©2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

Scalable Resilience for Software-Defined Networking Using Loop-Free Alternates with Loop Detection

Wolfgang Braun, Michael Menth
University of Tübingen, Department of Computer Science, Germany

***Abstract*—In this paper we propose a novel resilience scheme for OpenFlow-based Software-Defined Networking (SDN). To forward packets in line speed, OpenFlow switches store their flow tables in expensive, limited TCAM due to which the stored tables cannot be large. Most resilience mechanisms require additional entries thus the implementation in OpenFlow may quickly exceed the available TCAM.**

Loop-Free Alternates (LFAs) are a standardized mechanism for fast reroute in IP networks which do not require additional entries. However, LFAs cannot protect against all single link and node failures. Moreover, some LFAs even cause loops in case of some node failures or multiple failures, making additional links unusable. Excluding such LFAs reduces the fraction of protected destinations (protection coverage) in nodes even further.

We suggest a scheme for detection of loops caused by LFAs. It maximizes the protection coverage because it allows all LFAs to be used without creating loops. We describe how LFAs and the loop detection scheme can be implemented in OpenFlow networks with only little packet overhead and a single additional entry per switch.

We evaluate our proposal on real network topologies of varying size and connectivity. In particular, we quantify the benefit gained from loop detection. Amongst others, we study the percentage of failures that cannot be protected by LFAs at all, and the percentage of failures in which conventional LFAs cause extra loops.

***Index Terms*—Software-Defined Networking, OpenFlow, Resilience, Loop-Free Alternates, Scalability**

I. Introduction

Software Defined Networking [1] (SDN) has gained a lot of attention in the recent years due to its ability to program the network and introduce new features and services easily. Many of these approaches require many fine-grained flow definitions to control the traffic as desired [2]. This can significantly increase the number of required flow table entries. However, flow table entries must be stored in Ternary Content Addressable Memory (TCAM) to enable fast packet matching in OpenFlow switches. TCAM is expensive and, thus, often limited in size which can lead to severe scalability issues with OpenFlow-based SDN. Another example is inter-domain routing with OpenFlow [3]. Current routing tables contain so many IP prefixes that only a few or even none additional forwarding entries can be stored in the forwarding tables.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under support code 16BP12307 (EUREKA-Project SASER). The authors alone are responsible for the content of the paper.

Resilience is an important topic for OpenFlow-based SDN due to its separation of the control and data plane. Switches rely on the correct and timely configuration of its controller. Network failures can affect the performance of an OpenFlow network in several ways. The switch has to drop the packets until the controller reconfigures the switch. The situation is even more severe if the OpenFlow control channel is transmitted over the same physical infrastructure because it can be affected by the failure itself. Therefore, researchers are interested to establish backup paths that can mitigate the effects of failures.

In this work we address both the resilience and the scalability issues in OpenFlow networks. We are interested in a resilience mechanism that (1) has full coverage against single link failures, (2) requires the least amount of state in the switch, and (3) can handle node and multiple link failures appropriately. In this work, we focus on the implementation of LFAs which are a simple rerouting mechanism without additional state requirements. However, LFAs do not provide full coverage against all failures and do not fulfill (1) and we will address this in future work. In addition, we propose LFAs with loop detection that protects more traffic flows than LFA which are loop-preventing.

The remainder of the paper is structured as follows. Section II discusses related work in the area of fast reroute and scalability for OpenFlow. We present the concept of LFAs in Section III. Section IV explains how LFAs can be implemented with OpenFlow and we propose the loop-detecting LFAs. Section V presents the coverage statistics and Section VI summarizes this work and outlines our future work.

II. Related Work

In [4], the restoration process of OpenFlow in carrier-grade networks was analyzed. They measured the time a controller re-configures the switches in a real testbed when network failures occur. Their results show that the controller reacts in a time frame between 80 and 130 ms. They also state that the restoration time will be a magnitude higher for large networks and, therefore, local protection schemes are required.

