A Two-Stage Failover Mechanism for High Availability in Service Function Chaining

Hosung Baek[1], Haneul Ko[1], Gwangwoo Park[1], Sangheon Pack[1], Jiyoung Kwak[2]

[1] School of Electrical Engineering, Korea University, Korea
[2] Programmable Infra Research Section, ETRI, Korea
{gh1emd, st_basket, kwoo17, shpack}@korea.ac.kr, jiyoung@etri.re.kr

Abstract

Service function chaining (SFC) enables specific network flows to be processed through an ordered set of service functions. In this paper, we propose a two-stage failover mechanism to provide high availability (HA) in SFC. In the proposed mechanism, a secondary service function path (SFP) is pre-computed, which is maximally disjointed with the primary SFP, and a backup service function instance (SFI) is prepared for each service function. For a failure event, the first local repair stage is examined by means of the preconfigured backup SFI. If the backup SFI is not available and/or any link is failed, the second end-to-end repair stage is triggered by replacing the primary SFP with the secondary SFP. Simulation results demonstrate that the proposed failover mechanism can reduce the recovery time by 50%-75% compared to the conventional approach.

Keywords: Service function chaining (SFC), service function (SF), service node (SN), backup, high availability (HA)

1 Introduction

Incoming flows to a network should be processed by a set of service functions (SFs) or middleboxes (e.g., firewall (FW), network address translator (NAT), and deep packet inspection (DPI)) to meet service requirements and operator's policies. Service function chaining (SFC) defines an ordered set of SFs for a specific flow and steers the flow to be processed by them [1]. With advances of software-defined networking (SDN), SDN-based SFC has emerged and widely discussed in the literature because it can provide more flexible SFC management. For example, Qazi et al. [2] introduced a middlebox policy enforcement layer for SDN-based SFC, which translates policy rules posed by administrators and solves load balancing and loop avoidance issues. Meanwhile, Fayazbakhsh et al. [3] devised a technique to add a tag to the packet header to control SFC and address SF placement issues.

One of key requirements for SDN-based SFC is high availability (HA) at the control and data planes. Unfortunately, most previous studies have only focused on HA of the controllers [4-5] instead of the SFC data plane. Even though the SDN controller is the key element in SDN-based SFC, different kinds of failures can occur at the data plane (e.g., link failure between two SF instances or SF instance failure). Therefore, a failover mechanism for the data plane should be also devised for HA in SDN-based SFC. In addition, previous failover mechanisms are based on reactive operation, i.e., the SDN controller re-finds available service function path (SFP) or SF for a failure event. Such reactive operation may lead to the increased failover time, which will degrade the overall performance.

To address these drawbacks, we propose a two-stage failover mechanism, which consists of the local repair and end-to-end repair stages. In the proposed mechanism, a secondary SFP is pre-computed, which is maximally disjointed with the primary one. At the same time, backup service function instances (SFIs) are prepared for individual SFIs, which can guarantee minimal failover time. At the local repair stage, a failure at SFI is first detected by means of a bidirectional forwarding detection (BFD) protocol [6]. If a failure is detected, the flow is locally repaired by redirecting the flow to the backup SFI. Even with the local repair stage, a backup SFI failure or link failure between SFIs cannot be handled. For such cases, the end-to-end repair stage is triggered by replacing the primary SFP with the secondary SFP. For performance evaluation, we conducted extensive simulations in terms of failure recovery time under SFI and link failures. Simulation results show the proposed mechanism can reduce the recovery time by 50%-75% compared with the conventional mechanism.

The contribution of this paper is two-fold: 1) even though failover mechanisms are widely discussed for SDN controllers, little work on HA in SFC has been reported in the literature. The proposed two-stage failover mechanism is a tailored one for HA in SFC,

*Corresponding Author: Sangheon Pack; E-mail: shpack@korea.ac.kr
DOI: 10.3966/160792642018011901022

and 2) the two-stage failover mechanism can deal with link failures as well as SFI failures, and can reduce the recovery time significantly by choosing the most appropriate secondary SFP and backup SFI based on the optimization framework. In particular, the optimization framework considers a salient feature for stateful SFs, i.e., synchronization overhead between SFIs and backup SFIs.

