Efficient Data Plane Protection for SDN

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

***Abstract*—Software-defined networks (SDN) usually require intervention of a controller to restore connectivity in case of link or node failures. To avoid this dependence, a fast failover function enables switches to locally detect a failure and deviate affected traffic. In this paper, we develop fast reroute (FRR) methods leveraging that feature and the principle of loop-free alternates (LFAs) from IP networks for network-wide configuration. We explain how LFAs, remote LFAs, and novel explicit-path LFAs may be implemented with OpenFlow so that SDN with general destination-based routing can be protected. An advanced loop detection and termination function is suggested to prevent loops that may be caused by FRR in case of severe failures. We evaluate the FRR methods on a large set of representative network topologies and compare them with existing FRR methods. Classic rLFAs from IP networks generate many loops in case of node failures so that they are not appropriate for SDN. One developed protection method is particularly suitable for SDN as it can protect all flows in a network against any single link and node failure. The solution is efficient because it secures most traffic also in case of dual failures, leads to short backup paths, and requires in most networks only a few additional forwarding entries per node. The latter is important as OpenFlow switches can accommodate only a moderate number of forwarding entries in their flow tables.**

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

I. Introduction

Software-defined networks (SDN) decouple the data plane from the control plane. An occurring failure needs to be detected and communicated to a controller which then reconfigures affected paths. Depending on the number of affected flows, this imposes high load on the controller in a critical situation and may take considerable time to complete. In case of in-band signalling, the network elements and the controller may be even disconnected due to the failure. Local fast-reroute (FRR) implies that a node (point of local repair, PLR) can locally detect a failure and deviate affected traffic so that it eventually reaches its destination. OpenFlow supports only a basic fast failover function that needs to be applied network-wide in an appropriate way, for which we develop and evaluate a concept in this paper.

Existing FRR mechanisms for MPLS or IP networks could be ported to SDN, but this approach suffers from two major problems. Either, existing FRR mechanisms require lots of additional forwarding entries or they cannot protect all traffic against at least single link failures (SLF) and single node failures (SNF) without creating loops. The first is prohibitive because OpenFlow switches have flow tables of only moderate size so that only a few additional forwarding entries can be accommodated for protection purposes. The second is not acceptable for SDN because unprotected flows remain disconnected or FRR-caused loops persist until the controller comes to rescue.

Coverage denotes the percentage of affected flows that can be protected against a set of considered failure scenarios. As SDN are sensitive against failures, 100% (full) coverage against all SLF and SNF is desired. That means, a link or switch may break or be unplugged from the network and the data plane remains operational for all flows without intervention of the controller. In addition, high coverage for all dual link failures (DLF) and all combinations of single link and node failures (SLF+SNF) is also desirable. FRR mechanisms with full coverage against SLF but without a limit for packet redirections may cause loops in case of node failures or multiple link failures. To prevent FRR-generated loops, loop detection and termination (LDT) is recommendable for SDN.

In this paper, we pursue the vision of a robust data plane for SDN through FRR methods with full coverage against SLF and SNF and with LDT. It should be efficient in the sense that it (1) entails only a few additional forwarding entries in the flow tables, (2) does not lead to excessive path lengths, and (3) yields high coverage for DLF and SLF+SNF. Loop-free alternates (LFAs) have been proposed for FRR in IP networks. We describe their adaptation to SDN with general destination-based routing and propose novel LFAs with explicit paths (eLFAs). We develop an advanced LDT (ALDT) mechanism that avoids erroneous packet drops after first redirect of existing solutions. We devise various protection methods on that base and explain how the discussed features may be implemented with OpenFlow. We define metrics to measure coverage, path length, and additional forwarding entries, and compare various existing and novel FRR methods on a large set of representative network topologies. The results reveal that one of our proposed methods meets the goals and the discussion clarifies that the use of other standardized FRR methods in SDN would be inferior.

In Section II we review related work regarding FRR methods for IP and MPLS networks as well as existing approaches to protect the SDN data plane. Section III gives a more detailed overview of available LFA variants and an LDT scheme. In

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/1. The authors alone are responsible for the content of the paper.

Section IV we suggest novel LFA-based protection methods for SDN and ALDT, and show how they can be implemented in OpenFlow. Section V introduces our methodology and performance metrics. We present performance results in Section VI discuss them in Section VII and point out issues for further research. Finally, we summarize the work in Section VIII and draw conclusions.

II. Related Work

FRR methods for MPLS and IP networks [1], [2] have been an active research area over the last two decades and many standards have evolved or are still under review. We survey them and report activities to protect the SDN data plane.

MPLS-FRR offers one-to-one backup and facility backup with link and node protection [3]. One-to-one backup deviates traffic from the PLR to the destination bypassing the potential failure along an alternative preconfigured path. Multiple paths towards the same destination may be merged to reduce state overhead. With facility backup, traffic is tunneled around a link to its next-hop or around a node to its next-next-hop depending on whether link or node protection is desired.

