V and E are respectively the set of nodes and edges in the network. For any node $\forall v \in V$ in the network, $N(v)$ represents all the neighboring nodes of the node v, $spt(v)$ is the shortest path tree (SPT) rooted at node v. For $\forall x, y \in V (x \neq y)$, $sp(x, y)$ is the collection of nodes on the shortest path from node x to y, $cost(x, y)$ represents the shortest cost from node x to y in the network, and $dn(x, y)$ is the best next hop from node x to y.

3.2 Problem description

The intra-domain routing protocols deployed on the Internet are mainly link-state routing protocols, such as intermediate system to intermediate system (IS-IS) and open shortest path first (OSPF). In both routing protocols, all routers in the network have a complete topology within the autonomous domain. When the network is in a stable state, the topologies stored in all routers are consistent with one another. Each router in the network employs the shortest path first algorithm to compute shortest path tree (SPT) on the basis of network topology. Then, a routing table is constructed using the SPT. From the above description, the intra-domain routing protocol uses the shortest path to forward packets. When the network component fails, network interruption will occur, which seriously affects the performance of the network. Therefore, the IETF proposes to use LFA to deal with frequently occurring failures in the network to improve network availability and improve the user's experience. The three rules in the LFA are described separately as follows:

Loop-free criterion (LFC): x can be selected as a valid next hop from c to d in the case of $cost(x, d) < cost(x, c) + cost(c, d)$. When packets are routed from x to d, they are not routed back to c because $cost(x, c) + cost(c, d)$ is the lowest cost of any path from x to d that passes c. Thus, the protection route bypasses c, thereby also bypassing the link (c, b).

Node-protecting criterion (NPC): x can be selected as a valid next hop from c to d when $cost(x, d) < cost(x, f) + cost(f, d)$, which means that the protection route bypasses f, thereby also bypassing the node b.

Downstream path criterion (DC): x can be selected as a valid next hop from c to d when $cost(x,d) < cost(c,d)$, which means that the protection route bypasses c (and link (c,b)), and the remaining cost to the destination strictly decreases.

Studies have reported that not all source-destination node pairs have an optional LFA next hop. Therefore, to overcome the inherent problem of LFA, this study mainly solves how to deploy LFA scheme in a hybrid SDN network, so that it can protect all possible single failure scenarios in the network. We can express the problem solved in this study as follows: how to choose a group of nodes to deploy the SDN technology for a given network so that LFA can cope with all possible single link failure scenarios in the network. The above problem can be described as a 0–1 ILP model, namely:

Minimize:

$$\sum_{i\in V} x(i), \tag{1}$$

Subject to:

$$y(i,j,k) \in \{0,1\} \quad i,j,k \in V, \tag{2}$$

$$y(i,j,k) = 0 \quad i \notin S(j,k), \tag{3}$$

$$y(i,j,k) = 1 \quad i \in S(j,k), \tag{4}$$

$$x(i) \in \{0,1\} \quad i \in V, \tag{5}$$

$$x(i) = 0 \quad \sum_{(j,k)\in V} y(i,j,k) = 0, \tag{6}$$

$$x(i) = 0 \quad \sum_{(j,k)\in V} y(i,j,k) \geq 1, \tag{7}$$

$$f(i,j) \in \{0,1\} \quad i,j \in V, \tag{8}$$

$$f(i,j) = 0 \quad (i,j) \in E, \tag{9}$$

$$f(i,j) = 1 \quad (i,j) \notin E, \tag{10}$$

$$z\big((i,j),i,d\big) \in \{0,1\} \quad i,j,d \in V, \tag{11}$$

$$z\big((i,j),i,d\big) = 0 \quad (i,j) \notin sp(i,d), \tag{12}$$

$$z\big((i,j),i,d\big) = 1 \quad (i,j) \in sp(i,d), \tag{13}$$

$$\sum_{i\in V} y(i,j,k) \leq 1, \tag{14}$$

$$y(i,j,k) \leq x(i), \tag{15}$$

$$y(i,j,k) = 1 \quad x\Big(\big(j,dn(j,k)\big),j,i\Big) + \prod_i x\Big(\big(j,dn(j,k)\big),N(i),k\Big) = 0, \tag{16}$$

