Advances in Science, Technology and Engineering Systems Journal Vol. 4, No. 4, 388-393 (2019)

www.astesj.com

Special Issue on Advancement in Engineering and Computer Science

ASTESJ

ISSN: 2415-6698

A Proposal of Control Method Considering the Path Switching Time of SDN and Its Evaluation

Kosuke Gotani[1], Hiroyuki Takahira[1], Misumi Hata[1, 2], Luis Guillen[1], Satoru Izumi[*, 1], Toru Abe[1, 3], Takuo Suganuma[1, 3]

[1]*Graduate School of Information Sciences, Tohoku University, 980-8577, Japan*

[2]*Japan Society for the Promotion of Science, 980-8577, Japan*

[3]*Cyberscience Center, Tohoku University, 980-8577, Japan*

A R T I C L E   I N F O

Article history:
Received: 17 June, 2019
Accepted: 31 July, 2019
Online: 16 August, 2019

Keywords:
Software Defined Network
OpenFlow
Path Switching Time

A B S T R A C T

Recently, communication demands often change because of the various network services in companies and individuals. Software Defined Networking (SDN) has emerged as a viable control paradigm that allows flexible communication, using OpenFlow as its default standard and enabler. However, when changes happen frequently in SDN networks due to unforeseen reasons -such as a network failure or topology changes- it takes a long time to perform all the operations. For instance, to change a routing path, first new paths must be calculated, then the controller must transmit the commands to the network elements, which has to process those commands. This process can cause a delay, or even disruption, in the communication service. Therefore, this paper proposes a network control method to reduce the time to change a path using OpenFlow.

1. Introduction

1.1. Background and Overview

This paper is an extension of a previous work originally presented at the 5th International Conference on Information and Communication Technologies for Disaster Management (ICT-DM2018) [1]. Recently, the spread of cloud services has led to the diversification of communication [2], as well as the demand for frequent changes on the communication demand.. Thus, it is necessary to respond promptly to such demands, e.g., unexpected network failure. In particular, a technology that can control the network flexibly is needed.

Therefore, SDN [3,4] and OpenFlow [5,6] have recently attracted attention. SDN can control the network flexibly by software and OpenFlow is one of the most popular southbound implementation protocols of SDN. The OpenFlow architecture consists of an controller and OpenFlow-enabled switches which are centrally managed by the controller. In this OpenFlow network, when a failure or server configuration change occurs, all the paths of the traffic that is passing by through the involved switches from a source to a destination (also known as flow) must be changed.

To change the path of a flow, a new flow entry must be installed in all the switches' memory. A flow entry is represented by a condition part and an action part. The condition part includes specific values on fields within the flow header, such as MAC address or IP address. The action part includes how to route the flow when the values of its header match the values of the condition part.

However, it takes a long time to change various paths of the several flows.Specifically, it takes around 10 ms to change a single path due to the influence of the transmission time of a flow entry form the controller to the switch, and the processing time to add flow entry into the switch [7, 8, 9]. Therefore,in time-sensitive situations, such as disasters, in which several paths must be changes in a short period of time, not all the flow changes will be successful . For example, SONET (Synchronous Optical Network) needs to be recovered within 50 ms, but, it will be challenging to change more than six flows within that time [10]. This time constraint can lead to delays or even service outage. In this way, a technique to switch many flows in a short time is needed.

The main goal of this research is to reduce the delay or service outage in OpenFlow networks, with particular focus on the time needed to change the path of a flow. The proposal consists of a network control method that considers the time needed to change a path in each switch. In this method, the initial assumption is that

[*]Satoru Izumi, 2-1-1, Katahira, Aoba-ku, Sendai, 980-8577, +81-22-217-5453, izumi@ci.cc.tohoku.ac.jp

the network model consists of OpenFlow-enabled switches with different processing time, linked by connections with different bandwidth. Then, the paths with the shortest switching time and the least bandwidth requirements are selected. The basic design was described in [1], and the preliminary simulation experiments presented in [11]. In this paper, the authors evaluate the effectiveness of the proposed method using emulated environments. Based on the obtained results, we confirmed that the proposed method can reduce the switching time for all paths, so that it can realize the reduction of communication delay and network services outages.