For IP networks, not-via addresses [4] inject for every unidirectional link in the network an additional entry in the routing tables. These addresses are used to tunnel traffic around a failed link or node along shortest paths leading to a similar path layout as MPLS FRR facility backup with shortest paths [5]. With loop-free alternates (LFAs), a PLR reroutes traffic to a neighbor [6] which avoids loops. They do not induce additional forwarding entries in routing tables but have limited coverage. Remote LFAs (rLFAs) [7]–[9] complement the coverage of LFAs by forwarding traffic to remote next-hops using tunnels, but do not require additional entries, either. We describe (r)LFAs in more detail in Section III. Coverage of LFAs can be further increased by optimizing link costs [10] and coverage of (r)LFAs was extended by adding links to the network [11]. A self-configuring extension of LFAs [12] uses probes to configure LFAs, and increases flow coverage by installing alternative hops in other nodes than the PLR that are activated in failure cases to avoid loops. With segment routing (SR) [13], forwarding state in nodes may be traded against forwarding state in packet headers. In that context, topology-independent LFAs (TI-LFAs) [14] provide explicit-path tunnels when rLFAs are not available. They are conceptually similar to our proposed eLFAs but rely on multiple labels added to the packet. SR may also be used in IP networks to avoid microloops that may occur after a failure during the rerouting process [15]. Such loops can be avoided in SDN by appropriate link update orders. Failure insensitive routing (FIR) [16] utilizes interface-specific tables to encode failure information. Depending on the interface the packet is forwarded to a pre-computed failure avoiding backup path. Multiple routing configurations (MRCs) [17] leverage multiple routing topologies such that at least one is working in case of a failure. A packet affected by a failure is pinned to a working topology. This approach at least doubles the number of forwarding entries. Maximally redundant trees (MRTs) [18] compute in a distributed manner a set of red and blue forwarding entries such that at least one of them works in case of a failure, i.e., full coverage is provided. Affected traffic is pinned to the appropriate color and forwarded accordingly. MRTs triple the number of forwarding entries of conventional IP networks. MRTs may lead to long backup paths [19]. Combining MRTs with LFAs may reduce backup path length and link load in case of failures [20]. The IETF standardizes (r,TI-)LFAs and MRTs. Independent Directed Acyclic Graphs (IDAGs) [21] compute disjoint paths to guarantee protection against any single link or node failure with only two forwarding entries in every node per destination.

While many efforts were made to protect against failures of the SDN control plane [22], there are only a few advances securing the SDN data plane. The controller may reconfigure the network in case of a network failure. The authors of [23] measured that their controller reacts within 80 – 100 ms in their testbed. They also underlined that the restoration time depends on the number of flows to be restored, path lengths, traffic bursts in the control network, and is likely to take an order of magnitude longer for larger networks. Similar result are observed for centralized IP routing [24]. OpenFlow 1.1 provides a fast-failover action which was not available before. The authors of [25] developed a similar local failover scheme based on MPLS-TP. They added a bidirectional forwarding detection (BFD) component to an OpenFlow switch to locally detect failures and switch packets to other interfaces without intervention of the controller. For the Open vSwitch, link failures can be detected within 3 – 30 ms depending on BFD configuration [26]. SlickFlow encodes primary and backup paths in packet headers [27]. Its viability was shown through a prototype in a virtual testbed. SPIDER proposes an alternative to OpenFlow's fast failover action based on additional state in the OpenFlow pipeline [28]. SPIDER leverages packet labels to carry reroute and connectivity information. The path layout is inspired by MPLS crankback routing and can be optimized. In [29] we leveraged the idea of normal LFAs together with a simple LDT scheme which may erroneously drop packets after first redirect for FRR in SDN. This mechanism cannot protect against all single link failures. [30] encodes either a failed link or node in a packet label, but protects only against single failures. MRCs have been applied to SDN in [31].

III. State of the Art: Loop-Free Alternates

We explain LFAs and rLFAs in more detail. We report their naive application to SDN including a simple LDT mechanism.

A. Loop-Free Alternates

IP networks leverage shortest path routing based on a single set of link weights to avoid loops. In that context, LFAs have been proposed for FRR [6]. In Figure 1, the node P acts as PLR, i.e., it detects that the normal next-hop NH towards destination D is not reachable and locally redirects the traffic via the precomputed neighbor PQ_0. This neighbor is called a loop-free alternate (LFA) and must be chosen such that loops are avoided when the redirected traffic is further forwarded

on shortest paths towards its destination. Thus, LFAs are destination-specific.

Not all neighbors of a PLR may be used as LFAs because some cause loops for specific destinations when used for traffic redirection. LFAs reveal different protection and loop avoidance capability. We reflect that by notation if needed, e.g., {LP,NP}-{DS,nDS}-LFAs, and explain it in the following. Some LFAs deviate traffic around an unreachable next-hop and provide node protection (NP). Others deviate traffic only around the link towards an unreachable next-hop such that the deviated traffic is forwarded via the next-hop further downstream. Such LFAs provide only link protection (LP). Thus, we distinguish LP- and NP-LFAs. LFAs may cause loops in case of a node failure (e.g., LP-LFAs) or in case of multiple failures. LFAs closer to the destination than the PLR are denoted downstream (DS) LFAs (DS-LFA), others are called nDS-LFAs (non-DS). Loops can be avoided by using only DS-LFAs. The authors of [5] illustrate LFAs for any combination of {LP,NP}-{DS,nDS}-LFAs. Several papers have shown that the coverage by LFAs may be rather low [10], [11], [32] and that a significant amount of loops may occur in case of node failures [29].

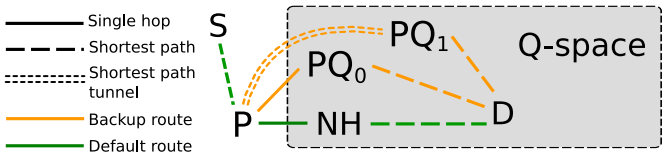


Fig. 1. A normal LFA (PQ_0) is a neighboring node of the PLR P while a remote LFA (PQ_1) is reachable only via a tunnel. Traffic sent to (r)LFAs reaches the destination D without returning to the PLR.