In the next section, we will explain the above ILP model in detail. In the model, Eq. (1) is the objective function, that is, the number of deployed SDN nodes is minimized. $x(i)$

indicates that whether the node i is an SDN node or not. If $x(i) = 1$, then the node i is an SDN node; otherwise, the value is 0. The variable $y(i,j,k)$ in Eqs. (2)-(4) indicates that whether node i is the SDN node of the source–destination pair of $j-k$ or not. If the node i is the SDN node of the source-destination pair of $j-k$, then the value of $y(i,j,k)$ is 1; otherwise, the value is 0. Eq. (6) indicates that if the node i is not an SDN node of any source-destination pair of $j-k$, then the value of $x(i)$ is 0; otherwise, Eq. (7) is correct. The variable $f(i,j)$ in Eqs. (8)-(10) indicates whether the link (i,j) is in the network or not. If the link (i,j) is in the network, then the value of $f(i,j)$ is 1; otherwise, the value is 0. The variable $z\big((i,j),i,d\big)$ in Eqs. (11)-(13) indicates whether the link (i,j) is on the shortest path from node i to j or not. If the link (i,j) is on the shortest path from node i to j, then the value is 1; otherwise, the value is 0. Eq. (14) means that only one SDN node is eventually selected for any source-destination pairs. Eq. (15) indicates that the node i will be upgraded to an SDN node if an arbitrary source-destination pair selects the node i as its SDN node. Eq. (16) indicates that node i is the SDN node of the source-destination pair of $j-k$, and the following two conditions must be satisfied: (1) the link $\big(j,dn(j,k)\big)$ is not on the shortest path from the node j to i; (2) the link $\big(j,dn(j,k)\big)$ is not on the shortest path from the node $m, m \in N(i)$ to k.

4 Algorithms

The problem that needs to be solved in this study has been proven to be a non-deterministic polynomial (NP) problem. Hence, obtaining the optimal solution in a reasonable time is impossible. For some small networks, we can use the linear programming calculator, for example, CPLEX, to obtain the optimal solution. However, for large networks, computing the optimal solution in a reasonable time by CPLEX is difficult, which is not suitable for deploying in the real networks. Therefore, we usually use heuristic algorithms to solve the above problem in practice. In the following section, we will introduce two heuristic algorithms, namely, GALFAHSDN and IGALFAHSDN, to solve the scientific problem. The former is a traditional greedy algorithm but has high time complexity and is not suitable for actual deployment on the Internet. The latter has improved the traditional greedy algorithm, which can greatly reduce the time complexity. Therefore, IGALFAHSDN is easier to deploy than GALFAHSDN in the real network. Considering the three rules in the LFA, the following algorithms are mainly designed for the LFC rule. Once you want to implement the other two rules in the LFA, you only need to do is to replace the LFC with NPC or DC and do not need to change the rest of the algorithms.

Algorithm 1 GALFAHSDN
Input:
$G = (V, E)$
Output:
SDN node set M
1: $M = \phi$
2: Compute the failure protection ratio of LFC $R(G, M)$
3: While $R(G, M) < 1$ and $M \neq V$, do

4: $k = \arg max\, R_{v \in nei(V)}(G, M \cup v)$
5: $M = M \cup k$
6: Compute the failure protection ratio $R(G, M)$
7: EndWhile
8: Return M

4.1 Greedy algorithm

In this section, we will introduce how to solve the above problem with greedy algorithm. Algorithm 1 describes how GALFAHSDN works. First, we set the initial value of deployment SDN node set to NULL ($M = \phi$) and then compute the initial failure protection ratio $R(G, M)$ according to LFC rule (lines 1-2). To obtain the set of deployed SDN node set M, the algorithm needs to perform a series of iterations until the failure protection ratio $R(G, M) = 1$ or $M = V$ is established. The function $nei(V)$ is used to randomly select a node in the set $v \in \{V/M\}$ to deploy the SDN technology. The function $\arg max\, R_{v \in nei(V)}(G, M \cup v)$ returns the node k with the maximum failure protection ratio. Then, node k is added to the set M, and the network failure protection ratio is updated at the same time (lines 3-7). Finally, the deployed SDN node set M is returned (line 8).