1.2. Novelty and Contribution

In this work, a novel approach which chooses communication paths considering not only the processing time to add flow entry of the switch, but also the available bandwidth in the links. Existing approaches mainly focus on the reduction of the calculation time of the paths, or using a backup path in advance. However, they do no deal with the processing time of the switch.

By considering both the processing time and the bandwidth, it can shorten the path switching time and improve the amount of data transferred in a short period. Moreover, tt can also contribute to the reduction of the packet-loss due to the communication interruption during the network failure. This method is effective for services that require lowdelay communication. Furthermore, it is possible to improve the amount of data transfer between two sites in the situation where path change frequently, such as is the case in a disaster situation.

1.3. Paper Organization

In the following Section 2, we explain the related work on fast path switching and the target problem. In Section 3, we describe the proposed control method. The design of the algorithm is shown in Section 4. Then, Section 5 shows the evaluation of the proposed method and its effectiveness. Finally, we conclude this paper in Section 6.

2. Related Work

2.1. Related Work on Fast Path Switching

There are several authors that studied fast path switching in OpenFlow networks, which are mainly categorized in two approaches. The first approach, consists of switching to a backup path registered in advance in a proactive manner [13,14,15], while the second one reactively calculates a new path when a request occurs [10,16,17,18]. These authors select the paths considering a variety of single parameters, e.g., switching time, bandwidth of the link between the controller and the switch, and power consumption. However, all of them largely depend on the available amount of memory (TCAM) to hold the flow table. In this research, we deal with methods for calculating and switching the path reactively.

Cascone et al. [13] proposed a recovery mechanism based on fast reroute of paths in disaster situations. Mohan et al. [14] investigated algorithms to choose a backup path to decrease the number of flow entries. Stephens et al. [15] showed a mechanism to compress the flow table for fast recovery from link failure.

Astaneh et al. [10] proposed a path selection method to reduce the path switching time while considering the communication bandwidth. They reduced the cost of the path switching compared to the traditional approach of using Dijkstra's algorithm. Sharma et al. [16] proposed an in-band based path switching method for failure recovery. Paris et al. [17] showed a dynamic control scheme for network reconfiguration. Malik et al. [18] proposed a method to reduce the path calculation and switching time by partially reusing the route before the change of path. Therefore it can reduce the number of added flow entries.

2.2. Target Problem

The related work presented in the previous section did not take into account environments in which there are heterogeneous switches. Many organizations often build their network by combining switches with various specifications because of budget constraints and differences on the time of purchase [12]. Moreover, the processing time is also different in these switches, causing large delays when there is a heavy traffic on low-processing time switches. Therefore, it is necessary to create a network control method to cope with the difference in processing time when adding flow entries.

3. A Proposal of Control Method Considering Path Switching Time

3.1. Overview

In this section, to solve the problems mentioned above, the authors of this paper propose a network control method that considers the path switching time of each switch. In this method, the network model is designed with different processing time for each switch, and the proposed algorithm selects paths with the shortest time for all paths. Our proposed method considers both the processing time of the switch and the network bandwidth.

The proposed method chooses a path with enough bandwidth, and that goes through high-performance switches to add a flow entry, so that the overall path switching time is reduced when there are several flows. In addition, the proposed method also selects paths sequentially without considering the combination of paths, so that the computational time per path is reduced.