B. Remote LFAs (rLFAs)

Remote LFAs (rLFAs) [7]–[9] complement the coverage of LFAs. Figure 1 illustrates the concept. In case of a failure, the PLR tunnels affected traffic to another non-neighboring node PQ_1 along shortest paths. From PQ_1, the traffic is carried along shortest paths to its destination. PQ_1 is denoted an rLFA. rLFA computation methods are proposed in [7], [8], [9]. First, LP or NP must be chosen as requirement. In case of LP, the so-called P-space comprises all nodes that are reachable from the PLR via shortest paths without traversing the link to the unreachable next-hop NH. In case of NP, NH must not be traversed. The Q-space is the set of nodes that can reach the desired destination D via shortest paths without traversing the link between PLR and NH if LP is required; otherwise the traffic must not traverse NH. The intersection of the P-space and the Q-space yields potential PQ nodes. It is recommended to choose the PQ node closest to the PLR and use a tie-breaker if needed [7, Sect. 5.2.2]. As the set of PQ nodes is smaller when NP is required instead of LP, it is more likely to find an rLFA for LP than for NP. The Q-space may contain reachable neighbors of the PLR; they may be used as normal LFAs because the PLR does not need a tunnel to redirect traffic to them. Thus, the presented method computes both LFAs and rLFAs. The Q-space may be approximated to save computation time [7] which may reduce its size in a few networks.

The extended P-space contains also those nodes that are reachable from all reachable neighbors of the PLR. The authors of [9] have shown that rLFAs based on the extended P-space provide full protection against SLF if shortest path routing in the network uses unit link costs. This result does not hold for networks with non-unit link costs.

Also rLFAs can be categorized into DS and nDS, but DS-rLFAs cannot prevent loops. A simple counterexample is a ring with two failures with a packet addressed to the other side of the ring. The neighbors of the failures may try to change the direction of the packet through rLFAs, which leads to a loop.

C. Simple Application of LFAs to SDN with Loop Detection and Termination (LDT)

In [29] we apply LFA principles to SDN. We assume that all traffic in the network is addressed to edge nodes. This limits the set of potential destination addresses so that one proactive forwarding entry per edge node is sufficient. OpenFlow's fast failover function is used for traffic redirection to LFAs if the next-hop is not reachable. The path layout for all flows followed shortest paths based on a single set of link costs and LFAs were chosen accordingly. This mechanism cannot protect against all single link failures. Furthermore, a LDT mechanism was defined and an implementation for OpenFlow 1.1 was suggested. Every node in the network is associated with an ID. A loop detection (LD) label (e.g., MPLS label) is added to a packet containing a bit string where each bit represents one node ID. If a node redirects a packet using an LFA which may cause a loop, it sets its own ID in the LD label. If a node receives a packet with its own ID activated in the LD label, it drops the packet as it obviously caused a loop. This behavior can be implemented with only a single additional flow entry per node. In large networks, the LD label size may not suffice to represent the IDs of all nodes. Therefore, nodes are partitioned into ID sets and every node in an ID set is associated with the same ID. With this workaround, nodes drop packets from preceding upstream PLRs if they belong to the same ID set. This can be considered as a false positive which may occur already after a first redirect. The authors showed that these erroneous packet drops are rather rare if the LD label is large enough and the IDs are assigned appropriately to the nodes. Nevertheless, erroneous packet drops are an undesired deficiency.

IV. LFA-Based Fast Reroute for SDN

We leverage LFA principles to protect general destination-based forwarding in SDN, i.e., a restriction to shortest paths based on a single set of link costs is not needed. We present advanced LDT (ALDT) to stop potential loops caused by (r)LFAs; in contrast to existing LDT, ALDT avoids erroneous packet drops after first redirect. We propose rLFAs with explicit-path tunnels (eLFAs) to complement existing (r)LFAs in order to achieve 100% coverage against SLF and SNF. We define various LFA-based protection methods for SDN that differ in complexity and coverage. Finally, we explain how the proposed mechanisms can be implemented with OpenFlow.

A. (r)LFA Protection for SDN with General Destination-Based Forwarding

We first explain how LFAs may be leveraged for protection of SDN and then we consider rLFAs for that purpose.

1) Application of LFAs to SDN: With destination-based routing, the set of paths from all sources to a destination forms a destination-based tree with explicit paths. Each such tree can be represented as shortest paths tree with destination-specific link costs. The cost of a link in the tree is set to 1 while other link costs are set to the number of nodes. These link costs are used for that destination's LFA computation. This yields a path layout that avoids loops because traffic follows the shortest paths of its destination's set of link costs before and after redirection.

2) Application of rLFAs to SDN: Application of rLFAs is similar, but the extended P-space for rLFA construction must be defined in a different way. It contains all nodes N that may be reached from the PLR or from its reachable neighbors over the destination-specific tree towards N without traversing the potentially failed link or node, respectively. This ensures that if a PLR redirects traffic towards a node PQ by means of encapsulation, the traffic eventually reaches that node PQ by following the PQ-specific shortest paths. Node PQ decapsulates the traffic so that the traffic is carried from PQ to its destination D via D-specific shortest paths.

B. Advanced Loop Detection and Termination (ALDT)

The simple LDT may erroneously drop packets after first redirect. We present advanced LDT (ALDT) which may erroneously drop packets only after second redirect so that it avoids dropping traffic erroneously in any single failure scenario. With ALDT, any redirecting node conceptually activates its ID in the LD label unless the packet was redirected for the first time. As OpenFlow does not provide a feature for this behaviour, we propose a supported workaround in Section IV-E4.

C. rLFAs Using Explicit-Path Tunnels (eLFAs)

Some destinations cannot be protected with (r)LFAs, e.g., in the presence of non-unit link costs or when NP is required. If (r)LFAs with desired properties towards a destination D cannot be found, we propose to set up an explicit-path tunnel from the PLR to a closest node eQ in D's Q-space. In case of a failure, the PLR can redirect the traffic destined towards D via the explicit-path tunnel to eQ from where it reaches its destination D. Such an explicit-path exists if PLR and D are still connected in the network. We denote such LFAs as eLFAs. Their path layout depends on protection requirements: while an NP-eLFA tunnel does not traverse the PLR's next-hop NH for destination D, an LP-eLFA tunnel does not traverse just the link PLR–NH between PLR and NH. Explicit-path tunnels require additional flow entries in any node along the explicit path. Therefore, eLFAs are more complex than rLFAs.

