Pre-provisioning of Local Protection for Handling Dual-failures in OpenFlow-based Networks

Pankaj Thorat*, Seil Jeon†, Syed M. Raza‡ and Hyunseung Choo§
*‡§College of Software, Sungkyunkwan University (SKKU), Suwon, South Korea,
†College of Information and Communication Engineering, Sungkyunkwan University (SKKU), Suwon, South Korea,
Email: *pankaj@skku.edu, †seiljeon@skku.edu, ‡s.moh.raza@skku.edu, §choo@skku.edu

***Abstract*—An essential requirement in operating a carrier-grade network (CGN) is ensuring the high availability and reliability. Software-defined networking (SDN) is expected to address such requirement while improving the network management. One challenging issue faced in the process of enhancing the reliability of SDN-enabled CGN is how to achieve rapid recovery with minimal effort. There are two well-known approaches to determine the failover scope: end-to-end (global) detouring and local detouring. Particularly, the local detouring approach provides an efficient means to achieve faster recovery, as it locally detours the disrupted flows around the failed network components using a preconfigured alternative path. However, it requires thousands of flow entries per switch to be configured. To address the technical challenges, we propose a fault-tolerant forwarding table design (FFTD), which groups the flows using group entries and aggregates the flows using a tagging mechanism for scalable and rapid recovery from the dual-failures of switches or links without overburdening the controller and the flow table's memory. Our extensive emulation results reveal that the proposed FFTD satisfies the CGN's 50 ms recovery requirement. Additionally, it reduces the alternate path flow storage requirement by up to 99%.**

***Index Terms*—OpenFlow, Software-defined networks, Computer network reliability, Fault tolerance, Network management**

I. Introduction

For the timely service delivery, it is necessary to minimize the service disruption time on a link or a switch failure. The traditional routing methods based on path-vector, distance-vector, and link-state usually take several seconds to update the routing table and reroute the connections after the failure detection. For the time-critical applications in the carrier-grade networks (CGN), the desired failure recovery is required to happen within 50 ms [1]. Over the years, Software-defined networking (SDN) has been employed in various use cases such as mobile networks [2], data center networks [3], wide-area networks [4] for simplifying the network management with a holistic view of the network, based on the control and data plane separation [5]. Such networking paradigm presented by SDN is expected to facilitate a rapid and efficient recovery in the CGN [2], [6].

A recovery can be achieved using either a restoration mechanism or a protection mechanism [7]. In the case of restoration mechanism, the controller finds and configures the alternate path to detour the disrupted flows after receiving the failure notification [8]–[10]. In SDN, the controller must instantly find and configure the alternate paths for disrupted flows to achieve a faster recovery. However, the time spent for the recovery procedure depends on the number of disrupted flows to be detoured, so it is challenging to guarantee a fixed recovery time. An attempt to rapidly recover a higher number of flows may overburden a resource-constrained controller and degrade its performance. Moreover, the recovery-specific flow modification messages may overwhelm the failure-affected switches.

In the protection mechanism, the SDN controller pre-provisions the alternate paths so that the source switch on the failure-affected path can reroute the disrupted flows without any assistance from the controller. Several proposals, based on the end-to-end path protection mechanism, have reduced the recovery time in comparison to the restoration-based approaches [11]–[16]. However, provisioning the alternate path for every flow on the network requires considerable Ternary Content-Addressable Memory (TCAM) to store the flow tables, which could magnify with the increase in the number of flows to protect. Preconfiguring the flow rules of the primary and alternate path of every incoming flow causes an additional overhead on the controller.

In an attempt to conserve TCAM while achieving the link recovery, authors of [17] proposed to label the failure-affected flows, so they can be identified using a common flow entry on the alternate path. Authors of [18], [19] proposed a flow compression scheme to dynamically merge the alternate path flow rules of the incoming flows with the existing flows on the alternate path based on the common header fields. Such dynamic compression procedure with the arrival of every new flow may increase the burden on the controller. The possible compression depends on the common header fields between the existing flow entry and the incoming flow, which makes the TCAM resource allocation planning unpredictable.