3.2. Path Selection Considering Path Switching Time

This section presents the basic concept of the path-selection mechanism that considers the path switching time as shown in the example depicted in Figure 1. In this example, there are four switches (S_1 to S_4) and the processing time to add a flow entry in each switch is different. Now, there is a communication flow from S_1 to S_4, and for some reason the path must be changed. In this case, there are two candidate paths; p_1 ($s_1 \rightarrow s_2 \rightarrow s_3$) and p_2 ($s_1 \rightarrow s_4 \rightarrow s_3$). The overall path switching time of the flow is the processing time for the switch on the path that has the maximum processing time. For instance, the path switching time of p_1 is 4.0 ms and that of p_2 is 6.0 ms. Thus, the proposed method chooses p_1 which has the shortest switching time.

3.3. Path Selection Considering Available Bandwidth

The path cost, which is used to choose the path with enough communication bandwidth, is calculated by the available bandwidth of the links on the path. Such that when the link with a larger communication bandwidth is used in the path, the path cost is smaller. Conversely, when the link bandwidth of the path is smaller, the path cost is larger. In the example shown in Figure 2, the path cost of the upper path is smaller because the bandwidth of the links on that path are larger than the ones in the lower path, and therefore the former is selected.

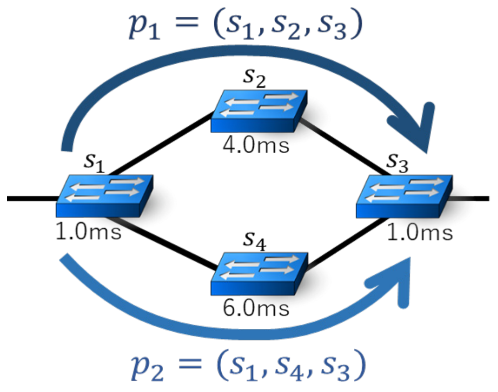

Figure 1: Example of Path Selection Considering Path Switching Time

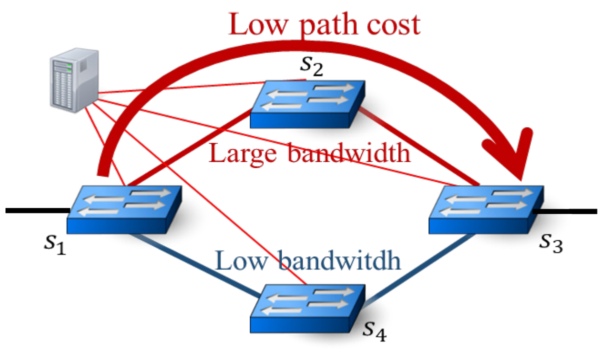

Figure 2: Example of Path Selection Considering Available Bandwidth

4. Design

4.1. Network Model

The network model is defined as below.

- $s_i \in S \ (i = 1, 2, ..)$: Switch
- t_i : Processing time for adding one flow entry into s_i
- $e_{i,j} \in E$: Link between s_i and s_j
- $B_{i,j}$: Available bandwidth of $e_{i,j}$
- p_k : Path (list of switches)
- $c_{i,j}$: Link cost of $e_{i,j}$ (= 1Gbps/$B_{i,j}$)

4.2. Definition of Path Switching Time

The path switching time is defined as:

"The time from the arrival of the first additional flow entry from the controller to the switch until the completion of the path switching on all switches"

Figure 3 shows an example. Suppose that tthere are three flows (flow 1, 2, and 3) assigned to the paths shown in Figure 3. In this case, t_1 is 1 ms, and s_1 needs to switch the three flows. Thus, the time to complete the processing of the additional flow entry is 1 ms * 3 = 3 ms. Then, t_2 is 4 ms and the s_2 needs to switch two flows (flow 1 and 2). Therefore, the time to complete the processing of the additional flow entries is 8 ms (4 ms * 2). For the other switches, the time is calculated in the same way. Finally, the path switching time in this example is 8 ms, which is the longest time to complete the process.