D. LFA-Based Protection Methods for SDN

We define protection methods specifying which types of LFAs (LFA, rLFA, eLFA) may be considered for precomputation and configuration in potential PLRs. Moreover, they determine an order of preference if multiple LFAs with different properties are available. We first explain how LFAs and rLFAs are computed for IP networks and denote these protection methods as classic. Then, we propose protection methods with loop avoidance (LA).

1) Classic Protection Methods: We differentiate two classic (C) protection methods that leverage only normal LFAs or both normal LFAs and rLFA. We call them LFA-C and rLFA-C, respectively. For computation of rLFA-C, we leverage the method reported in Section III-B and choose LP and the extended P-space to maximize coverage. If this yields multiple PQ nodes, we prefer a PQ node closest to the PLR and the lowest node ID for tie breaking. This selection gives preference to normal LFAs. LFA-C is computed accordingly but only normal LFAs are used.

2) Protection Methods with Loop Avoidance (LA): In an SDN context, LFA precomputation can be carried out by a server with sufficient resources. This facilitates a classification of available (e,r)LFAs into LP and NP, DS and nDS. We leverage this classification for a ranking-based selection of LFAs that maximizes coverage while avoiding loops if possible. We explain the ranking. Our first goal is to maximize protection, our second criterion is operational simplicity, and our third criterion is loop avoidance. Therefore, we prefer NP-(e,r)LFAs to LP-(e,r)LFAs in the first place, LFAs to rLFAs and rLFAs to eLFAs in the second place, and DS-LFAs to nDS-LFAs in the third place. Table I compiles the resulting ranking of LFA types by protection capability, operational simplicity, and loop avoidance capability. As an NP-(r,e)LFA bypasses a potentially failed next-hop, only LP-(r,e)LFAs can be used to protect a last link of a flow's path. Therefore, LP-(r,e)LFAs are also needed in any network in spite of their lower protection capability.

TABLE I
RANKING OF LFA TYPES ACCORDING TO PROTECTION CAPABILITY, OPERATIONAL SIMPLICITY, AND LOOP AVOIDANCE CAPABILITY.

Rank	LFA type	Rank	LFA type	Rank	LFA type
1	NP-DS-LFA	4	NP-eLFA	7	LP-rLFA
2	NP-nDS-LFA	5	LP-DS-LFA	8	LP-eLFA
3	NP-rLFA	6	LP-nDS-LFA		

In the following we define four different protection methods that leverage different types of LFAs. When multiple LFAs are available, the ranking from Table I is applied to select a best candidate, IDs are used for tie-breaking.

a) LFA-LA: Only normal LFAs may be used. This method has been proposed in connection with simple LDT and was studied for datacenter networks in [29].

b) rLFA-LA: Normal LFAs and rLFAs may be used.

c) eLFA-LP-LA: Normal LFAs, rLFAs and eLFAs with tunnels for LP may be used.

d) eLFA-NP-LA: Normal LFAs, rLFAs and eLFAs with tunnels for LP and NP may be used.

Our goal is full coverage against SLF and SNF and we do not want to limit how often a packet can be redirected. Under such conditions, loops occur if the destination of a flow fails because then packets are redirected without a chance to

reach the destination. To avoid such loops, we combine these methods with ALDT. Figure 2 summarizes properties of the protection methods that derive from the properties of involved LFA types.

Protection profile	LFA-C	rLFA-C	LFA-LA	rLFA-LA	eLFA-LP-LA	eLFA-NP-LA
Loop avoidance			●	●	●	●
Full coverage against SLF		o		o	●	●
Full coverage against SNF						●
Additional entries					●	●
Known as	Classic LFAs	Classic rLFAs	LD-LFAs			

Fig. 2. Properties of LFA-based protection methods.
Legend: o = only for unit link costs; ● = independent of link costs.

E. Implementation of LFA-Based Protection Methods

We first describe how normal LFAs, rLFAs, and eLFAs can be supported with OpenFlow. Then, we suggest an implementation of ALDT for normal LFAs and LFAs using tunnels.

1) Implementation of LFAs: OpenFlow uses a forwarding pipeline consisting of several tables. A flow table matches incoming packets against flow entries. OpenFlow 1.1 [33] introduces group tables with fast failover actions. For the implementation of an LFA, an entry in the flow table needs to point at an entry in the group table with the group type fast-failover. This entry contains a list of so-called action buckets with forwarding actions for the primary next-hop and one or more secondary next-hops (LFAs). If the primary next-hop works, it is selected for packet forwarding, otherwise the next working secondary next-hop is selected.

2) Implementation of rLFAs: rLFAs leverage tunnels to deliver traffic towards other nodes along shortest paths. As tunnel endpoints are existing IP addresses, additional flow entries along the tunnels are not needed. However, OpenFlow does not provide actions for encapsulation with an additional IP header. We point at two workarounds for this shortcoming.

A first option is to create for each required tunnel endpoint an additional interface on the switch to perform the desired encapsulation. The controller can leverage the OF-CONFIG protocol [34] to set up these tunnel interfaces.

A second option is to extend the OpenFlow protocol and the switches with appropriate actions for IP encapsulation and decapsulation. This has been implemented in [35]. It allows the controller to use OpenFlow instead of OF-CONFIG to configure the IP tunnels.