Most of the current failure management proposals in SDN can be categorized based on its recovery scope i.e. link recovery or switch recovery. There are few proposals that handle the joint recovery from the link and switch failure [11], [20]. Authors of [11] proposed a path protection-based joint recovery scheme, but it suffers from the inherent shortcomings of path protection mechanisms. From the review of the previous studies and its limitations for a rapid and lightweight recovery, we conclude that the recovery scheme must follow following requirements; (i) protection mechanism should be employed, (ii) TCAM consumption should be minimum, (iii)

978-3-901882-98-2 © 2017 IFIP

recovery should be local, (iv) intervention of the controller should be minimum.

In this paper, we extend our earlier work in [20] and propose a fault-tolerant forwarding table design (FFTD) for a scalable and rapid joint failure recovery on dual-failures. Our proposed FFTD guarantees an end-to-end available path even if a flow encounters an additional link or switch failure on the alternate path. FFTD provides the local protection to network components (link/switch) on the alternate path calculated for every switch/link in the network. We modified the flow aggregation and flow grouping strategies used in our previous work for handling single failures to efficiently handle the dual-failures. We evaluated the overall performance regarding the control plane recovery overhead, recovery time, and alternate path flow rules requirement. From the early evaluation results, we confirm that our proposed FFTD takes 3 ms on average for detouring without the proportional increase of TCAM requirement.

II. Background Work

A. *Controller Independent Proactive Network Design for Single Link/Switch Recovery*

In the previous work [20], we presented a controller independent proactive (CIP) network design with flow grouping and flow aggregation methods to achieve a rapid and lightweight failure recovery from a single switch or link failure. The Fast Failover (FF) group feature of the OpenFlow has been used for grouping the flows. It detects the port failure and locally detour the disrupted flows on the preconfigured alternate path. The flows having the common action output port are redirected to the FF group entry in the group table, which further executes the primary action bucket being responsible for forwarding the packet onto the output port as shown in Fig. 1. An FF group entry consists of a group ID, counters, and set of action buckets. If the output port status in the first action bucket goes down, it would be no longer available, and then the FF group executes the next available action bucket. Such local detouring of the grouped flows eliminates the need of controller intervention for per-flow detouring, and thus enables faster recovery.

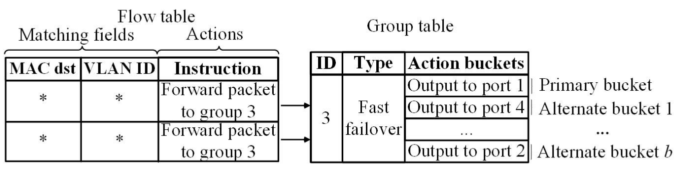


Fig. 1: Structure of FF group.

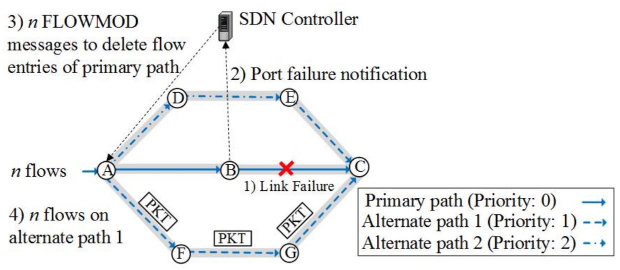
A flow on its primary path originates from the source edge-switch and reaches to the destination edge-switch. Before reaching to the destination, it may traverse zero or more core-switches or links and an edge-link connected to the destination edge-switch. For the flow aggregation, the CIP scheme labels the flows with failed component ID so that all the detoured flows can be identified using a single flow entry in the switches of alternate path. The flows that have encountered the upstream core-link/switch failure are tagged with the upstream switch ID, while the flows that have encountered the upstream edge-link failure are tagged with the upstream edge ID. Therefore, instead of configuring the alternate path on per-flow basis, a single flow entry matching Virtual Local Area Network (VLAN) label and input port in each switch of the alternate path performs forwarding. Therefore, the flow aggregation strategy reduces the TCAM consumption of the switches, and also eliminates the burden on the controller to configure the alternate path for every incoming flow.