The path switching time $T(P_m)$ for a set of paths (P_m) is defined as in (1)

$$T(P_m) = \max_{\{i|s_i \in S\}} \{t_i \times \textstyle\sum_{p_k \in P_m} z_i(p_k)\} \tag{1}$$

Here, $z_i(p_k)$ means whether the list of p_k includes s_i.

$$z_i(p_k) = \begin{cases} 1 \ (s_i \in p_k) \\ 0 \ (otherwise) \end{cases} \tag{2}$$

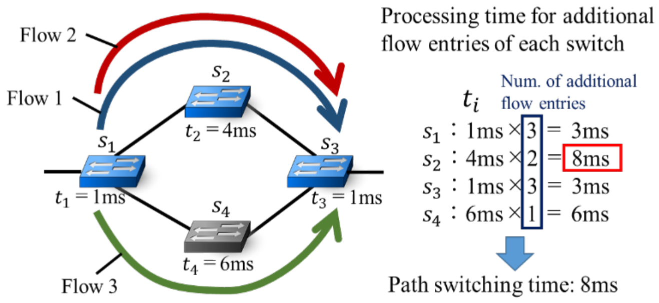

Figure 3: Example of Path Switching Time

4.3. Link Cost

The proposed method prioritizes the path switching time rather than the path cost, and chooses paths sequentially. The overview of the sequential path selection method is as follows:

- Choose a path for one flow at a time.
- Substract the required bandwidth of the flow from the link bandwidth on the path, and update the link cost every time the path is selected.
- Assign a flow to the path until there is no available bandwidth left for the flow.

Figure 4 shows an example of the link cost update. If the link bandwidth is 1 Gbps, the link cost is 1. Then, when a flow requires 300 Mbps of bandwidth, it is inserted into the link. The available bandwidth of the link becomes 700 Mbps, and the link cost increases to about 1.4.

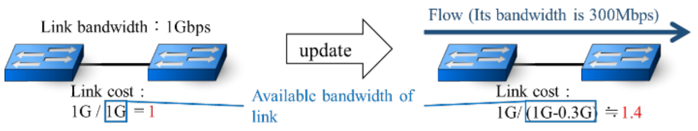
Link bandwidth : 1Gbps
Link cost : 1G / 1G = 1
update
Available bandwidth of link
Flow (Its bandwidth is 300Mbps)
Link cost : 1G/ (1G-0.3G) ≒ 1.4

Figure 4: Example of Link Cost Update

4.4. Flow of Path Selection

This section explains the process of path selection. In order to consider both the processing time of the switch and the path cost, the proposed method chooses paths as detailed below.

1. Compute the paths with the smallest increase in the path switching time.
2. Choose the path with the lowest path cost form the computed paths.
3. Repeat process 1 and 2 for the number of flows that needs to change.

5. Experimentation

5.1. Overview

The proposed method was evaluated in an emulated environment. The network used in the experiment is shown in Figure 5. Here, the available bandwidth $B_{i,j}$ of all links is set as 1 Gbps. Also, the processing time of the source switch, and destination switch was set to 1 ms; while remaining switches were randomly set from 1 to 10 ms.

Several flows with different required bandwidth were inserted, and randomly shutdown links. After the link is down the paths of the flows are switched. To compare the effectiveness of the proposal, the obtained throughput is obtained and compared with an existing method, which selects paths considering only path cost.

In this experiment, Mininet (ver. 2.2.2) [19] was used as a network emulator on a single computer, whose specification is as below:

- CPU: Intel (R) Xeon (R) CPU E5-2650 v4 @ 2.20 GHz) x 8 cores
- Memory: 16 GB

We also used OpenDaylight (0.3.3-Lithium-SR3) [20] as the OpenFlow controller, and OpenVSwitch (ver. 2.5.5) [21] as the OpenFlow switch.