Local failure protection is considered in [5]. They introduce a bidirectional forwarding detection (BFD) component to the OpenFlow switch design to achieve fast protection. A BFD component is responsible to monitor the ports by sending *hello* and *echo* messages frequently over the links to detect network

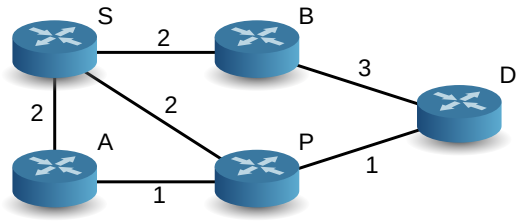

Fig. 1: Example network for LFA computation.

failures. The authors show the viability of the approach using experiments based on MPLS Transport Profile. A similar approach is presented in [6]. The authors also provide fast protection using BFD for the Open vSwitch and analyze the protection switching times. They were able to show that the implementation was able to react within 3 and 30 ms depending on different BFD configurations.

The SlickFlow approach [7] provides resilience in data center networks (DCN) using OpenFlow and is based on source routing. Primary and alternative backup paths are encoded in the packet header and OpenFlow primitives enable hardware-based forwarding. The authors showed positive benefits in a virtualized testbed on DCN topologies.

IP fast reroute (FRR) is an important topic for the routing working group in the IETF and various approaches are discussed: LFAs [8], remote LFAs (rLFAs) [9], [10], not-via addresses, and maximally redundant trees [11]. LFAs and rLFAs are simple and require no additional forwarding entries but cannot protect against all single link failures. The latter mechanisms provide full coverage but require additional forwarding entries. The combination of LFAs and not-via addresses cannot significantly reduce the required state [12]. MPLS FRR [13] defines several ways to protect the network against failures but explicit backup paths require additional entries which can be reduced by the use of shared tunnels. A comparison of IP-based and MPLS FRR is given in [14].

Inter-domain routing for OpenFlow is improved using wildcard compression in [3]. However, the required forwarding state is still challenging and there is not much space left in the forwarding entries for backup paths. Another compression for Access Control Lists (ACLs), i.e., applicable to OpenFlow table entries, is proposed in [15]. This method preserves space in the forwarding table that can be used for backup paths.

III. Loop-Free Alternates (LFAs)

Loop-Free Alternates (LFAs) [8] were proposed by the IETF for IP fast reroute (IP FRR). LFAs are simple but cannot protect against all single link and node failures. They do not require additional forwarding entries but a forwarding entry has to contain a list of alternate next-hops.

The idea of LFAs is very simple: if a failure occurs, packets can be sent to a neighbor if this redirection will not cause a loop. LFAs can be easily computed with a distance function $\mathrm{dist}(u, v)$ that is used for the primary hop computation. LFAs can protect against (1) single link failures, (2) node failures, and (3) multiple network failures depending on the use of the following three LFA conditions. We will explain these conditions by example with the network shown in Figure 1. Packets are sent from source S towards destination D on the shortest path through node P.

The loop-free condition (LFC) protects against single link failures. A neighbor N of S fulfills the condition if

$$\mathrm{dist}(N, D) < \mathrm{dist}(N, S) + \mathrm{dist}(S, D) \tag{1}$$

holds. In the example, this is true for the nodes A (LFC: $2 < 5$) and B (LFC: $3 < 5$) because the distance from them towards destination is smaller than the redirection over S.

The node protecting condition (NPC) prevents the selection of neighbors that may cause a loop if the primary next-hop has failed. The condition is defined as

$$\mathrm{dist}(N, D) < \mathrm{dist}(N, P) + \mathrm{dist}(P, D). \tag{2}$$

Consider that node P in the example has failed. Node A fulfills the LFC but not the NPC ($2 \not< 2$). Packets sent to A will loop because S is a simple LFA for A to D (LFC: $3 < 4$). Node B is node protecting ($3 < 5$) and packets sent to B will not traverse P.