For the aggregation of flows on its core component, the flows were tagged with the core-switch ID and then redirected to the group entry that forwards packets to the 2-hop neighboring switch of the detour switch. However, in the case of multiple destination switches, the controller needs to store information about all the group entries in the network and retrieve it while pointing the primary path flow rule to its respective group entry. Additionally, it also needs to store the information about group entries responsible for handling edge-link failure. This approach of flow aggregation could increase the storage as well as computational cost at the controller. The CIP achieves recovery from a single switch or link failure. Therefore, if the flows encounter an additional link or a switch failure on the alternate path, then the provisioned alternate path configuration cannot maintain the flow connectivity. However, it is crucial to provide the resilience against multiple link or node failures because almost 41% of the failures in data centers involves multi-link failures [21]. The most common approach for handling multiple failures is multipath protection, where the disrupted traffic is switched from primary path to the disjoint alternate path.

B. *Multipath Protection for Multiple Recovery*

Yang et al. proposed a multipath protection scheme for handling multiple failures in OpenFlow-based networks [22]. After failure on the primary path with the highest priority (0), the controller deletes the flow entries of the primary path from the source switch to enable disrupted flows to select the flow entries of the alternate path with lower priority (1). To protect flows against m failures, it configures m disjoint alternate path with distinct priorities. Fig. 2 illustrates the functionality of multipath protection mechanism for handling dual-failures [22], where the controller preconfigures the alternate path 1 to protect flows on primary path and an additional alternate path 2 to protect flows on alternate path 1. On failure on the primary path, the switch affected by the failure transmits a port down notification to the controller. In response to the failure notification, the controller deletes n flows rules of primary path from the source switch that enables the incoming flows to detour to alternate path 1.

For the multipath protection, the dependence on the controller for detouring the higher number of disrupted flows may increase the recovery time. It may also overburden the controller with the procedure for per-flow detouring. Moreover, the configuration of the alternate path for every flow



3) n FLOWMOD messages to delete flow entries of primary path
SDN Controller
2) Port failure notification
D
E
n flows
A
B
C
1) Link Failure
4) n flows on alternate path 1
PKT
PKT
PKT
F
G
Primary path (Priority: 0)
Alternate path 1 (Priority: 1)
Alternate path 2 (Priority: 2)

Fig. 2: Multipath protection mechanism.

on the primary path increases the TCAM requirement for storing flow rules. After switching to the alternate path, the forwarding entries for the disrupted flows become obsolete and need to be deleted to free the switch resources. Considering the challenges in the existing work and limitation of our previous work to handle just a single link/switch failure, we propose an FFTD for scalable and rapid recovery from the dual-failures at the data plane.

III. Fault-tolerant Forwarding Table Design for Handling Dual-Failures in SDN

In this section, we describe how dual-failures can be locally handled; how FFTD configures the switches for the failover operation with an example.

A. *Configuration of Fault-tolerant Forwarding Tables*

To facilitate the local recovery against dual-failure, FFTD configures the forwarding rules in the alternate path switches such that it can locally reroute the flows around the second failed component on the alternate path. FFTD employs flow grouping and flow aggregation methods with the FF group table provided in the OpenFlow specification, and VLAN tagging mechanism, respectively. In the proposed FFTD, instead of tagging the flows with the failed component ID as done in our previous work [20], the flows on its upstream edge-link are tagged with the edge-switch ID, while, the flows on its upstream core-link are tagged with ID of the 2-hop neighboring switch of the detour switch. On the alternate path, the disrupted flows are forwarded using single flow entry matching the ID of immediate 2-hop or 1-hop neighboring switch. This approach of flow aggregation reduce the complexity of flow aggregation in dual-failure handling.

The FFTD preconfigures an alternate path between 1-hop neighbors for every switch. The source switch of the alternate path, which acts a detouring switch on failure, is configured with an FF entry with three action buckets, where the first bucket forwards the packets on the primary path while the second and third buckets forward the packets on the alternate paths. For the simplicity of understanding, we denote the alternate path between 1-hop neighbor as AP1 and the alternate paths configured for protecting the network components on the AP1 as AP2. The group ID is set to the destination switch ID in AP1.