The remainder of this paper is organized as follows. Section 2 surveys the related works. Section 3 and Section 4 describe the system model and the two-stage failover mechanism, respectively. Section 5 presents simulation results and compares the results of the proposed mechanism with the conventional mechanism. Section 6 concludes this paper.

2 Related Work

To guarantee the HA of SDN, a number of studies have been reported and most of them focus on the HA in the control plane. Miller *et al.* [4] investigated an optimal controller placement such that connectivity is maximized for HA. To this end, Miller et al. chooses the positions of controllers that provide the highest number of disjoint paths between switches and controllers. Botelho *et al.* [5], proposed a fault-tolerant controller architecture with a data store based on the replicated state machine of the control plane and a lease management algorithm selecting a master controller for fault-tolerant SDN. These studies only consider the HA for the control plane, not the data plane. However, in SDN networks, a failure event can occur in the data plane including network link, SF, and SFF. Regarding the data plane, only a few works have been recently introduced. Desai *et al.* [7] proposed an algorithm that utilizes a detection method to check the link failures among neighboring switches and notifies the link failures to all the neighboring switches in order to refrain from sending messages to the failed link. Kempf *et al.* [8] proposed to implement a monitoring function on OpenFlow switches without posing a processing load on the controller. However, [7-8] consist only of detecting link failures without restoration after link failures. On the other hand, Staessens *et al.* [9] considered data plane resiliency both in terms of restoration and protection. For restoration, after the controller gets a link failure notification, a list is made of all affected paths. Afterwards, a restoration path is calculated using a shortest path algorithm on the remaining topology. Since switches which are on both the working and the restoration path are affected, the flow entry is modified. For protection, a backup path is precomputed and established with the original path. However, in [7-9], these approaches, despite their ability to restore link failures, have several drawbacks. For instance, the type of SF and the failure of SFI are not considered in [7-9]. Therefore, the proposed algorithms in [7-9] cannot be directly applied to the SFC data plane, a new mechanism to restore both link failures and SFI failures needs to be devised.

3 System model

In this section, we describe the underlying network model for the two-stage failover mechanism and relevant protocols (i.e., BFD and network service header (NSH)).

3.1 Network Model

Figure 1 shows the network model based on the SFC architecture [10] consisting of a SDN controller, ingress/egress nodes, service function forwarders (SFFs), and service nodes (SNs). The SDN controller has global knowledge on the network and computes SFPs for given conditions, and the controller is placed D hops away from the ingress node. The ingress node in the SFC domain classifies incoming flows based on the pre-defined network policy. Meanwhile, the egress node forwards the processed packets to the outside of the SFC domain. SN denotes a physical or virtual element that has the constraints on resources and hosts one or more SFIs by means of NFV [11]. Note that SN can be connected to one or more SFFs. Lastly, SFFs are responsible for forwarding flows to one or more connected SNs and handling packets coming back from SNs. It is assumed that the link between aforementioned nodes supports bidirectional communications.

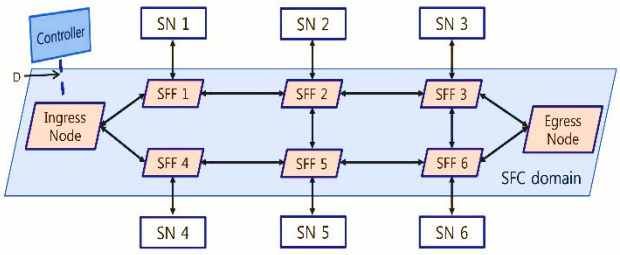

Figure 1. Network model

The network model can be represented by $G=(V,E)$ that consists of vertices (i.e., nodes) and edges. The set of nodes, V, consists of the sets of SNs (denoted by V_{SN}), SFFs (denoted by V_{SFF}), ingress nodes (denoted by V_{ing}), and egress nodes (denoted by V_{eg}). On the other hand, (i, j) represents the link between node i and node j, which belongs to the set of links, E. $L_{i,j}$ denotes a binary variable to represent whether or not node i and node j are inter-connected, and $C_{i,j}$ represents the bandwidth of the link (i, j).