4.2 IGALFAHSDN

Algorithm 2 IGALFAHSDN
Input:
$G = (V, E)$
Output:
SDN node set M
1: Calculate source-destination pairs $L = \{(s, d), s, d \in V\}$ that are not protected by LFC
2: Calculate SDN nodes $D(s, d)$ for each source–destination pair $(s, d) \in L$
3: Calculate $\sum_{(i,j,k) \in V} y(i, j, k)$
4: $M = \phi$
5: While $R(G, M) < 1$ and $M \neq V$ do
6: $m = max \sum_{(i,j,k) \in V} y(i, j, k)$
7: $M = M \cup m$
8: For $(s, d) \in L$
9: If $m \in D(s, d)$ then
10: Clear $D(s, d)$
11: $L = L - \{s, d\}$
12: EndIf
13: EndFor
14: Update $\sum_{(i,j,k) \in V, i \neq m} y(i, j, k)$
15: Update $R(G, M)$

16: EndWhile
17: Return M

The above algorithm GALFAHSDN is a typical greedy algorithm. To select a node from the network to deploy SDN technology, the algorithm needs to go through several iterations, so that the time complexity of the algorithm is extremely high in large networks. We propose an improved greedy algorithm, IGALFAHSDN, to reduce the complexity of the algorithm GALFAHSDN, thus making the algorithm easy to deploy in the real network. We will describe the algorithm IGALFAHSDN in detail. First, we calculate all source-destination pairs in the network that are not protected by the LFC rule and store them in the variable $L = \{(s,d), s,d \in V\}$ (line 1). Then, we calculate all SDN nodes for each source-destination $(s,d) \in L$ according to Eq. (16) and stored them in the variable $D(s,d)$. The value of $\sum_{(i,j,k)\in V} y(i,j,k)$ is calculated, and the initial value of deployment SDN node was set to NULL ($M = \phi$) (lines 2-4). To obtain the set of deployed SDN node set M, the algorithm needs to go through several iterations until the failure protection ratio $R(G,M) = 1$ or $M = V$ is established. In each iteration, the node m with the largest $\sum_{(i,j,k)\in V} y(i,j,k)$ was selected to deploy SDN technology and update the node set M (lines 6-7). Each pair of source-destination node only selects a unique SDN node. Thus, for any source-destination node pair, if $m \in D(s,d)$, then the SDN node for this source-destination pair is determined, and computing SDN nodes for it is not necessary. The value of $D(s,d)$, L, $\sum_{(i,j,k)\in V, i\neq m} y(i,j,k)$ and $R(G,M)$ is calculated (lines 8-16). Finally, the deployed SDN node set M is returned (line 17).

4.3 Algorithm complexity

This section theoretically analyzes the time complexity of the algorithms GALFAHSDN and IGALFAHSDN.

Theorem 1. The time complexity of the algorithm GALFAHSDN is $O(|V|^5)$.

Proof. To obtain the node set that is needed to deploy SDN technology, the algorithm must execute the function $\arg max\, R_{v\in nei(V)}(G, M\cup v)$ at most $|V|$ times. The function needs to calculate the SDN nodes for all source-destination pairs in the network, and the algorithm complexity is $O(|V|^4)$. Therefore, the time complexity of GALFAHSDN is $O(|V|^5)$.

Theorem 2. The time complexity of the algorithm IGALFAHSDN is $O(|V|^3)$.