Intermediate switches on the AP1, except the switch attached to the destination switch, are configured with a flow entry and FF group entry, where the packets are matched using the ID of the destination switch and forwarded to FF group entry whose first action bucket forwards the packet on AP1 and the second action bucket, on a failure of any component on AP1, forwards the packet on AP2. In the switch attached to the destination switch, the second action bucket of FF group entry is configured to forward the packets to the next-hop of the destination switch, which is a part of the primary path, to mitigate the failure of the destination switch. Intermediate switches on the AP2 are configured with a flow entry, which matches the packets using the ID of the destination switch and forwards it on the AP2 till the flow merges with the AP1.

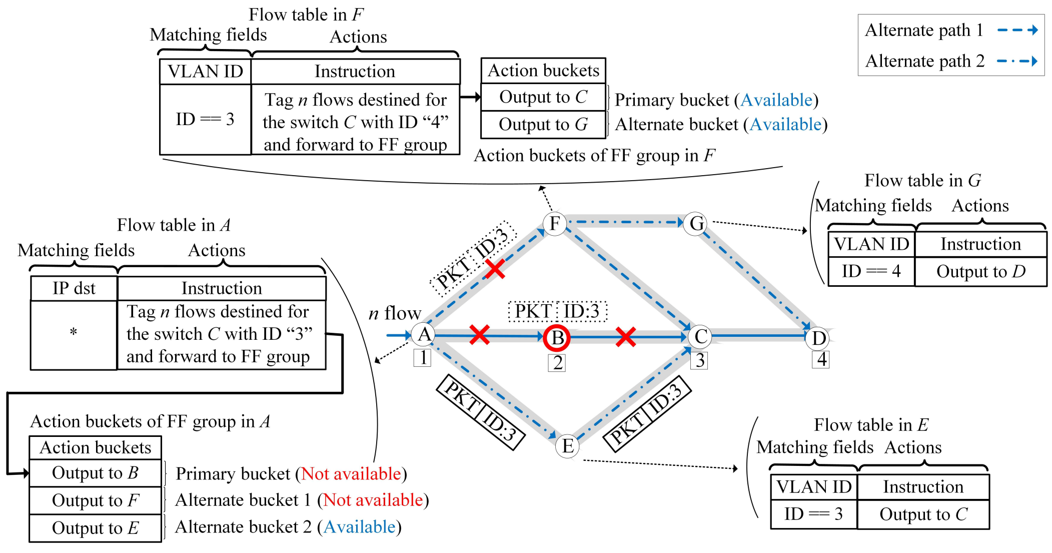
The edge-switch recovery is not possible; but, the edge-link recovery is possible. Therefore, the failure of edge-link should not be treated as the edge-switch failure. For the edge-link protection, all the intermediate switches on the AP1 are configured with a flow entry and FF group entry, where the packets are matched using the ID of the failed edge-switch and forwarded to FF group entry whose first action bucket forwards the packet on AP1 and the second action bucket, on failure of any component on AP1, forwards the packet on AP2. Intermediate switches on the AP2 are configured with a flow entry, which matches the packets using the ID of the edge-switch and forwards it on the AP2 till the flow merges with the AP1.

For an incoming flow, switches on the primary path, except the switches attached to edge-link, are configured to tag the packets with the ID of its two-hop neighbor and then use the same ID to forward the packets to the preconfigured FF group entry. The downstream switch attached to edge-link is configured with a flow entry to tag the packets with edge-switch ID and then use the same ID to forward it to the preconfigured FF group entry. In the switch attached to the destination host is configured to pop the VLAN header and forward packet to the destination host.