3.2 BFD and NSH Protocols

In this paper, we use BFD protocol as a failure detection method and exploit NSH to contain service

Table 1. *Summary of notations*

Parameter	Description
V	Set of nodes
E	Set of links
$C_{i,j}$	Capacity of link (i, j)
$Q_{i,h,t}$	Binary variable denotes whether or not SF h in SN i is type t
$f_{m,t}$	Binary variable denotes whether or not the m-th SF of SFC is type t
$w_{m,i,h}$	Binary variable denotes whether or not SF h in SN i is the m-th backup SFI
$x_{m,i,j}$	Binary variable denotes whether or not node i forwards packet to node j for processing the m-th SF of the primary SFC
$y_{m,i,j}$	Binary variable denotes whether or not node i forwards packet to node j for processing the m-th SF of the secondary SFC
$y^{IN}_{m,i,j}$	Binary variable denotes whether or not node i forwards packet to node j for processing packets at the backup SF of the m-th SF
$y^{OUT}_{m,i,j}$	Binary variable denotes whether or not node i forwards packet to node j to deliver packets from the backup SF of the m-th SF to the (m+1)-st SF
$z_{m,i,j}$	Binary variable denotes whether or not node i forwards packet to node j for synchronization between the m-th SF and backup SF
N_m	The number of SFs in SFC
N_H	The number of SF instances in SN
N_T	The number of SF types in SFC
m	The order of SF in SFC
b_m	Link bandwidth needed for flow transmission
B_m	Link bandwidth needed for synchronization between the m-th SF of SFC and the m-th backup SF (or between the m-th SF of primary SFP and the m-th SF of secondary SFP)

path information. BFD [6] is a detection protocol designed to provide fast forwarding path failure detection in routing protocols, which is independent of media types and topologies. The procedure of BFD is as follows. BFD implements control/echo messages to detect liveness of links or paths between end-points. First, a BFD session is established by a session request between two nodes and a failure detection period is configured. After that, BFD peers send periodic control packets to detect liveness. A node receiving a control message, replies with an echo message containing its respective session status. If no reply message is received, a failure is considered and the BFD session is down. In this manner, BFD achieves faster failure detection and restoration. Throughout this paper, the use of BFD is assumed to detect link and SFI failures.

NSH [12] is a protocol to contain metadata and service path information in SFC. By using NSH, different next hops for each SFF can be established. That is, different next hops for primary SFP, secondary SFP, and backup SFI can be designated by assigning different weights, which facilitates the implementation of the two-stage failover mechanism.

4 Two-stage Failover Mechanism

In this section, we describe the two-stage failover mechanism in detail. In the proposed mechanism, backup SFIs are prepared for individual SFIs, which can guarantee minimal failover time. At the same time, a secondary SFP is pre-computed, which is maximally disjointed with the primary one. We first formulate an integer linear programming (ILP) problem to configure the backup SFIs and secondary SFP. After that, we explain the local repair stage and the end-to-end repair stage.

4.1 Configuration of Secondary SFP and Backup SFI

To trigger the local repair and end-to-end repair stages, the backup SFIs and secondary SFP should be pre-configured. To this end, the following ILP problem is formulated and important notations for the ILP problem are summarized in Table 1. Note that the formulated ILP problem can be solved by a well-known ILP solvers such as IBM CPLEX and GLPK.

When a backup SFI is selected and used, we need to change SFP after restoring by the backup SFI. Therefore, backup SFI selection has to consider the performance of restored SFP. In other words, by minimizing the increase of the SFP length after the restoration, the impact of the restoration can be mitigated. To obtain the backup SFIs considering the increased length after restoration, $y^{IN}_{m,i,j}$ and $y^{OUT}_{m,i,j}$ are defined. $y^{IN}_{m,i,j}$ and $y^{OUT}_{m,i,j}$ are binary variables that represent whether or not node i forwards packets to node j for processing the packets at the m-th backup

SFI and to represent whether or not node *I* forwards packets to node *j* to deliver packets to the *(m+1)*-st SFI, respectively. Then, the length of routing path from the *(m-1)*-st SFI to the *(m+1)*-st SFI after replacing with the *m*-th backup SFI can be calculated by $\sum_{i\in V}\sum_{j\in V}(y^{IN}_{m,i,j}+y^{OUT}_{m,i,j})$. Meanwhile, the length of original routing path from the *(m-1)*-st SFI to the *(m+1)*-st SFI in the primary SFP can be represented as $\sum_{i\in V}\sum_{j\in V}(x_{m,i,j}+x_{m+1,i,j})$. Consequently, the increased length after replacing with the *m*-th backup SFI is denoted by $\sum_{i\in V}\sum_{j\in V}\{(x^{IN}_{m,i,j}+y^{OUT}_{m,i,j})-(x_{m,i,j}+x_{m+1,i,j})\}$.