The downstream condition (DSC) protects against multiple failures by only redirecting traffic to neighbors that are closer to the destination:

$$\mathrm{dist}(N, D) < \mathrm{dist}(S, D). \tag{3}$$

Node B is not downstream ($3 \not< 3$) and packets would loop between B and S when both $S \to P$ and $B \to D$ are failing because S is a simple LFA for B towards D ($3 < 5$). A is downstream ($2 < 3$) and will not cause a loop even if $A \to P$ has failed because S is not downstream for A towards D ($3 \not< 2$). Note that DSC and NPC are orthogonal, i.e., a downstream LFA can either be node protecting or not.

In this work we compute three kinds of LFAs. First, simple LFAs that only fulfill the loop-free condition (LFA-LFCs) which may cause loops when node or multiple failures occur. Second, node protecting LFAs that both fulfill LFC and NPC (LFA-NPCs) that prevent loops for single link and single node failures, and finally loop-preventing LFAs that fulfill the downstream condition (LFA-DSCs).

IV. OpenFlow and LFAs

In this section, we propose LFAs with loop detection that use additional failure information in the packet header to drop looping packets. We discuss how LFAs with and without loop detection can be implemented with OpenFlow. Then, we discuss how failure information can be encoded in packets with little packet overhead.

A. Improving LFAs with Loop Detection

When LFAs for a network are computed to protect against node failures, the node protecting condition can exclude neighbors that protect against single link failures. In general, this leads to fewer alternate next-hops that are allowed to be selected. In addition, if only link failures occur, there may be

a LFA available but it is not allowed to be chosen due to the more restricted condition.

Our LFA approach selects LFAs with the highest protection first and reverts to less protecting LFAs if necessary, i.e., we select potential LFAs with degrading degree of protection: (a) NPC and DSC, (b) DSC, (c) LFC and NPC, and (d) LFC only. LFAs of category (a) cannot loop and LFAs with (b)-(d) may loop depending on the exact failure scenario. Therefore, we apply a mark for LFAs (b)-(d) into the packet header. This mark encodes the failure detecting node. Packets either are successfully redirected towards the destination or a loop occurs. In the latter case, the node can check if the packet contains its location mark and prevents the loop by dropping the packet.

It is important that marks can be incrementally applied to a packet that is sent on alternative paths when multiple failures occur. We explain this with the example network shown in Figure 1. Consider that packets are sent from S to D and the links $S \rightarrow P$ and $A \rightarrow P$ have failed. Packets have to be redirected over A because there is no LFA of category (a) and A is a LFA of category B. The mark for node S is applied and redirected to A. The packet is not dropped because it has no mark for node A and S is a LFA of category d for A. The mark for A is applied and the packet is sent to S. The loop can be successfully detected in S if the mark of A does not overwrite the mark of S.

Such marks can be implemented using a bit string of a certain length n. Each node has an ID i with $1 \leq i \leq n$ and its mark corresponds to the i-th bit in the string. If the number of available bits in the field is less than or equal to the number of nodes, the loop detection is optimal. If there are more nodes in the network than available bits, we share IDs across nodes. This can cause false positives when detecting loops and, thus, can lead to unnecessary packet drops. We provide an algorithm to compute appropriate IDs in Section IV-C and analyze the impact of various bit lengths in Section V. We refer LFAs with loop detection to LFA-IDs.

B. Implementation in OpenFlow and State Requirements

OpenFlow data plane resilience requires OpenFlow 1.1 [16] or later and such an OpenFlow switch contains both flow tables and a group table. A flow table consists of multiple entries and each entry consists of a match and instructions. Flow table entries match packets to flows and each flow entry can define various actions or refer to a group table entry for advanced packet processing. Note that the flow tables have to reside in TCAM which is expensive and limited in size but enables fast packet processing. The group table must not be stored inside TCAM and can be part of less expensive memory.

A group table entry can be referenced by multiple flow table entries and handles all related packets in the same fashion. The group table entry contains a group type, a counters field, and a field for action buckets. The group type defines the kind of group. Backup paths can be implemented using the *fast failover* group type. The action buckets are used to implement the primary and secondary paths. For *fast failover* groups, each action bucket has an associated port which defines its *liveness* and triggers the use of a bucket.