3) Implementation of eLFAs: eLFAs leverage tunnels to deliver traffic to other nodes along explicit paths. We suggest to encapsulate corresponding traffic with a new IP address that is specific to the explicit path of the eLFA. Moreover, the controller installs additional forwarding entries for this IP address on the nodes along the explicit path. Tunneling may be performed like for rLFAs.

4) Implementation of ALDT: We explain why ALTD cannot be implemented with OpenFlow in a straightforward way. We present a conceptual workaround, describe how it can be implemented for normal LFAs with OpenFlow and extend it for LFAs using tunnels.

a) Limitation of OpenFlow for Implementation of ALDT and Conceptual Workaround: The ALDT mechanism uses an LD label containing IDs to indicate whether a node of a specific ID set already redirected the packet. Any redirecting node activates its ID in the LD label unless the packet was redirected for the first time. However, this behaviour cannot be implemented with current versions of OpenFlow because conditional setting of header values are not supported at the group table stage or later in the pipeline.

We propose a solution pursuing the following idea. A redirecting node sets the ID in the LD label if the corresponding LFA may cause a loop. The next-hop immediately clears this ID if the packet was redirected for the first time. Other nodes drop the packet if they find their own ID in the LD label. Thus, deactivating the ID immediately after first redirect guarantees that a packet with at most one redirection is erroneously dropped by a downstream node with the same ID.

b) ALDT Implementation for Normal LFAs: We define the LD label containing a potential-loop bit (PLB), an auxiliary PLB (APLB), and a field for ID encoding. When an IP packet enters the SDN domain, the LD label is pushed with all bits set to zero. If a node redirects a packet using an LFA that may cause a loop, it activates the APLB and its own ID in the LD label. If the next-hop detects the PLB and APLB set to '01', the packet was redirected for the first time. Then, all IDs are cleared and the PLB is activated. To detect and stop potential loops, nodes check whether the PLB and their own ID are set in the LD label of a packet and drop it in that case.

We propose an MPLS label as LD label which provides 20 bits for PLB, APLB, and ID field. MPLS labels belong to the optional features of OpenFlow 1.1 – other implementations may be possible as well (see [29]). To support the proposed behavior, the OpenFlow packet processing pipeline may be configured with an MPLS table as a first stage. It is used to match on a packet's LD label in order to possibly modify the LD label using a set-field action or to drop the packet after loop detection. The next stage is a matching IP table to support packet forwarding followed by a group table with the fast-failover instructions. The latter modify the LD label using a set-field action in case of a redirect.

c) ALDT Implementation for LFAs Using Tunnels: To support rLFAs and eLFAs, ALDT must cope with tunneling. We propose two different solutions which depend on switch capabilities. The first solution requires tunnel interfaces that insert the encapsulating IP header between the existing IP header and the MPLS label. In that case, no changes to the previous description are needed. This is the preferred solution and the base for the coverage analysis in the remainder of the paper. The second solution may be used if the above switch behaviour is not available. We propose that the PLR encapsulates the IP/MPLS packet with an additional IP/MPLS header. An unused bit of the DSCP/ECN field in the encapsulating IP header should be activated to mark that the packet belongs to a tunneling LFA (rLFA, eLFA). We call it tunneling-LFA bit

(TLB). Furthermore, the PLB and the ID for the PLR should be activated in the LD label of the outer MPLS header upon encapsulation. This helps the PLR to detect if it receives the packet again and drop it. As this variant ignores all previous redirects in the LD label of the outer MPLS header, further redirects of the packet must be avoided, leading to slightly lower coverage for multiple failures than the first variant. To enforce this behavior, nodes require an additional rule to drop traffic with both an activated TLB and activated APLB.

V. Evaluation Methodology

We explain the methodology to evaluate LFA-based protection methods. We describe the general approach, the metrics of interest, the studied topologies, and the generation method for non-unit link costs.

A. General Approach

Although the proposed protection methods can protect general destination-specific forwarding by design, we focus in our evaluation only on shortest path routing based on a single set of link costs to reduce the parameter space. Thus, for a given network topology including link costs, we precompute LFAs for a specific protection method using the shortest path principle. In case of equal-cost paths, we use the node ID as a tie breaker to determine consistent single paths. We consider SLF, SNF, DLF and SLF+SNF as potential failure scenarios $\mathcal{S}$. We consider all source-destination pairs $\mathcal{F}$ and call them flows. For each failure $s \in \mathcal{S}$, we calculate the path layout for each flow $f \in \mathcal{F}$ and analyze it.

B. Performance Metrics

Based on the path layout and installed eLFAs, we determine coverage, path lengths, and additional entries.

1) Average Coverage: We consider only flows whose path is affected by a failure. A flow is working (+) if it has working source and destination nodes and a working path between them can be found in the network. The traffic of a working flow may arrive (+a), loop (+l), or be dropped (+d), depending on how well the applied protection method works. A flow is failed (-) if source or destination is down or if there is no working path between them due to a failure. Then, traffic may loop (-l) or be dropped (-d). We consider the flow states (+a) and (-d) as protected. We consider the flow state (+d) as unprotected, and the flow states (+l) and (-l) as looped. To characterize the coverage, we classify all affected flows for all scenarios $s \in \mathcal{S}$ as protected, unprotected, or looped. For a given protection method and set of failure scenarios $\mathcal{S}$, we calculate coverage values averaged over all affected flows in a network and all failures $s \in \mathcal{S}$.

2) Maximum Path Lengths: Based on the path layout for a protection method we derive the length of all affected but working flows for which traffic can be successfully forwarded. We consider only the longest path length of a flow in any $s \in \mathcal{S}$. We compute the average and maximum path length of all considered flows.