Figure 2 shows an example when SFC is specified as FW-NAT-DPI and the primary SFP is ingress node-SFF1-SN1-SFF1-SFF2-SN2-SFF2-SFF3-SN3-SFF3-egress node. If FW in SN4 is selected as a backup SFI, the replaced SFP will be ingress node-SFF4-SN4-SFF4-SFF5-SFF2-SN2-SFF2-SFF3-SN3-SFF3-egress node and the increased length becomes 1. Since our goal is to minimize the increase of the SFP length after restoration, the objective function for constructing the backup SFI can be defined as

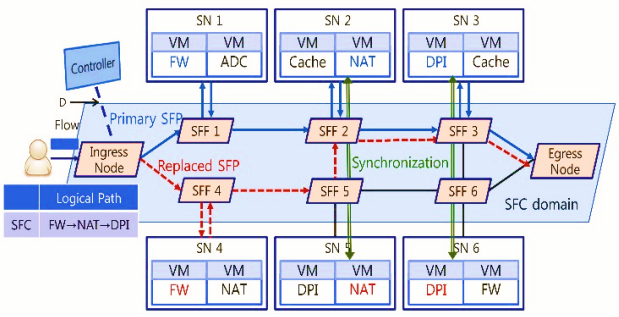

Figure 2. Backup SFI example

$$\min \sum_{i\in V}\sum_{j\in V}\{(y^{IN}_{m,i,j}+y^{OUT}_{m,i,j})-(x_{m,i,j}+x_{m+1,i,j})\} \quad \textbf{(1)}$$

On the one hand, if backup SFI or link between SFIs fails, restoration with backup SFI is not available. Therefore, the end-to-end restoration with secondary SFP has to be triggered for such cases. To maximize the efficiency of the secondary SFP, the effect of failure in the primary SFP on the secondary SFP should be minimized. To achieve this goal, the number of links that are shared between the primary and secondary SFP should be minimized. The number of the shared links can be represented by $\sum_{i\in V}\sum_{j\in V}\sum_{m_1=1}^{N_m+1}\sum_{m_2=1}^{N_m+1}\{(x_{m_1,i,j}+x_{m_1,j,i})y_{m_2,i,j}\}$, where $x_{m,i,j}$ and $y_{m,i,j}$ are binary variables to represent whether or not node i forwards packets to node j for processing at the *m*-th SFI in the primary SFP and secondary SFP, respectively. In this case, since the primary SFP is given in advance, the value of $x_{m,i,j}$ is known. N_m is the number of SFs in SFC, and *m* is the order of SF in SFC. Note that since a link supports bidirectional communications, both $x_{m_1,i,j}$ and $x_{m_1,j,u}$ have to be considered to count the number of joint links between primary and secondary SFP. Consequently, the objective function for constructing the secondary SFP can be represented by

$$\min \sum_{i\in V}\sum_{j\in V}\sum_{m_1=1}^{N_m+1}\sum_{m_2=1}^{N_m+1}\{(x_{m_1,i,j}+x_{m_1,j,i})y_{m_2,i,j}\}. \quad \textbf{(2)}$$

To solve (1) and (2), we have the following constraints. First of all, the traffic volume for state synchronization between SFIs and the backup SFIs (see double lines in Figure 2) should be carefully managed for stateful SFs (e.g., DPI and NAT). Specifically, the total traffic volume for synchronization should be smaller than or equal to a pre-defined threshold, δ, and thus we have

$$\sum_{i\in V}\sum_{j\in V}\sum_{m=1}^{N_m} z_{m,i,j}B_m \le \delta, \quad \textbf{(3)}$$

where $z_{m,i,j}$ is a binary variable to represent whether or not node i forwards packets to node j for synchronization between the *m*-th SFI and the backup SFI, and B_m is the link bandwidth needed for synchronization.