Proof. The time complexity of the second line of the IGALFAHSDN is $O(|V|^3)$. The time complexity of lines 5-15 of the IGALFAHSDN is $O(|V|^2)$. Therefore, the time complexity of the algorithm IGALFAHSDN is $O(|V|^3)$.

5 Performance evaluations

LFA has three rules. However, the implementation methods of the three rules are basically similar. The experiment results are basically the same, so this section only lists the experiment results of LFC rule in LFA. We will not list the experiment results of the other two rules in detail.

In this section, we will use experiments to test the performance of algorithms

GALFAHSDN and IGALFAHSDN. The evaluation indicators include the number of SDN nodes, failure protection ratio, computational overhead, and path stretch. As the number of SDN nodes deployed in the network decreases, the deployment overhead also decreases. In the experiment, we compare the failure protection ratio of LFC, GALFAHSDN and IGALFAHSDN. If the corresponding failure protection ratio of an algorithm is 100%, then the algorithm can deal with all possible single failure scenarios in the network; otherwise, the algorithm cannot protect some of the failures. We have theoretically analyzed the time complexity of algorithms GALFAHSDN and IGALFAHSDN in Section 4. In this section, we will use the real computation time of the algorithm to compare the overhead of algorithms GALFAHSDN and IGALFAHSDN. In the experiment, we compared the path stretch of GALFAHSDN with that of IGALFAHSDN. The path stretch directly affects the network delay and path overhead, so we hope that the path stretch of the algorithm is as small as possible. We first describe the network topology of the algorithm, then introduce the experiment results, and last analyze the experiment results in detail.

To evaluate the performance of GALFAHSDN, IGALFAHSDN, and LFC, we run three algorithms on a considerable number of topologies. The experiment topology of this paper includes the following three types of topologies:

(1) Real network topology. In this type of network topology, we select three real network topologies, that is, Abilene, TORONTO, and USLD.

(2) The topologies measured using Rocketfuel. In this type of network topology, we select four measurement topologies, that is, AS1755, AS1239, AS3257, and AS3967.

(3) The topology generated by the simulation software Brite. The model used by Brite is Waxman, the number of nodes in the topology ranges from 100 to 500, the parameter of alpha and beta is set to 0.15 and 0.2 respectively, the average node degree parameter of the network ranges from 2 to 4, and the distribution of nodes in the network is subject to heavy tailed distribution. The bandwidth parameter of the link ranges from 10 to 1024.

5.1 Number of SDN nodes

In this study, SDN nodes are deployed in the traditional network to improve the failure protection ratio of LFC, but the deployment of SDN nodes requires additional overhead. If the number of deployed SDN nodes is less, then the additional overhead is also less. Therefore, in this section, we will discuss the number of SDN nodes that need to be deployed for LFC to cope with all possible single failure scenarios in a hybrid SDN network.

Tab. 1 shows the number of SDN nodes deployed in different network topologies. In Tab. 1, Brite (m, n) represents the topology generated by the Brite, the number of nodes is m, and the average node degree of the network is n. In all networks, except for Abilene, only a few SDN nodes need to be deployed to achieve a full failure protection ratio. From the above results, we can draw a conclusion that many SDN nodes are deployed in a sparse graph. For example, in the Abilene topology, the number of SDN nodes needs to be deployed is 5, and nearly half of the nodes need to be upgraded to SDN nodes. This finding is because the Abilene topology and the average node degree are relatively small. In the dense graph, the number of SDN nodes deployed is relatively small. For example, the number of SDN

nodes that need to be deployed is 18 in Brite (500, 2), whereas the number of SDN nodes that need to be deployed is only 4 in Brite (500, 4). This event has two reasons: (1) if the network topology is dense, then the source-destination pairs are protected by LFA, so the number of SDN nodes that need to be deployed is small. (2) If the network is dense, then the re-routed path will less likely bypass the failure component.