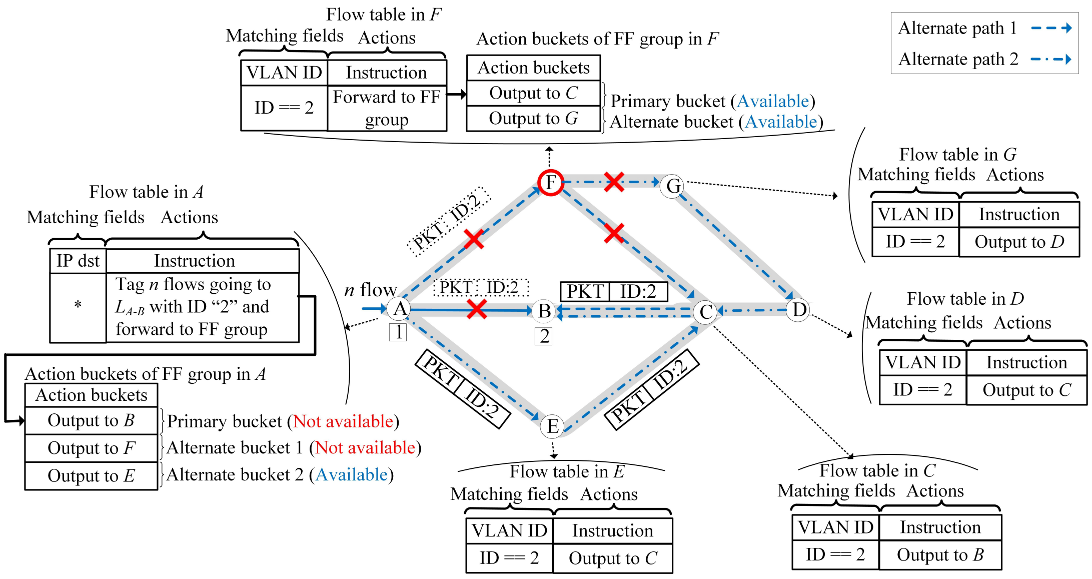
B. *Operations*

The snapshot in Fig. 3(a) shows the failover operation for core-switch SW_B in the primary path and link L_{A-F} in the AP1, where the SW_B denotes the switch B and L_{A-F} denotes the link between switch A and F. The packets of incoming n flows at SW_A are tagged with the ID “3”, which is the ID of 2-hop neighboring switch SW_C, and forwarded to the FF group whose primary bucket, alternate bucket 1, and alternate bucket 2 forwards the packet to SW_B of primary path, SW_F of AP1, and SW_E of AP2, respectively. To handle the failure of SW_B or L_{A-B} (core components), AP1 ($SW_A \rightarrow SW_F \rightarrow SW_C$) is configured, which reroutes the flows around the SW_B. Similarly to handle the failures of L_{A-F} or SW_F on AP1, AP2 ($SW_A \rightarrow SW_E \rightarrow SW_C$) is configured. For the protection against the failure of 2-hop neighbor SW_C or L_{F-C}, AP2 ($SW_F \rightarrow SW_G \rightarrow SW_D$) is configured.

- On a failure of SW_B and L_{A-B}, SW_A detects the failure of ports corresponding to L_{A-B} and L_{A-F}.



(a) Core switch/link protection.



(b) Edge link protection.

Fig. 3: Failover operation of FFTD.

- SW_A change the status of the primary bucket and alternate bucket 1 to unavailable and executes the alternate bucket 2, which forwards the n disrupted flows with label "3" to SW_E.
- A single VLAN label and input port matching flow entry in SW_E matches and forwards the n disrupted flows to the SW_C.

The snapshot in Fig. 3(b) shows the failover operation for the flows having L_{A-B} as edge-link in the primary path and switch SW_F in the AP1. The packets of incoming n flows at SW_A are tagged with the ID "2", which is the ID of edge-switch SW_B for n incoming flow and then forwarded to the FF group whose primary bucket, alternate bucket 1, and alternate bucket 2 forwards the packet to SW_B of primary path, SW_F of AP1, and SW_E of AP2, respectively. To handle the failure of the L_{A-B}, AP1 ($SW_A \rightarrow SW_F \rightarrow SW_C \rightarrow SW_B$) is configured, which reroutes the flows around L_{A-B}. Similarly to handle the failures of L_{A-F} or SW_F on AP1, AP2 ($SW_A \rightarrow SW_E \rightarrow SW_C \rightarrow SW_B$) is configured. For the protection against the failure of L_{F-C}, a dedicated AP2 ($SW_F \rightarrow SW_G \rightarrow SW_D \rightarrow SW_C \rightarrow SW_B$) is configured.