On the one hand, flow constraints of secondary SFP can be described as follows.

$$\sum_{j\in V} y_{m,j,i}-\sum_{k\in V} y_{m,i,k}=0,\ \forall i\in V_{SFF}, \quad \textbf{(4)}$$

$$\sum_{j\in V} y_{1,j,i}-\sum_{k\in V} y_{1,i,k}=-1,\ \forall i\in V_{ing}, \quad \textbf{(5)}$$

$$\sum_{j\in V} y_{N_m+1,j,i}-\sum_{k\in V} y_{N_m+1,i,k}=1,\ \forall i\in V_{eg}, \quad \textbf{(6)}$$

$$y_{m,i,j}=y_{m+1,j,i},\ \forall i\in V_{SFF},\ \forall m\in j\in V_{SN},\ \forall m\in\{1,\ldots,N_m\} \quad \textbf{(7)}$$

(4) means that the traffic that comes into any SFF should go out to another SFF. Meanwhile, (5) indicates that the ingress node delivers traffic to other nodes, and (6) represents that the flow is terminated when the traffic is delivered to the egress node. Finally, (7) represents that the traffic delivered from an SFF to an SN should be transmitted to the SFF after processing at the SF.

On the other hand, flow constraints of backup SFI can be described as follows.

$$\sum_{j\in V} y^{IN}_{m,j,i}-\sum_{k\in V} y^{IN}_{m,i,k}=0,\ \forall i\in V_{SFF}, \quad \textbf{(8)}$$

$$\sum_{j\in V} y^{OUT}_{m,j,i}-\sum_{k\in V} y^{OUT}_{m,i,k}=0,\ \forall i\in V_{SFF}, \quad \textbf{(9)}$$

$$\sum_{j\in V} y^{IN}_{1,j,i}-\sum_{k\in V} y^{IN}_{1,i,k}=0,\ \forall i\in V_{ing}, \quad \textbf{(10)}$$

$$\sum_{j\in V} y^{OUT}_{N_m+1,j,i} - \sum_{k\in V} y^{OUT}_{N_m+1,i,k} = 0, \forall i \in V_{eg}, \quad (11)$$

$$y^{IN}_{m,i,j} = y^{IN}_{m,j,i}, y^{OUT}_{m,j,i} = y^{OUT}_{m,i,j}, \quad (12)$$
$$\forall i \in V_{SN}, \forall i \in V_{SN}, \forall m \in \{1, ..., N_m\}.$$

As a capacity constraint, the total bandwidth used to transmit traffic should be equal to or smaller than the link capacity, $C_{i,j}$. Therefore, we have the following capacity constraint of secondary SFP:

$$\sum_{m=1}^{N_m} y_{m,i,j} b_m + y_{m,j,i} b_m \le C_{i,j}, \forall i, j \in V, \quad (13)$$

where $\sum_{m=1}^{N_m} y_{m,i,j} b_m + y_{m,j,i} b_m$ represents the total bandwidth to transmit incoming traffic, and b_m is the link bandwidth needed for flow transmission. Likewise, the capacity constraint of backup SFI is as follows.

$$\sum_{m=1}^{N_m} y^{IN}_{m,i,j} b_m + y^{IN}_{m,j,i} b_m + y^{OUT}_{m,i,j} b_m + \quad (14)$$
$$y^{OUT}_{m,j,i} b_m \le C_{i,j}, \forall i, j \in V,$$

Additionally, in selecting the secondary SFP, all SFs of SFC should be processed and the corresponding constraint is given by

$$\sum_{i\in V} \sum_{h=1}^{N_H} \sum_{t=1}^{N_T} Q_{i,h,t} w_{m,i,h} f_{m,t} = 1. \quad (15)$$

$Q_{i,h,t}$ is a binary variable denotes whether or not SFI h in SN i is type t, $w_{m,i,h}$ is a binary variable denotes whether or not SFI h in SN i is the m-th backup SFI and $f_{m,t}$ is a binary variable denotes whether or not the m-th SF of SFC is type t.

4.2 Local Repair and End-To-End Repair Stages

Figure 3 shows the overall procedure of the two-stage failover mechanism. When a new flow is generated, a primary SFP for a given SFC is constructed. Note that the primary SFP can be configured by different methods (e.g., [13]), which is beyond scope of this paper. After that, the SFIs and links for the primary SFP are used as input parameters into the ILP tool. That is, a backup SFI is prepared for each SFI by means of the ILP formulation. In this case, it is assumed that the controller maintains a global view on the link capacity and the SFI type. At the same time, a secondary SFP is configured based on the ILP formulation to substitute the primary SFP for link or backup SFI failures. Only when all of these configurations are set, the flow is serviced accordingly. Therefore, if the backup SFIs and secondary SFP are preconfigured, the flow can be serviced with the minimal service interruption time in case of a link failure or a SFI failure.