Table 1: Number of SDN nodes

Network	The number of SDN nodes
Abilene	5
USLD	4
TORONTO	2
AS1239	13
AS1755	9
AS3257	12
AS3967	8
Brite (100, 2)	6
Brite (100, 4)	2
Brite (200, 2)	9
Brite (200, 4)	2
Brite (300, 2)	14
Brite (300, 4)	3
Brite (400, 2)	17
Brite (400, 4)	3
Brite (500, 2)	18
Brite (500, 4)	4

5.2 Failure protection ratio

In this section, we will use the failure protection ratio to measure the ability of LFC, GALFAHSDN, and IGALFAHSDN to cope with network failures. Tab. 2 shows the corresponding failure protection ratio of the three algorithms in different network topologies. Tab. 2 shows that the failure protection ratio of LFC is less than 100%. Algorithms GALFAHSDN and IGALFAHSDN can greatly improve LFC in a hybrid SDN network. The failure protection ratio of GALFAHSDN and IGALFAHSDN is much higher than that of LFC. In this section, we will verify how much higher such ratio is than that of LFC through experiments. From this experiment, the failure protection ratio of algorithms GALFAHSDN and IGALFAHSDN are 100%, that is, they can cope with all possible single failure scenarios in the network. However, the failure protection ratio of LFC is not 100%. Even in the Abilene topology, the failure protection ratio of LFC is only 65.45% because algorithms GALFAHSDN and IGALFAHSDN deploy a small number of SDN nodes in the traditional network, making the packet forwarding more flexible. When a packet encounters a failure element in the process of forwarding, the node can forward the packet to a specific SDN node, and the SDN node will forward the packet to the destination node eventually.

Table 2: Failure protection ratio

Network	GALFAHSDN/%	IGALFAHSDN/%	LFC/%
Abilene	100	100	65.45
USLD	100	100	85.58
TORONTO	100	100	98.83
AS1239	100	100	85
AS1755	100	100	70.76
AS3257	100	100	64.42
AS3967	100	100	80.07
Brite (100, 2)	100	100	90.13
Brite (200, 2)	100	100	91.45
Brite (300, 2)	100	100	90.54
Brite (400, 2)	100	100	92.13
Brite (500, 2)	100	100	92.67

5.3 Computation overhead

Table 3: Computation overhead

Network	GALFAHSDN (ms)	IGALFAHSDN (ms)
Abilene	0.003	0.0001
USLD	0.007	0.001
TORONTO	0.001	0.0001
AS1239	289.788	0.599
AS1755	5.956	0.158
AS3257	31.833	1.922
AS3967	1.53	0.046

This study proposes to deploy LFC scheme in hybrid SDN network by using algorithms GALFAHSDN and IGALFAHSDN. In Section 4, we theoretically analyze the time complexity of the above algorithms. In this section, we will use the real computation time of different algorithms to measure their computation overhead. The experiment platform is a PC with Inter(R) Core (TM) i7-5500U and 4 GB of memory running on a 64-bit Windows 10 operating system. Tab. 3 shows computation overhead of IGALFAHSDN and IGALFAHSDN in real network topologies and Rocketfuel measured topologies. Tab. 3 shows that the computation overhead of IGALFAHSDN is much lower than that of GALFAHSDN. For example, in AS1239, the computation overhead of IGALFAHSDN is 1/100 times of GALFAHSDN. Therefore, IGALFAHSDN greatly reduces the computation overhead of the algorithm and is easier to deploy in practice. This result is because GALFAHSDN selects the best node to deploy SDN technology every time, which requires a considerable computation overhead. However, IGALFAHSDN uses the previously calculated results to decide which node to deploy SDN technology, instead of

calculating it from scratch.

Fig. 3 depicts the computation overhead results of algorithms GALFAHSDN and IGALFAHSDN in the Brite topologies. Fig. 3 shows that as the network topology size increases, the computation overhead of GALFAHSDN increases, and the computation overhead of IGALFAHSDN does not change substantially. The computation overhead of IGALFAHSDN is smaller than that of GALFAHSDN. To further verify the performance of the two algorithms deployed in the real network, we first installed the router software Quagga and Click on 28 computers to simulate the real router. Then, both were connected in accordance with the Abilene, USLD, and TORONTO topologies. We then run the algorithms GALFAHSDN and IGALFAHSDN on the real experiment environment.