We explain the group table for simple LFAs by the example given in Section IV-A. Packets are sent from S to D. The switch at S contains a flow table entry that matches specific header fields, e.g., its IP address and this entry refers to a group entry. The group entry has type *fast failover* and consists of two action bucket. The first action bucket contains the action to "forward to P" and the second bucket "forward to A". If the link to node P goes down, packets are immediately sent over the next defined bucket, i.e., to node A. Note that no additional flow table entries are required for LFAs.

LFAs with loop detection are implemented similarly. Consider that the ID for S is 1 and the used bit string has length 5. We add an additional flow table entry that matches on the bit string with the wildcard expression $****1$ and its action is "packet drop". Thus, only packets that loop back to S, i.e., are marked with ID 1, will be dropped. The first action bucket of the group is unchanged. The second action bucket now contains two actions: "apply ID 1" and "forward to A".

In OpenFlow, all header fields that are used for forwarding have the potential for ID encoding. For example, the DSCP and ECN field (8 bits) of an IP header can be used if they are not needed in the OpenFlow network. Another potential solution uses the MPLS header to encode the ID. The MPLS header consists of 32 bits and 20 bits are reserved for the MPLS label.

C. Computing IDs for Fixed Bit Lengths

For correctly detecting forwarding loops, each node needs a unique ID. Thus, we want to avoid that two nodes that are part of the same backup path to have the same ID. We have developed an algorithm that computes IDs for nodes in such a way that each node with ID x is maximally distant towards nodes with the same ID x. The algorithm is shown in Algorithm 1 and colors the nodes of the graph with the IDs.

First we order the nodes by descending node degree because nodes of high node degree are probably part of many paths. (1) Then, we consecutively assign the colors to the first n_b nodes. If all nodes are colored, the algorithm is finished. (2) Otherwise, we compute for each uncolored node v the color with the maximum distance. (3) We rate each color with respect to the current node v by the sum $\frac{1}{\mathrm{dist}(v_c,v)}$ for every colored node v_c. (4) We select the color v using the lowest rating which corresponds to the maximum distant color.

Note that, the proposed algorithm can be changed very easily in a SDN environment. The computation runs in a logically centralized control plane and, thus, only a few elements must be updated. In general, the distance function $\mathrm{dist}(u, v)$ is already computed in the control plane for the primary next-hops which enables more complicated algorithms as the presented for large scale networks.

V. Results

In this section we quantify the percentage of flows that LFAs can protect, cannot protect, or for which LFAs cause loops.

Algorithm 1: Color computation for restricted ID bit length.

input : $G = (V, E)$, distance function dist, and number of bits n_b
output: A node to color mapping $C : V \to \mathbb{N}$

$V_{ordered} \leftarrow$ ordered V with descending node degrees;
$C \leftarrow$ empty color mapping;
$i \leftarrow 0$;
foreach $v \leftarrow V_{ordered}$ **do**
  if $i \leq n_b$ **then** $C(v) \leftarrow i$;
  else
    $C_{\text{dist}} \leftarrow$ empty $: \mathbb{N} \to \mathbb{R}$;
    foreach *colored node* v_c **do**
      $C_{\text{dist}}(C(v_c)) \leftarrow C_{\text{dist}}(C(v_c)) + \frac{1.0}{\text{dist}(v_c, v)}$;
    end
    $C(v) \leftarrow \min_{0 \leq i \leq n_b} C_{\text{dist}}(i)$;
  end
  $i \leftarrow i + 1$;
end

The latter can be avoided through loop detection. We first describe considered failure scenarios, then the performance metrics, and the networks under study. We study the performance of different types of conventional LFAs, and compare it to the one of LFAs with loop detection. Finally, we discuss the impact of the bit length for the ID encoding in packets.