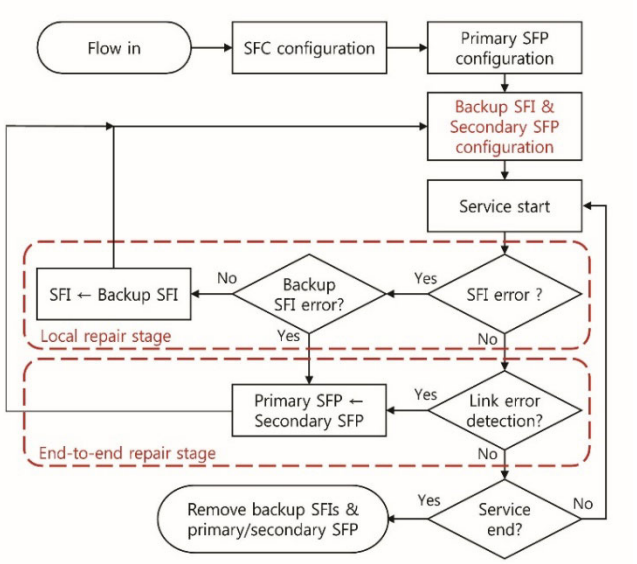

Figure 3. Flow chart

During the service time, if a SFI fails, the failure is detected by means of BFD. In this case, the local repair stage is initiated independent of the SDN controller. First of all, the failed SFI is replaced with the backup SFI without replacing whole SFP, and the previous SFI of the failed SFI forwards the flow to the replaced backup SFI. Therefore, the flow can be serviced continuously. Even with the local repair stage, a link failure between SFIs cannot be handled. Also, if SFI and backup SFI fail together at the same time, the local repair is not feasible. Therefore, for such cases, the end-to-end repair stage is required by substituting the primary SFP with the preconfigured secondary SFP. Specifically, when a failure is notified to the ingress node, the ingress node reconfigures its forwarding rule on the flow towards the secondary SFP and thus the flow can be serviced with minimal interruption. Additionally, once the controller is informed of the failure of SFI or link, it can reconfigure new backup SFI and secondary SFP without traffic interruption.

5 Simulation Results

For performance evaluation, we compare the two-stage failover mechanism with the conventional mechanism [14], where the restoration for the link or SF failure is conducted by the controller. Specifically, when a link or SF failure is detected, the failure notification message is sent to the controller. Then, the controller establishes a new SFP and sends the flow table update message to switches. Finally, flow tables of switches are updated to deploy the new SFP and the flow can be serviced again. Simulations are conducted on the well-known network topology (i.e., Abilene network of Internet2) [15]. The nodes in Abilene network are mapped to SFFs, ingress/egress nodes, and six SNs are connected to some SFFs. The total link capacity is uniformly set as 10 Gbps for all links. In

this simulation, the CPU capacity is only considered as a resource type, and all SNs can accommodate at most two SFIs. The given SFC is FW-NAT-DPI and the flow demand of SFC is set to 3 Gbps. On the other hand, the default time to compute a new SFP in the conventional mechanism is assumed as 50 ms, and the propagation time varies from 10 ms to 30 ms. The required link bandwidths (in Gbps) for synchronization of NAT and DPI are 10% and 5% of the total link capacity respectively, and δ is set to 30. The detection of SFI and link failures in both mechanisms is done by the BFD protocol.

5.1 Effect of Propagation Time

Figure 4 shows the recovery time for SFI and link failures as a function of the propagation time. Note that since the procedures of the conventional failover mechanism to deal with SFI and link failures are same, the recovery time for two cases is equal. It can be seen that the recovery times of the proposed and conventional mechanisms increase as the propagation time increases. This is because the time to transmit messages for restoring SFI and link failures increases with the increase of the propagation time.