3) Additional Entries: Let n be the number of nodes in the network. To support destination-based forwarding, any node requires $n - 1$ forwarding entries. eLFAs leverage explicit-path tunnels which impose additional forwarding entries in the nodes along their paths. The number of additional entries is node-specific. Therefore, we compute both the average and the maximum percentage of additional entries over all nodes.

C. Network Topologies under Study

We chose for the evaluation 205 wide area, research, and commercial networks from the Topology Zoo [36] as well as three datacenter topologies (fat-tree, DCell, BCube) that have been studied in [29]. To deliver a compact view on the performance results from all network topologies, we report average values for coverage and complementary cumulative distribution functions (CCDFs) for path lengths and additional entries.

D. Non-Unit Link Costs

Link costs have a significant impact on coverage of LFA-based protection methods [9]. Therefore, we perform experiments with unit link costs and non-unit link costs. However, the Topology Zoo lacks both link costs and real traffic data [36]. For that reason we construct non-unit link costs according to the recommendation in [37]. We derive the link-specific load imposed by a homogeneous traffic matrix for shortest-path routing and unit link costs. The link cost is the inverse of the resulting link load multiplied by the largest link load so that the smallest link cost is 1. This algorithm results into link costs with average mean of 6.4 and average coefficient of variation of 1.0 over all 208 topologies. Hence, the method produces significantly different non-unit link costs.

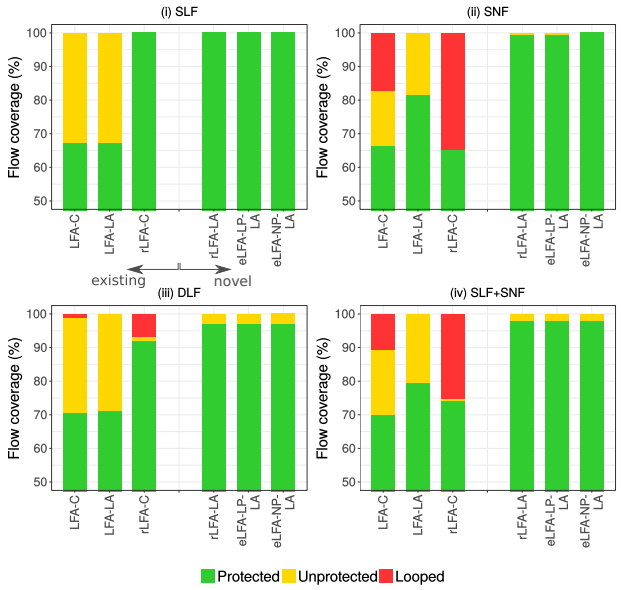
VI. Performance Evaluation

We compare the LFA-based protection methods for SDN of Section IV-D with regard to coverage, path length, and additional entries using the methodology presented in Section V.

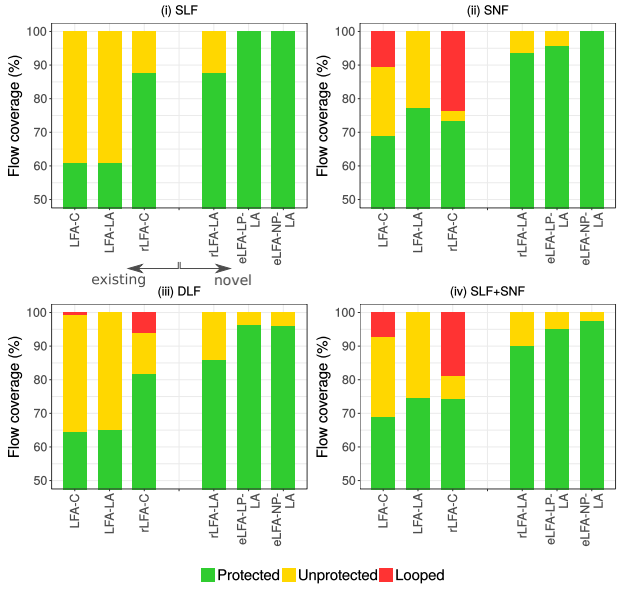
A. Coverage

We first consider unit link costs. Figure 3(a) illustrates average coverages from 208 topologies for different protection methods and various sets of failure scenarios. Subfigure (i) shows that the existing methods LFA-C and LFA-LA cannot protect 32% of the flows. In contrast, rLFA-C and the novel methods rLFA-LA and eLFA-{LP,NP}-LA achieve full coverage. This result for rLFA-C and unit link costs has been proven in [9].

Subfigure (ii) reports for the classic methods LFA-C and rLFA-C lots of loops (17% and 34%) in case of SNF, which is rather unsatisfactory in an SDN context. As rLFA-C provides full coverage against SLF and does not limit the number of redirects per packet, it causes loops when a flows's destination fails. This finding is also relevant for IP networks. For LFA-LA we observe more protected traffic for SNF than for SLF which seems counterintuitive. However, SNF affect more flows than SLF so that the comparative base is different. Moreover, LFA-LA terminates flows with failed destinations instead of creating



(a) Unit link costs.



(b) Non-unit link costs.

Fig. 3. Coverage averaged over 208 topologies depending on protection method and set of failure scenarios.

loops. These flows count as protected rather than looped, which also increases the coverage of LFA-LA compared to LFA-C. Apart from that, these two methods have the same amount of unprotected traffic. Both rLFA-LA and eLFA-LP-LA avoid loops and yield only a minor fraction of less than 1% unprotected flows. The novel eLFA-NP-LA is the only method protecting all traffic by design. Without ALDT, (r)LFA-LA and eLFA-{NP,LP}-LA would suffer from a very similar amount of loops as rLFA-C.

Subfigures (iii) and (iv) show that no protection method achieves full coverage for DLF and SLF+SNF, but rLFA-LA and eLFA-{LP,NP}-LA come close with 2–3% unprotected traffic although they are not designed for that purpose.