A. Failure Scenarios

A failure scenario s is a set of failed links and nodes. We define different failure scenario sets that cover different types of network failures. The set $\mathcal{S}^{l,n}$ contains all failure scenarios where l links and n nodes have failed. We analyze all single link failures $\mathcal{S}^{1,0}$, double link failures $S^{2,0}$, all single node failures $\mathcal{S}^{0,1}$, and finally the combination of single link and single node failures $S^{1,1}$.

B. Metrics

In our analysis, we route the flow for a specific failures scenario s through the network while applying certain fast reroute algorithms. We investigate if the flow (1) successfully reaches the destination, (2) is dropped because s removed all physical paths to its destination, (3) is dropped although a physical path to the destination still exists, and (4) causes a microloop. We denote flows as protected if (1) and (2) hold, as unprotected if (3) applies, and as looped if (4) holds. We consider all possible flows in the network and calculate the percentage of protected, unprotected, and looped flows for a single failure scenario s. Considering a set of failure scenarios $\mathcal{S}$, we average these values over all failures contained in that set.

C. Networks under Study

We evaluate the fast reroute mechanisms for various networks from the topology zoo [17]. We classify these topologies into three categories: star topologies T_S, ring topologies T_R, and mesh topologies T_M. Table I provides an overview of the selected networks. We omit T_S in our analysis due to the lack of alternate neighbors.

	$\lvert T\rvert$	$\lvert V\rvert$	$avg(\lvert V\rvert)$	$\lvert E\rvert$	$avg(\lvert E\rvert)$	δ
T_s	37	4 – 82	25	4 – 82	24.6	1.89
T_R	68	6 – 103	31	6 – 103	34.6	2.2
T_M	82	6 – 76	32	10 – 105	44.9	2.96

TABLE I: Statistics for the topology sets. For each topology set we provide the number of topologies, nodes, and bidirectional edges. We also provide the average node degree δ.

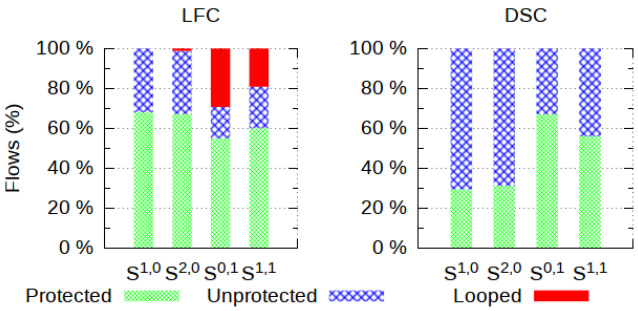

Fig. 2: Basic LFA performance on mesh topologies T_M. LFA-LFCs provide significant more coverage for link failures but cause many loops for node failures.

D. Flow Analysis for LFAs without Loop Detection

We evaluate the percentage of protected, unprotected, and looped flows for various types of LFAs and for various failures sets. As LFAs can protect significantly fewer flows in ring topologies than in mesh topologies, we conduct our study separately for mesh and ring topologies.

Figure 2 shows the average protection in mesh topologies T_M for LFA-LFCs and LFA-DSCs. For single link failures, LFA-LFCs protect approximately 68.1% which is 2.3 times as effective compared to LFA-DSCs that only protect about 29.5%.

LFA-LFCs cause loops for the other failure scenario sets. There is a similar protection ratio in $\mathcal{S}^{2,0}$ but LFA-LFCs cause loops for approximately 1.2% of the traffic. For $\mathcal{S}^{0,1}$ and $\mathcal{S}^{1,1}$, a significant number of loops occur. 29.2% of the traffic loops in single node failure scenarios and 19.1% in $\mathcal{S}^{1,1}$. LFA-DSCs protect a similar amount of traffic but cause no loops in these scenarios.