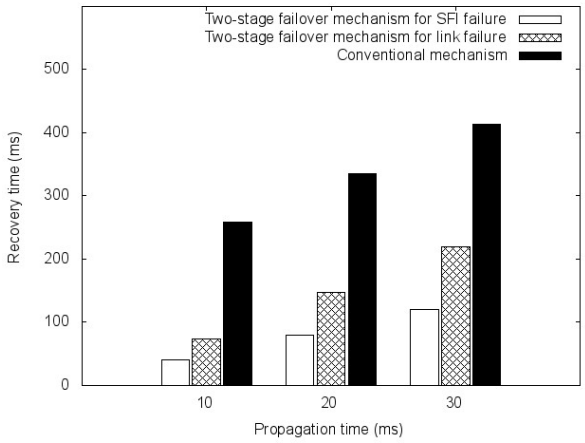

Figure 4. Effect of propagation time

On the other hand, it can be also seen that the recovery time of the proposed mechanism is lower than that of the conventional mechanism. This can be explained as follows. In the conventional mechanism, when a SFI failure occurs,

it is reported to the controller and the controller establishes a new SFP and sends flow table update messages to switches. As a result, longer recovery time is expected and it is highly affected by the propagation time. Meanwhile, in the proposed mechanism, the failed SFI is replaced with the backup SFI right after a SFI failure is detected. That is, the recovery time for the SFI failure in the proposed mechanism only includes the SFI failure detection time by BFD. In addition, in case of a link failure, the ingress node redirects the flow to the secondary SFP without any intervention of the controller. Therefore, the proposed mechanism shows the reduced recovery time for both link and SFI failures.

5.2 Effect of D

Figure 5 shows the recovery time for SFI and link failures when the distance between the controller and ingress node, D, is changed. Note that the controller can be reached only through the ingress node in our system model. As shown in Figure 5, the recovery time of the conventional mechanism increases linearly to the increase of D because SFI or link failures should be reported to the controller and the controller has to send flow table update messages to switches. On the contrary, in the proposed mechanism, the recovery time is not affected by D since the pre-configured backup SFI or secondary SFP is exploited without reporting to the controller. In short, the gain of the proposed mechanism in terms of the recovery time becomes apparent as D increases.

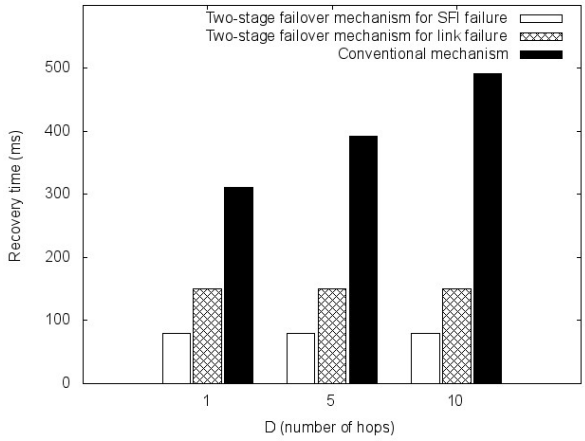

Figure 5. Effect of D

6 Conclusion

In this paper, we proposed a two-stage failover mechanism to provide HA in SDN-based in SFC. By means of pre-established secondary SFP and backup SFIs, the proposed two-stage failover mechanism guarantees shorter failover time, which is verified by extensive simulations. In our future work, we will implement the two-stage failover mechanism in open source (e.g., Open Daylight (ODL))-based testbeds and carry out extensive performance study.

Acknowledgement

This work was supported by the ICT R&D program of MSIP/IITP [B0101-16-0233: Smart Networking Core Technology Development and B0190-15-2012: Global SDN/NFV Open-Source Software Core Module/ Function Development].

References

[1] T. Nadeau, P. Quinn, *Problem Statement for Service Function Chaining*, IETF RFC 7498, April, 2015.

[2] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, SIMPLE-fying Middlebox Policy Enforcement Using SDN, *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, 2013, pp. 27-38.

[3] S. K. Fayazbakhsh, V. Sekar, M. Yu, J. C. Mogul, FlowTags: Enforcing Network Wide Policies in the Presence of Dynamic Middlebox Actions, *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, 2013, pp. 19-24.

[4] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, M. P. Bar Cellos, Survivor: An Enhanced Controller Placement Strategy for Improving SDN Survivability, *IEEE Global Communications Conference 2014*, Austin, TX, 2014, pp. 1909-1915.