Figure 3(b) provides corresponding results for non-unit link costs. Subfigure (i) shows that LFA-C, rLFA-C, LFA-LA, and rLFA-LA protect less traffic than for unit link costs, in particular rLFA-{C,LA} lose full coverage against SLF, which is significant and has not been quantified before. In contrast, eLFA-{LP,NP}-LA still achieve full coverage by design. Subfigure (ii) reveals that for SNF the fraction of unprotected traffic with rLFA-LA and eLFA-LP-LA is still small but visibly larger than for unit link costs. eLFA-NP-LA achieves again full coverage against SNF by design. Subfigure (iii) and (iv) reveal that with non-unit link costs more traffic remains unprotected in the presence of multiple failures than with unit link costs.

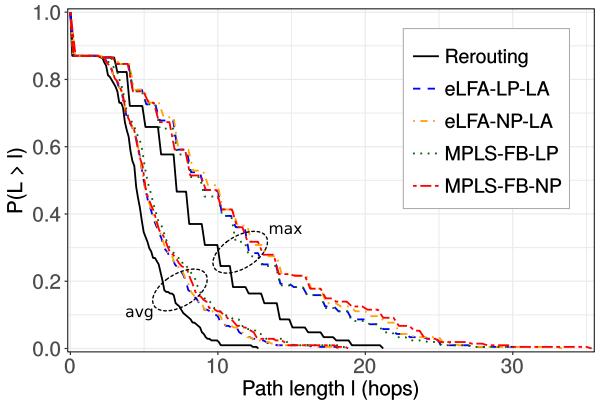


Fig. 4. CCDF of average and maximum path lengths for different protection methods, unit link costs, and SLF.

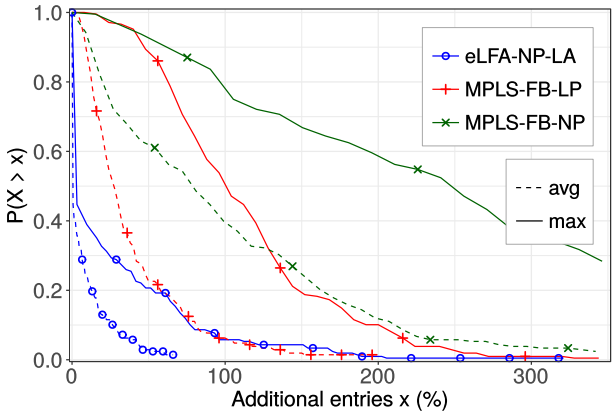
B. Path Lengths

Figure 4 illustrates the CCDF of the average and maximum path lengths of affected flows observed over 208 topologies in case of SLF. For a meaningful comparison, we consider only protection methods with full coverage against SLF. Rerouting provides a lower bound for path length as it reestablishes shortest paths after a failure. MPLS facility backup with LP and NP (MPLS-FB-{LP,NP}), respectively, are widely used FRR methods and also serve for comparison [3]. MPLS-FB-NP yields slightly longer maximum path lengths than MPLS-FB-LP in most networks while it is vice-versa for the average path length. Differences between the two methods are small for both metrics. eLFA-{LP,NP}-LA lead to very similar average values that are slightly lower than those of MPLS-FB-NP. Maximum path lengths for eLFA-LP-LA are smaller than or equal to those of eLFA-NP-LA and both are very similar to those of MPLS-FB-{LP,NP}. The presented results are derived for unit link cost networks. Under such conditions, rLFA-LA yields exactly the same paths as eLFA-LP-LA so that corresponding curves are omitted. Path lengths for non-unit link costs are slightly longer than those of unit link costs. As the results do not provide any further insights, we omit corresponding figures.

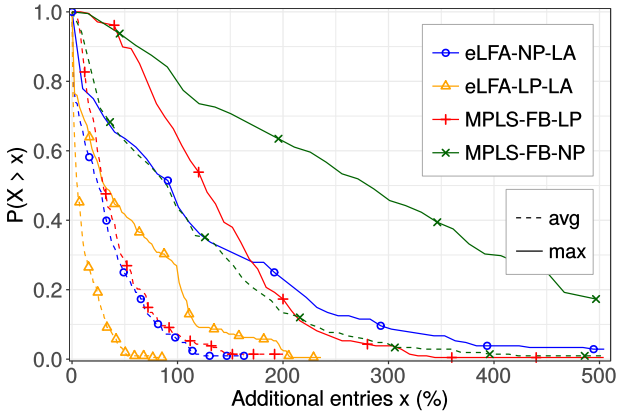
C. Additional Entries

With destination-based routing, any node in the network requires one entry for each other node in the forwarding table.

The protection methods eLFA-{LP,NP}-LA require additional entries. Figures 5(a) and 5(b) present CCDFs of the average and maximum percentage of additional entries per node, evaluated over all investigated networks. The results are very topology-dependent.



(a) Unit link costs.



(b) Non-unit link costs.

Fig. 5. CCDFs for additionally needed forwarding entries.

Figure 5(a) shows additional entries for unit link costs. We omitted curves for eLFA-LP-LA because it does not induce any additional entries for unit link costs. The comparable MPLS-FB-LP requires in 22% of the networks 50%–200% additional entries per node on average, 50% of the networks have at least one node with more than 100% additional entries, and 9% a node with even more than 200% state overhead.

eLFA-NP-LA leverages NP-eLFAs to provide full coverage against SLF and SNF. Nevertheless, more than 56% of the topologies do not need explicit-path tunnels, 97% require less than 50% additional entries on average, and the average state overhead does not exceed 66%. Also maximum numbers of entries are very low: 80% of the networks require at most 50% more entries per node and only 7% of the networks exhibit at least one node with more than 100% additional entries. The comparable MPLS-FB-NP method imposes in 40% of the networks more than 100% state on average per node, in 12% even more than 200%. More than 75% of the networks have at least one node with 100% and more overhead, and 57% have a node with a state overhead between 200% and 500%.