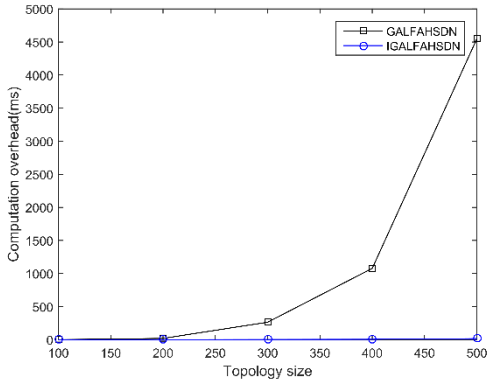

Figure 3: Computation overhead in Brite topologies (ms)

Table 4: Computation overhead on real networks

Network	GALFAHSDN (ms)	IGALFAHSDN (ms)
Abilene	0.0035	0.00017
USLD	0.0073	0.0011
TORONTO	0.0016	0.00013

Tab. 4 shows the computation overhead of algorithms GALFAHSDN and IGALFAHSDN. Tab. 4 shows that the computation overhead corresponding to the experiment is basically the same as that on a PC. The above 28 computers are configured with an Inter(R) Core (TM) i7-5500U and a 4 GB PC running the 64-bit Windows 10 operating system.

5.4 Path stretch

This section measures the path overhead of algorithms GALFAHSDN and IGALFAHSDN by path stretch. In this experiment, we define the path stretch as the ratio of the path cost calculated by the algorithm GALFAHSDN or IGALFAHSDN to the shortest path cost calculated by OSPF when the network failure occurs. Tab. 5 lists the path stretch of both algorithms in three simulation topologies, which is less than 1.36. For example, in Abilene topology, the path stretch is only 1.034, which is nearly the same as the cost of the shortest path calculated employing OSPF convergence method.

Table 5: Path stretch

Network	GALFAHSDN	IGALFAHSDN
Abilene	1.034	1.034
USLD	1.234	1.234
TORONTO	1.145	1.145
AS1239	1.232	1.232
AS1755	1.256	1.256
AS3257	1.145	1.145
AS3967	1.231	1.231
Brite (100, 2)	1.232	1.232
Brite (200, 2)	1.321	1.321
Brite (300, 2)	1.352	1.352
Brite (400, 2)	1.143	1.143
Brite (500, 2)	1.245	1.245

6 Conclusion

In this study, the deployment of LFA in the hybrid SDN network is described as a 0-1 ILP problem. Then, algorithms GALFAHSDN and IGALFAHSDN are proposed to solve this problem. However, the time complexity of GALFAHSDN is extremely high and not suitable for deployment in the real network. The experiment results show that the algorithm IGALFAHSDN proposed in this study not only has less computation overhead but also has low path stretch and high failure routing protection ratio. IGALFAHSDN can protect all possible single failure scenarios in the network without introducing excessive computation overhead just by upgrading a small number of nodes in the network to SDN nodes. Therefore, IGALFAHSDN is a competitive routing protection algorithm for internet service providers.

Funding Statement: This work is supported by the Program of Hainan Association for Science and Technology Plans to Youth R & D Innovation (No. QCXM201910), the National Natural Science Foundation of China (No. 61702315, No. 61802092), the Scientific Research Setup Fund of Hainan University (No. KYQD (ZR) 1837), the Key R & D program (international science and technology cooperation project) of Shanxi Province China (No. 201903D421003), Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi (No. 201802013).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Amin, R.; Amin, M.; Shah, N. (2018): Hybrid SDN networks: a survey of existing approaches. *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3259-3306.

Astuto, B. N.; Mendonca, M.; Nguyen, X. N.; Obraczka, K.; Turletti, T. (2014): A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1617-1634.