- On a failure of edge-link L_{A-B} and SW_F, SW_A detects the failure of ports corresponding to L_{A-B} and L_{A-F}.
- SW_A change the status of the primary bucket and alternate bucket 1 to unavailable and executes the alternate bucket 2, which forwards the n disrupted flows with label "2" to SW_E.
- A single VLAN label and input port matching flow entry in each of SW_E and SW_C matches and forwards the n disrupted flows to the SW_B.

IV. Performance Evaluation

For performance evaluation, we emulate the AT&T topology with 25 switches and 52 links in Mininet, which creates a virtual network with OpenFlow-enabled switches [23]. For SDN controller, we install the CPqD version of NOX controller on a Linux server with an Intel(R) CPU 2.60GHz 12 core processor and 64 GB RAM along with the Mininet [24]. In our emulation environment, there exists a dedicated network connection between the controller and each of the network switches, which is termed as the out-of-band connection. We use iPerf to generate UDP traffic flows with the data rate of 100 Mbps and packets size of 50 Bytes in the network [25].

For performance evaluation of the proposed FFTD in terms of data plane failure recovery, we install the flows traveling from San Francisco (SFO) to Washington (IAD) via Kansas (MCI), St. Louis (STL), Nashville (BNA) and Atlanta (ATL) as shown in Fig. 4. For the simplicity of representation, cities are labeled with its airport code. To test the recovery from dual-failures of core components, we initially failed the core-switch MCI of the primary path and then the link between SFO and Dallas (DFW) of the AP1. Similarly to evaluate dual-failures recovery at edge-link, we first failed the link between ATL and IAD and then the switch at Rayleigh (RLGH). We compared the performance of the proposed FFTD with multipath recovery, where two alternate paths are preconfigured for protection against dual-failures. To protect the primary path between SFO and IAD with highest priority (0), the implemented multipath protection approach preconfigures the first alternate path as ⟨SFO-DFW-New Orleans (MSY)-Orlando (MCO)-Rayleigh (RLGH)-IAD⟩ with lower priority (1) and the second alternate path as ⟨SFO-Denver (DEN)-Chicago (ORD)-Cleveland (CLE)-Philadelphia (PHL)-IAD⟩ with lowest priority (2).

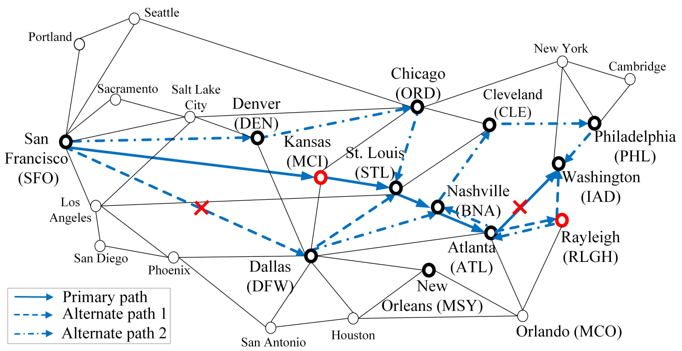
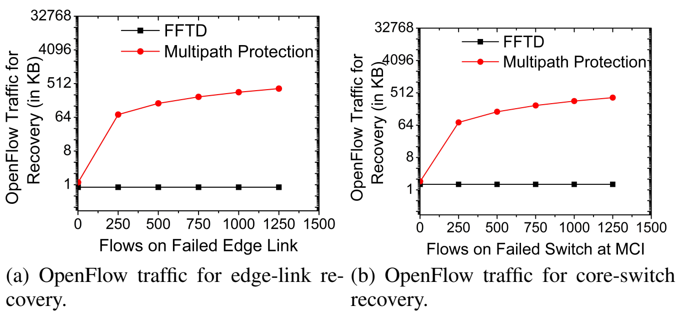


Fig. 4: AT&T network topology.