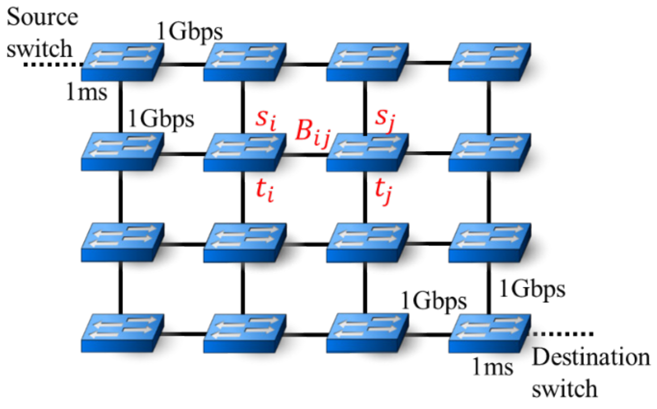

Figure 5: Network Topology Used in the Experiment

5.2. Results

The authors categorize the throughput comparing the proposed method with the existing method into the following three patterns:

- The proposed method always dominates (Win).
- The proposed method dominates until a certain time (Win[time]).
- The proposed method is equal to the existing method (Eq).

Examples of graphs of the three patterns are shown in Figures 6, 7, and 8. The comparison of the data transfer amount between the proposed method and the existing method is shown in Table 1. From the table, it is observed that when the total bandwidth of the flows is 1,800 Mbps or more, and the number of disconnected links is large, the data transfer amount of the proposed method is always better. However, when the total bandwidth of the flows is 1,800 Mbps or more, and the number of link disconnections is small, then the amount of transferred data of the proposed method is larger within 0.1 to 0.2 seconds.

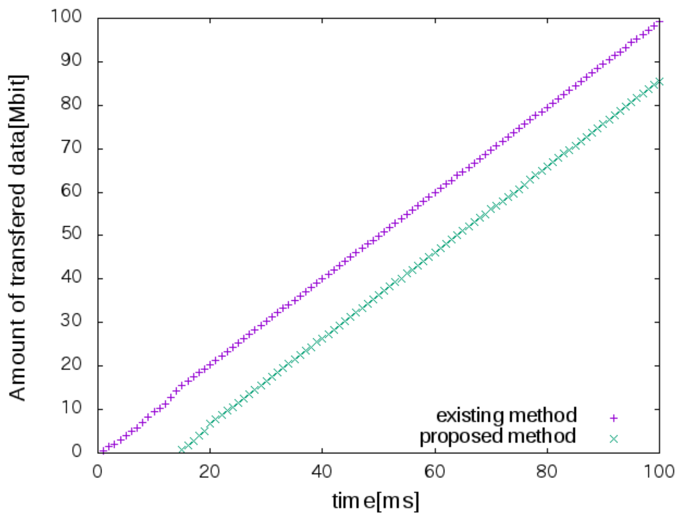

Figure 6: The Proposed Method Dominates Always (Win)

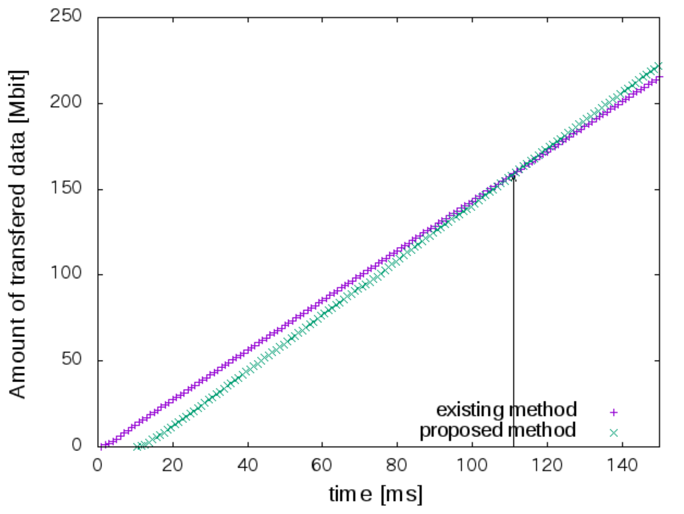

Figure 7: The Proposed Method Dominates Until A Certain Time (Win[112s])

Moreover, when the total bandwidth of flows is less than 1,400Mbps, and the amount of transferred data of the proposed method is larger is within 1.5 seconds, the proposed method is larger regardless of the number of disconnected links. Finally, when the total bandwidth of flows is less than 1,200 Mbps, the proposed method is always better.