Figure 3 shows the average coverage for ring topologies T_M. The coverage of LFAs is clearly reduced for all failure scenarios for LFA-LFCs. For single link failures, 23.4% less traffic is protected which corresponds to 44.7%. The protection for double link failures is reduced from 67.2% to 59.1%. Note that, the number of caused loops is generally reduced. This is especially true for $\mathcal{S}^{0,1}$ where the amount of loops is reduced by 21.2% to 8%. We observe a reduction by 14.4% to 4.7% for $\mathcal{S}^{1,1}$. We explain this behavior due to the fact that the overall number of available LFAs is significantly smaller in

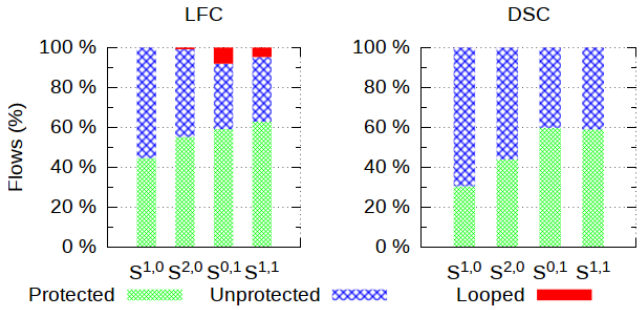

Fig. 3: Protection of LFAs in ring topologies T_R. LFA-LFCs protect significantly less traffic but causes fewer loops. LFA-DSCs provide similar performance compared to mesh topologies.

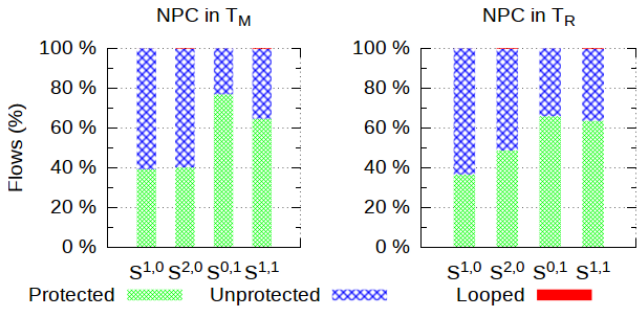

Fig. 4: Node protecting LFAs do not cause loops for single link or node failures but the protection is reduced. When multiple failures $(S^{2,0}, S^{1,1})$ occur, less than 0.5% of the traffic loops.

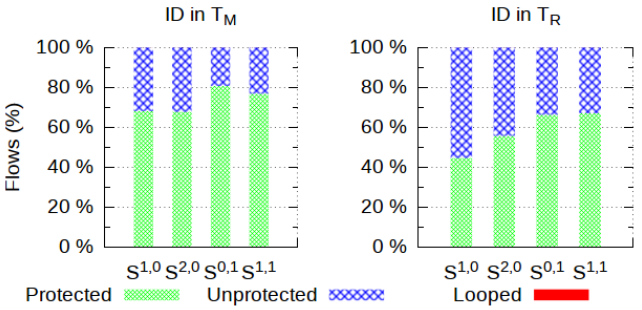

Fig. 5: Loop-detecting LFAs avoid any loops in mesh and ring topologies. Coverage against single link failures is equivalent to LFA-LFCs but is increased for node or multiple failures.

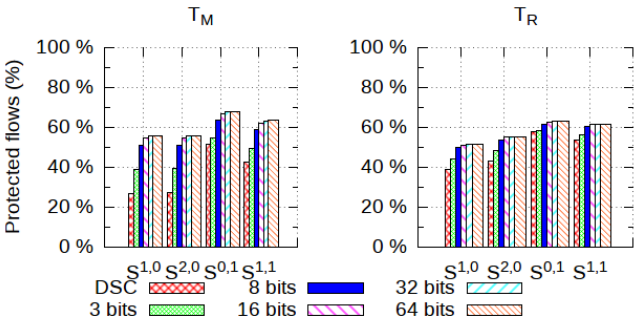

Fig. 6: Protected traffic in large mesh and ring networks for LFA-DSCs and LFA-IDs with varying bit lengths. Bit lengths of 8 or 16 are sufficient and protect comparable to optimal ID lengths in large networks.

ring structures. Note that LFA-DSCs perform very similarly in mesh topologies and ring topologies.