[5] F. Botelho, A. Bessani, F. M. V. Ramos, P. Ferreira, On the Design of Practical Fault-tolerant SDN Controllers, *Third European Workshop on Software Defined Networks*, Budapest, Hungary, 2014, pp. 73-78.

[6] D. Katz, D. Ward, *Bidirectional Forwarding Detection,* IETF RFC 5880, June, 2010.

[7] M. Desai, Nandagopal, Coping with Link Failures in Centralized Control Plane Architecture, *IEEE Communication Systems and Networks*, Bangalore, India, 2010, pp. 79-88.

[8] J. Kempf, M. Menth, Scalable Fault Management for OpenFlow, *IEEE International Conference on Communications 2012*, Ottawa, Canada, 2012, pp. 302-336.

[9] D. Staessens, S. Shama, D. Colle, M. Pickavet, P. Demeester, Software Defined Networking: Meeting Carrier Grade Requirements, *IEEE Workshop on Local & Metropolitan Area Networks 2011*, Chapel Hill, North Carolina, 2011, pp. 1-6.

[10] J. Halpern C. Pignataro, *Service Function Chaining (SFC) Architecture*, draft-ietf-sfc-architecture-11, July, 2015.

[11] M. Chiosi et al., *Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action*, ETSI White Paper, October, 2012.

[12] A. P. Quinn, et al., *Network Service Header (NSH)*, draft-quinn-sfc-nsh-07, February, 2015.

[13] A. M. Medhat, G. Carella, C. Lück, M. Corici, T. Magedanz, Near Optimal Service Function Path Instantiation in a Multi-Datacenter Environment, *International Conference on Network and Service Management 2015*, Barcelona, Spain, 2015, pp. 336-341.

[14] S. Sharma, D. Staessens, D. Colle, M. Pickavet, P. Demeester, Openflow: Meeting Carrier-grade Recovery Requirements, *Computer Communications*, Vol. 36, No. 6, pp. 656-665, March, 2013.

[15] Internet2 Abilene Network Topology, http://www.cs.utexas.edu/yzhang/research/ AbileneTM/.

Biographies

Hosung Baek received the B.S. degree from Korea University, Seoul, Korea, in 2015. He is currently an M.S. and Ph.D. integrated course student in School of Electrical Engineering, Korea University, Seoul, Korea. His research interests include 5G networks, programmable dataplane language, and SDN/NFV.

Haneul Ko received the B.S. and Ph.D. degrees from Korea University, Seoul, Korea, in 2011 and 2016, respectively, both in School of Electrical Engineering. He is currently Postdoctoral Researcher with the mobile network and communications, Korea University, Seoul, Korea. His research interests include 5G networks, mobility management, mobile cloud computing, SDN/NFV, and Future Internet.

Gwangwoo Park received the B.S., M.S., Ph.D. degrees from Korea University, Seoul, Korea, in 2010, 2012, and 2016, respectively. He is currently working at Samsung Electronics. His research interests include content networking, multimedia streaming, information-centric networking, and Future Internet.

Sangheon Pack received the B.S. and Ph.D. degrees from Seoul National University, Seoul, Korea, in 2000 and 2005, respectively, both in computer engineering. In 2007, he joined the faculty of Korea University, Seoul, Korea, where he is currently a Professor in the School of Electrical Engineering. He was the recipient of KICS Haedong Young Scholar Award 2013, IEEE ComSoc APB Outstanding Young Researcher Award in 2009, and LG Yonam Foundation Overseas Research Professor Program in 2012. He was a publication co-chair of IEEE INFOCOM 2014, and a co-chair of IEEE VTC 2010-Fall transportation track. He is an editor of Journal of Communications Networks (JCN) and a senior member of the IEEE. His research interests include Future Internet, SDN/ICN/DTN, mobility management, mobile cloud networking, multimedia networking, and vehicular networks.

Jiyoung Kwak received the B.S. and M.S. degrees in computer engineering from Chonnam National University, Korea, in 1999 and 2001, respectively. Since 2001, she has been working as a Senior Research Engineer in the Electronics and Telecommunications Research Institute (ETRI), Korea. Her research interests include intelligent network, SDN, ad-hoc group communication, multimedia middleware, IoT and lightweight OS.