Table 1: Comparison of the data transfer amount between the proposed method and the existing method

		Total bandwidth of the flows [Mbps]										
		1,000	1,200	1,400	1,600	1,800	2,000	2,200	2,400	2,600	2,800	3,000
Number of down links	0	Win	Win	Win [1.5ms]	Win [0.3ms]	Win [0.1ms]	Win [0.1ms]	Win [0.1ms]	Win [0.1ms]	Win [0.1ms]	Win [0.1ms]	Win [0.1ms]
	4	Win	Win	Win [1.5ms]	Win [0.3ms]	Win [0.2ms]	Win [0.2ms]	Win [0.2ms]	Win [0.1ms]	Win	Win	Win
	8	Win	Win	Win	Win [0.2ms]	Win [0.1ms]	Win	Win	Win [0.1ms]	Win	Win	Win
	12	Win	Win	Win	Win [0.1ms]	Win	Win	Win	Win	Win	Win	Win
	16	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Eq
	18	Eq	Eq	Eq	Eq	Eq	Eq	Eq	Eq	Eq	Eq	Eq

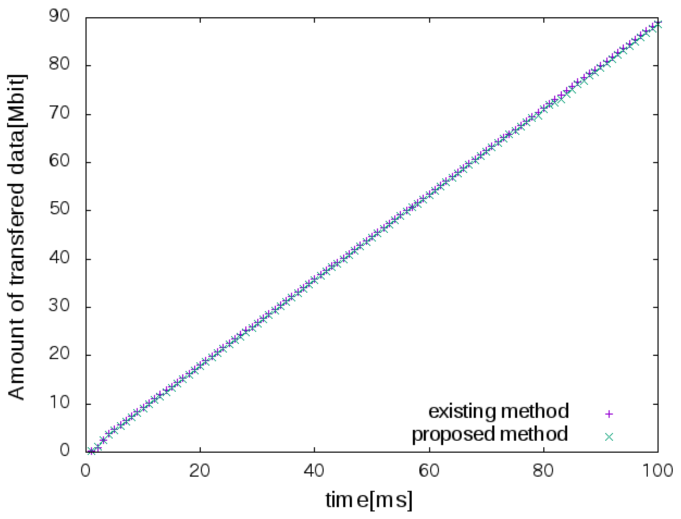

Figure 8: The Proposed Method Is Equal To The Existing Method (Eq)

5.3. Discussion

From the experimental results in the previous sections, the authors of this paper confirm that the proposed method is more effective than the existing method, since the amount of transferred data is equal or larger to the existing method while the path switching time is shortened. Morever, the path switching process is more effective than the existing method when these events occurs frequently and there is not enough bandwidth.

6. Conclusion

This paper presents a network control method considering the path switching time in SDN. We designed the path selection method and evaluated with an emulated experimentation. From the experimental results, it is confirmed that the proposed method is effective in situation where changes occurs frequently in a short time due to link disconnection and restoration (e.g., disasters).

As future work, we will extend the proposed method by introducing other parameters and experiment with real networks. Currently, we focused on the time from the arrival of the first additional flow entry from the controller to the switch until the completion of the path switching on all switches. Thus, we will evaluate the total time of changing paths from the moment some events occur until they complete changing the paths in comparison with related work.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] K. Gotani, H. Takahira, M. Hata, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "OpenFlow Based Information Flow Control Considering Route Switching Cost," Proc. the 5th International Conference on Informationand Communication Technologies for Disaster Management (ICT-DM 2018), pp.1-4, 2018.