A. OpenFlow Control Traffic in the Network

We measure the variation in OpenFlow control traffic from the failure notifications to the recovery-specific messages generated by the controller for increasing number of flows to be recovered with two network failures use cases: (a) edge-link failure and (b) core-switch failure. The purpose of this experiment is to analyze and compare the control traffic overhead at the controller among the target schemes. As mentioned in the previous paragraph, we repeated the same failure sequence to analyze the OpenFlow traffic generated for dual-failures recovery of the core-switch and edge-link.



(a) OpenFlow traffic for edge-link recovery. (b) OpenFlow traffic for core-switch recovery.

Fig. 5: OpenFlow control traffic in the network.

For both of the graphs in Fig. 5, the OpenFlow recovery traffic increases as the number of flows to be recovered increases for the multipath protection scheme due to its per-flow detouring granularity. On the contrary, in the proposed FFTD, the post-failure traffic remains constant irrespective of the number of flows to be recovered. The proposed FFTD benefits from the forwarding table design enabled with the flow grouping and aggregation. From the results obtained, we identify the proposed recovery schemes are scalable and efficient in handling a large number of flows.

B. Switchover Time

This experiment compares the switchover time on the network failure events for increasing number of flows to recover, which is same for the edge-link and core-switch failure. To evaluate the impact of flows to be recovered on the failure recovery time as shown in Fig. 6, we repeated the same failure sequence to analyze the switchover time for achieving dual-failures recovery. The overall failure recovery time is measured as the time difference from the time the network component failed to the time the last disrupted flow was detoured.

From the results shown in Fig. 6, it is confirmed that the recovery time for multipath protection scheme depends on the number of disrupted flows, as the per-flow detouring requires granular operation of the controller. On the contrary, due to the preconfigured flow grouping method employed in the proposed FFTD, the recovery time is mostly constant irrespective of the increasing number of flows to be recovered, on the strengths of no controller intervention. The proposed FFTD scheme meet the carrier-grade requirement to recover from the failure within 50 ms interval.

C. Alternate Path Forwarding Rules

In Fig. 7, the x-axis represents the flows on the primary path and y-axis represents the total number of forwarding rules (group entries and flow entries) required to protect flows passing through switch at MCI and edge-link ⟨BNA-IAD⟩ from dual-failures. Multipath protection scheme configures two alternate paths for every flow on the primary path. Therefore, the number of flow rules required to configure the alternate paths

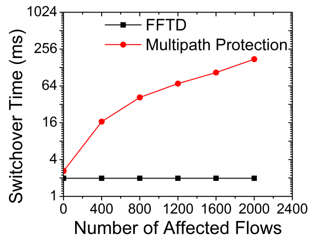


Fig. 6: Switchover time.

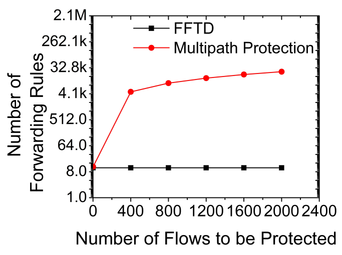


Fig. 7: Alternate path flow rules requirement.

increased with the higher number of flows to be protected. However, with the flow aggregation method employed in the FFTD, the alternate path forwarding rules requirement remains constant irrespective of the increasing number of flows to be recovered. The obtained results identified proposed FFTD can reduce the alternate path rules in the flow table by more than 99% and results in smaller flow table size and switch memory saving.

V. Conclusion and Future Work

In this paper, we proposed the FFTD to achieve a rapid recovery from dual-failures without overwhelming the controller with control traffic. Our proposal considerably reduces the switchover time using the flow grouping method and lowers the memory requirement for alternate path setup using the flow aggregation method. Based on the performance evaluation, the proposed FFTD achieved 99% reduction in the flow storage for alternate path setup using the flow aggregation method. We showed that the controller dependence to perform the recovery results in delayed recovery and higher traffic at the controller in a very short interval. In addition, we could identify that FFTD achieved switchover in around 3 ms and fulfilled the Carrier-grade recovery requirement of 50 ms failover delay. As a further work, we are working on handling the traffic congestion that may occur after the localized recovery.