The protection for node-protecting LFAs is shown in Figure 4. For all topologies, we observe that loops are significantly reduced compared to LFA-LFCs: there are no loops for single link or node failures and only a minimum amount of traffic $(< 0.5\%)$ loops for multiple failures. The protection is only slightly reduced for ring topologies. However, the protection for single and double link failures is reduced to 40% by approximately 28.9% and 27%, respectively.

E. Flow Analysis for LFAs with Loop Detection

With loop detection, all LFA-LFCs can be used for fast reroute because potential loops can be detected and prevented by dropping packets. In this section, we evaluate the performance of LFAs with loop detection when unique IDs can be assigned per node to record redirecting nodes which prevents any erroneously detected loops. The results for LFA-IDs are shown in Figure 5 .

The protection against single link failures is equivalent to simple LFA-LFCs with corresponds to 68.1% in mesh and 44.7% in ring topologies. LFA-IDs prevent all loops for all network scenarios which can also be achieved with LFA-DSCs. However, LFA-IDs provide significantly more coverage. We observe an improvement of approximately 36% – 38% for single and double link failures, 13.6% for single node failures, and 20% for single link and node failures in mesh topologies. There is less improvement in ring topologies which correspond to about 6.5% – 14.2% more coverage in the different scenarios.

F. Impact of Number of Bits Available for ID Encoding

In this section we discuss the impact of the bit length for the ID encoding. The number of available bits for the ID can be a limiting performance factor in large networks. If there are more nodes than available IDs, some nodes share the same ID. If packets are redirected over a LFA-ID and traverse a node with the same ID, the packet is discarded although there is no microloop.

Therefore, we analyze the impact of the bit length on networks that consists of 50 or more nodes. There are 14 mesh and 11 ring networks of the required size in T_M and T_R. We compare LFA-DSCs and LFA-IDs with varying bit lengths. The percentage of protected flows is illustrated in Figure 6. We observe significantly more protected flows for LFA-IDs compared to LFA-DSCs even for short bit lengths. LFA-IDs protect 12.4% – 29.2% more traffic for single link failures than basic LFAs. Very short bit lengths lead to clearly

less protection than LFA-IDs with long bit lengths, i.e., a bit length of 64 protects 1.4 times as much traffic as with a bit length of 3. Bit length 8 leads to 51% and bit length 16 to 54.6% protection. The difference of bit lengths from 16 to 64 is negligible.

In ring topologies we observe the same basic trend as for mesh topologies. However, the differences between LFA-DSCs and LFA-IDs are generally less significant which can be explained by the reduced availability of LFAs in ring structures.

VI. Conclusion and Future Work

We proposed loop-free alternates (LFAs) as fast reroute algorithm for OpenFlow-based SDN. LFAs are simple and require no additional forwarding entries but have the drawback that they mostly cannot protect all traffic and some of them cause microloops in case of node failures or multiple failures. Renouncing on such LFAs reduces the protection coverage even further. To maximize protection coverage, we have suggested loop detection for LFAs and proposed how to implement it in OpenFlow in such a way that only a single additional flow entry is needed per switch. Each node requires an ID which can be encoded in the packet header to signal the failure location and each ID is encoded as a single bit in the bit string. Thus, large networks require large bit strings for loop detection. As this is not likely to scale, we suggested to share IDs among nodes to provide a fixed length label for the implementation.

We evaluated the percentage of traffic that a set of LFAs can protect, or cannot protect, or for which they cause microloops. We studied different types of LFAs and different types of network failures on multiple mesh and ring topologies. If all LFAs are used for protection, about 30% of node failures lead to extra loops in mesh networks. Allowing only those LFAs that cannot cause loops in case of node failures, reduces the protected traffic to only 40% for single link failures. With loop detection, all LFAs can be used so that 70% of the traffic can be protected in that case. This example shows the effectiveness of the proposed loop detection for LFAs in OpenFlow networks.