[2] Ministry of Internal Affairs and Communications, ahttp://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nd252140.html (Accessed 2019).

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On Scalability of Software-Defined Networking," IEEE Communication Magazine, Vol. 51, No. 2, pp.136-141, 2013.

[4] "Software-Defined Networking (SDN) definition," Open Networking Foundation, https://www.opennetworking.org/sdn-resources/sdndefinition, (Accessed 2019).

[5] N. McKeown, T. Anderson, H. Balakrishna, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, Vol. 38, Iss. 2, pp.69-74, 2008.

[6] Open Networking Foundation, "OpenFlow Switch, Specification (Version 1.5.1)," https://www.opennetworking.org/wp-content/uploads/2014/10/openow-switch-v1.5.1.pdf (Accessed 2019).

[7] A. Nguyen-Ngoc, S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, and M. Jarschel, "Performance Evaluation Mechanisms for FlowMod Message Processing in OpenFlow Switches," Proc. of IEEE Sixth International Conference on Communications and Electronics (ICCE2016), pp.40-45, 2016.

[8] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. Moore, "OFLOPS: An open framework for openflow switch evaluation," Proc. of Passive and Active Measurement (PAM2012), LNCS 7192, pp.85-95, 2012.

[9] R. Bifulco and A. Matsiuk, "Towards Scalable SDN Switches: Enabling Faster Flow Table Entries Installation," Proc. of 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015), pp.343-344, 2015.

[10] S. A. Astaneh and S. Shah Heydari, "Optimization of SDN Flow Operations in Multi-Failure Restoration Scenarios," IEEE Transactions on Network and Service Management, Vol.13, No.3, pp.421-432, 2016.

[11] K. Gotani, H. Takahira, M. Hata, L. Guillen, S. Izumi, and T. Abe, "OpenFlow Based Information Flow Control Considering Route Switching Cost," Proc. the 2nd IEEE International Workshop on Information Flow Oriented Approaches in Internet of Things and Cyber-Physical Systems (InfoFlow 2019), pp.527-530, 2019.

[12] A. Feldmann, P. Heyder, M. Kreutzer, S. Schmid, J.P. Seifert, H. Shulman, K. Thimmaraju, and M. Waidner, "NetCo: Reliable Routing With Unreliable Routers," Proc. of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), pp.128-135, 2016.

[13] C. Cascone, L. Pollini, D. Sanvito, A. Capone, and B. Sanso, "SPIDER: Fault resilient SDN pipeline with recovery delay guarantees," Proc. of IEEE NetSoft Conference Workshops (NetSoft), pp. 296-302, 2016.
[14] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "TCAM-Aware Local Rerouting for Fast and Efficient Failure Recovery in Software Defined Networks," Proc. of 2015 IEEE Global Communications Conference (GLOBECOM 2015), pp.1-6, 2015.
[15] B. Stephens, A. L. Cox, and S. Rixner, "Scalable Multi-Failure Fast Failover via Forwarding Table Compression," Proc. ACM SIGCOMM Symposium on SDN Research Article (SOSR 2016), no. 9, pp.1-12, 2016.
[16] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "In-Band Control Queuing and Failure Recovery Functionalities for OpenFlow," IEEE Network, vol. 30, no. 1, pp. 106-112, 2016.
[17] S. Paris, G. S. Paschos, and J. Leguay, "Dynamic Control for Failure Recovery and Flow Reconfiguration in SDN," Proc. International Conference on the Design of Reliable Communication Networks (DRCN 2016), pp.152-159, 2016.
[18] A. Malik, B. Aziz, M. Adda, and C. Ke, "Optimization Methods for Fast Restoration of Software-Defined Networks," IEEE Access, Vol.5, pp.16111-16123, 2017.
[19] Mininet, http://mininet.org/
[20] OpenDaylight, https://www.opendaylight.org/
[21] Open vSwitch, http://openvswitch.org/