VI. Acknowledgments

This research was supported in part by MOE and MSIP, Korean government, under the G-ITRC support program (IITP-2017-2015-0-00742) and ICT R&D program (B0101-15-1366) through IITP and PRCP (NRF-2010-0020210) through NRF.

References

[1] W. John, A. Kern, M. Kind, P. Skoldstrom, D. Staessens, and H. Woesner, "Splitarchitecture: Sdn for the carrier domain," *IEEE Communications Magazine*, vol. 52, no. 10, pp. 146–152, 2014.

[2] E. Logota, D. Corujo, S. Jeon, J. Rodriguez, and R. L. Aguiar, "The 5g internet," in *Fundamentals of 5G Mobile Networks*. John Wiley & Sons, Ltd, 2015, pp. 29–62.

[3] Y. Han, S.-s. Seo, J. Li, J. Hyun, J.-H. Yoo, and J. W.-K. Hong, "Software defined networking-based traffic engineering for data center networks," in *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*. IEEE, 2014, pp. 1–6.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[6] P. Thorat, S. M. Raza, D. T. Nguyen, G. Im, H. Choo, and D. S. Kim, "Optimized self-healing framework for software defined networks," in *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*. ACM, 2015, p. 7.

[7] E. Harrison, A. Farrel, and B. Miller, "Protection and restoration in mpls networks," *Data Connection White Paper*, 2001.

[8] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in openflow networks," in *Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the*. IEEE, 2011, pp. 164–171.

[9] Y. Yu, L. Xin, C. Shanzhi, and W. Yan, "A framework of using openflow to handle transient link failure," in *2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*. IEEE, 2011, pp. 2050–2053.

[10] B. Raeisi and A. Giorgetti, "Software-based fast failure recovery in load balanced sdn-based datacenter networks," in *International Conference on Information Communication and Management (ICICM)*. IEEE, 2016, pp. 95–99.

[11] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Openflow: Meeting carrier-grade recovery requirements," *Computer Communications*, vol. 36, no. 6, pp. 656–665, 2013.

[12] ——, "Fast failure recovery for in-band openflow networks," in *2013 9th international conference on the Design of reliable communication networks (drcn)*. IEEE, 2013, pp. 52–59.

[13] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 1066–1075, 2013.

[14] ——, "Effective flow protection in open-flow rings," in *National Fiber Optic Engineers Conference*. Optical Society of America, 2013, pp. JTh2A–01.

[15] Y.-D. Lin, H.-Y. Teng, C.-R. Hsu, C.-C. Liao, and Y.-C. Lai, "Fast failover and switchover for link failures and congestion in software defined networks," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[16] N. L. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *2014 Third European Workshop on Software Defined Networks*. IEEE, 2014, pp. 61–66.

[17] J. Chen, J. Chen, J. Ling, and W. Zhang, "Failure recovery using vlan-tag in sdn: High speed with low memory requirement," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, Dec 2016, pp. 1–9.

[18] B. Stephens, A. L. Cox, and S. Rixner, "Scalable multi-failure fast failover via forwarding table compression," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 9.

[19] ——, "Plinko: building provably resilient forwarding tables," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013, p. 26.

[20] P. Thorat, S. Jeon, and H. Choo, "Enhanced local detouring mechanisms for rapid and lightweight failure recovery in openflow networks," *Computer Communications*, vol. 108, pp. 78 – 93, 2017.

[21] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[22] H. Yang, L. Cheng, J. Yuan, J. Zhang, Y. Zhao, and Y. Lee, "Multipath protection for data center services in openflow-based software defined elastic optical networks," *Optical Fiber Technology*, vol. 23, pp. 108–115, 2015.

[23] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[24] CPqD, "Nox zaku with openflow 1.3 support," https://github.com/CPqD/nox13oflib, 2013.

[25] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *http://dast.nlanr.net/Projects*, 2005.