Braun, W.; Menth, M. (2016): Loop-free alternates with loop detection for fast reroute in software-defined carrier and data center networks. *Journal of Network and Systems Management*, vol. 24, no. 3, pp. 470-490.

Das, T.; Sridharan, V.; Gurusamy, M. (2020): A survey on controller placement in SDN. *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472-503.

Foerster, K. T.; Pignolet, Y. A.; Schmid, S.; Tredan, G. (2018): Local fast failover routing with low stretch. *ACM SIGCOMM Computer Communication Review,* vol. 48, no. 1, pp. 35-41.

Geng, H. J.; Shi, X. G.; Wang, Z. L.; Yin, X. (2018): A hop-by-hop dynamic distributed multipath routing mechanism for link state network. *Computer Communications*, vol. 116, no. 4, pp. 225-239.

Geng, H. J.; Shi, X. G.; Wang, Z. L.; Yin, X.; Yin, S. P. (2017): Algebra and algorithms for multipath QoS routing in link state networks. *Journal of Communications and Networks*, vol. 19, no. 2, pp. 189-200.

Qiu, K.; Zhao, J.; Wang, X.; Fu, X. M.; Secci, S. (2019): Efficient recovery path computation for fast reroute in large-scale software defined networks. *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1755-1768.

Rubio, L. J.; Galis, A.; Astorga, A.; Serrat, J.; Lefevre, L. et al. (2011): Scalable service deployment on software-defined networks. *IEEE Communications Magazine*, vol. 49, no. 12, pp. 84-93.

Salsano, S.; Ventre, P. L.; Lombardo, F.; Siracusano, G.; Gerola, M. et al. (2016): Hybrid IP/SDN network: open implementation and experiment management tools. *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 138-153.

Scott-Hayward, S.; Natarajan, S.; Sezer, A. S. (2016): A survey of security in software defined networks. *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 623-654.

Sezer, S.; Scott-Hayward, S.; Chouhan, P. K.; Fraser, B.; Lake, D. et al. (2013): Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36-43.

Shi, X. J.; Li, Y. Y.; Xie, H. Y.; Yang, T. F.; Zhang, L. C. et al. (2020): An openflow-based load balancing strategy in SDN. *Computers, Materials & Continua*, vol. 62, no. 1, pp. 385-398.

Vissicchio, S.; Vanbever, L.; Cittadini, L.; Xie, G. G.; Bonaventure, O. (2017): Safe update of hybrid SDN networks. *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1649-1662.

Wang, X.; Deng, Q.; Ren, J.; Malboubi, M.; Wang, S. et al. (2020): The joint optimization of online traffic matrix measurement and traffic engineering for software-defined networks. *IEEE/ACM Transactions on Networking*, vol.28, no. 1, pp. 234-247.

Xu, H.; Li, X. Y.; Huang, L.; Deng, H.; Huang, H. et al. (2017): Incremental deployment and throughput maximization routing for a hybrid SDN. *IEEE/ACM*

Transactions on Networking, vol. 25, no. 3, pp. 1861-1875.

Xu, H. L.; Huang, H.; Chen, S. G.; Zhao, G. M.; Huang, L. S. (2018)**:** Achieving high scalability through hybrid switching in software defined networking. *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 618-632.

Yang, Y.; Xu, M. W.; Li, Q. (2018): Fast re-routing against multi-link failures without topology constraint. *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 384-397.

Zhang, X.; Cheng, Z.; Lin, R. P. (2017): Local fast reroute with flow aggregation in software defined networks. *IEEE Communications Letters*, vol. 21, no. 4, pp. 785-788.

Zhang, Y.; Cui, L.; Wang, W. (2018): A survey on software defined network with multiple controllers. *Journal of Network and Computer Applications*, vol. 103, no. 3, pp. 101-118.

Zheng, J.; Xu, H.; Zhu, X.; Chen, G.; Geng, Y. (2019): Sentinel: failure recovery in centralized traffic engineering. *IEEE/ACM Transactions on Networking*, vol.27, no. 5, pp. 1859-1872.