When IDs are shared, some nodes erroneously detect loops and drop traffic. This can happen only in case of node failures and multiple failures. Our evaluation showed that a visible amount of traffic is dropped in these cases if the bit string for the ID is 3 bits long, but bit string lengths of 16 and more bits lead to hardly any erroneous packet drops.

Loop detection just helps to prevent loops for LFAs, but it cannot protect traffic for which no LFAs exist. Therefore, LFAs with loop detection should be complemented with an additional mechanism to reach 100% protection in OpenFlow networks. They are known to achieve this goal for unit routing link costs, e.g., a network that is not optimized, but their implementation in OpenFlow is still to be done and requires more state than normal LFAs. We currently pursue this idea to provide fast protection for 100% of the traffic in OpenFlow networks with a minimum number of additional flow entries.

We will further investigate the use of LFAs for OpenFlow networks. We will integrate remote LFAs (rLFAs) in OpenFlow-based SDN which can protect 100% traffic for single link failures when the network is configured with unit link costs.We will enable the loop-detecting concept for rLFAs and investigate its protection with regard to different failure scenarios. Finally, we want to enhance rLFAs to protect against all single link failures with introducing a minimum amount of additional forwarding entries.

References

[1] 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 Communications Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014. [Online]. Available: http://www.mdpi.com/1999-5903/6/2/302

[3] ——, "Wildcard Compression of Inter-Domain Routing Tables for OpenFlow-Based Software-Defined Networking," in *European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 25–30.

[4] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "OpenFlow: Meeting Carrier-Grade Recovery Requirements," *Computer Communications*, 2012.

[5] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takàcs, and P. Sköldström, "Scalable Fault Management for OpenFlow," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 6606–6610.

[6] N. L. van Adrichem, B. J. van Asten, and F. A. Kuipers, "Fast Recovery in Software-Defined Networks," in *European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 61–66.

[7] R. M. Ramos, M. Martinello, and C. E. Rothenberg, "SlickFlow: Resilient Source Routing in Data Center Networks Unlocked by OpenFlow," in *IEEE Conference on Local Computer Networks (LCN)*, Oct. 2013.

[8] A. Atlas and A. Zinin, "RFC5286: Basic Specification for IP Fast Reroute: Loop-Free Alternates ," Sep. 2008.

[9] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote LFA FRR," http://tools.ietf.org/html/draft-rtgwg-remote-lfa, May 2013.

[10] L. Csikor and G. Retvari, "IP Fast Reroute with Remote Loop-Free Alternates: the Unit Link Cost Case," in *IEEE International Workshop on Reliable Networks Design and Modeling (RNDM)*, 2012.

[11] M. Menth and W. Braun, "Performance Comparison of Not-Via Addresses and Maximally Redundant Trees (MRTs)," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, Apr. 2013.

[12] R. Martin, M. Menth, M. Hartmann, T. Cicic, and A. Kvalbein, "Loop-Free Alternates and Not-Via Addresses: A Proper Combination for IP Fast Reroute?" *Computer Networks*, vol. 54, no. 8, pp. 1300 – 1315, Jun. 2010.

[13] P. Pan, G. Swallow, and A. Atlas, "RFC4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels," May 2005.

[14] M. Pioro, A. Tomaszewski, C. Zukowski, D. Hock, M. Hartmann, and M. Menth, "Optimized IP-Based vs. Explicit Paths for One-to-One Backup in MPLS Fast Reroute," in *International Telecommunication Network Strategy and Planning Symposium (Networks)*, Warsaw, Poland, Sep. 2010.

[15] C. R. Meiners, A. X. Liu, and E. Torng, "Bit Weaving: A Non-prefix Approach to Compressing Packet Classifiers in TCAMs," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 488–500, Apr. 2012.

[16] OpenFlow Switch Consortium and others, "Openflow switch specification version 1.1.0," 2011. [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

[17] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765 –1775, Oct. 2011.