Figure 5(b) reports additional entries for non-unit link costs. With non-unit link costs, eLFA-LP-LA install indeed a few explicit-path tunnels but the average number of additional entries remains below 50% for almost all networks. Only 25% of the networks exhibit a node with more than 100% additional entries. This is very little compared to MPLS-FB-LP which causes slightly more additional entries for non-unit link costs than for unit link costs. eLFA-NP-LA requires roughly three times more additional entries than with unit link costs. But still, 94% of the networks require less than 100% additional entries on average and only 24% of the networks exhibit a single node with more than 200% additional entries. MPLS-FB-NP needs only slightly more additional entries compared to unit link costs. This is again a multiple of the required additional entries needed by eLFA-NP-LA.

VII. Discussion

In Section IV we proposed novel LFA-based protection methods and investigated them with some existing FRR methods in Section VI. In the following, we discuss main results in a wider context and point out topics for further research.

A. Summary

We evaluated the different LFA-based protection methods on a large set of 208 representative topologies using unit and non-unit link costs with regard to coverage, path lengths, and additional entries. The classic use of LFAs and rLFAs (LFA-C, rLFA-C) cannot protect all traffic in case of non-unit link costs and generally suffers from a large fraction of loops or unprotected traffic in case of SNF. Therefore, they cannot sufficiently protect SDN. However, a combination of LFAs, rLFA, and eLFAs (eLFA-NP-LA) seems appropriate for that purpose. It provides full coverage against all SLF and SNF for unit and non-unit link costs and protects a very large fraction of affected traffic in case of multiple failures. Potential loops are reliably detected and terminated. eLFA-NP-LA requires less than 50% additional entries in most networks. MPLS-FRR facility backup imposes significantly more additional entries (70%–200%) in most networks. IDAGs cause 100% additional rules. MRTs require 200% more state and not-via addresses require $d \cdot 100\%$ more state where d is the average node degree. eLFA-NP-LA does not extend path lengths compared to MPLS-FRR facility backup which has the same path layout as not-via addresses while MRTs lead to even longer paths [19]. Finally, MPLS-FRR facility backup, not-via addresses, and MRTs cannot protect traffic on backup paths as long as it is tunneled or marked. This is different for LFA-based methods by design. Thus, eLFA-NP-LA can efficiently protect SDN with regard to additional entries, backup path lengths, and coverage even against multiple failures.

B. Future Work

We proposed several attractive LFA-based FRR protection methods for SDN. A next step is the deployment of the proposed mechanisms on a real SDN testbed, preferably in OpenFlow due to its wide adoption, possibly also in P4.

The optimization potential for rLFAs and eLFAs may be studied. rLFAs and eLFAs may be chosen to minimize path length rather than distance to the PLR. If several PLRs use explicit-path tunnels towards the same eLFA, the tunnels may be merged to save additional entries. The potential for such savings can be increased by choosing eLFAs that can be shared by many PLRs and for different destinations. The traffic engineering potential to avoid overload on backup paths may be investigated.

ALDT supports multiple packet redirects, which may be beneficial in the presence of multiple failures. However, packets may be dropped only when traversing a node for the third time. This may consume too much bandwidth in case of a failure. ALDT can also be adapted to drop packets after second redirect. Both approaches may be studied taking traffic rates and link bandwidths into account.

VIII. Conclusion

In this work, we proposed how local fast failover functions may be used in software-defined networks (SDN) to restore connectivity without intervention of the controller in case of all single link and node failures if topologically feasible. This is challenging as SDN switches can accommodate only a moderate number of forwarding entries in their flow tables.

We showed how the concept of (remote) loop-free alternates ((r)LFAs) from IP networks can be applied for fast reroute (FRR) in SDN that use general destination-based forwarding. We suggested explicit-path LFAs (eLFAs) to increase flow coverage against failures. We proposed rank-based selection of (e,r)LFAs to maximize coverage and minimize operational complexity. To avoid FRR-caused loops, we developed an advanced loop detection and termination (ALDT) function. We explained how these features may be implemented in OpenFlow and used them to define new protection methods. They differ in coverage and operational complexity and leverage ALDT for loop avoidance (LA).

We evaluated these novel and some existing FRR methods on a set of 208 representative topologies with unit and non-unit link costs. Classic rLFAs protect against all single link failures, but require unit link costs for that purpose and are likely to create loops in case of node failures. In contrast, the novel "eLFA-NP-LA" method provides full coverage against single link and node failures, leads to comparable or shorter path lengths than other existing FRR approaches (MPLS-FRR facility backup, not-via addresses, LFAs, rLFAs, MRTs), and is superior in terms of additional forwarding entries and coverage in case of multiple failures.

References

[1] S. Rai *et al.*, "IP Resilience within an Autonomous System: Current Approaches, Challenges, and Future Directions," *IEEE Communications Magazine*, vol. 43, no. 10, Oct. 2005.

[2] A. Raj and O. Ibe, "A Survey of IP and Multiprotocol Label Switching Fast Reroute Schemes," *Computer Networks*, vol. 51, no. 8, 2007.

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

[4] S. Bryant, S. Previdi, and M. Shand, "RFC6981: A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses," Jul. 2013.

[5] 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, Jun. 2010.

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

[7] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "RFC7490: Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)," Apr. 2015.

[8] P. Sarkar (Ed.) *et al.*, "RFC8102: Remote-LFA Node Protection and Manageability," https://tools.ietf.org/html/rfc8102, Mar. 2017.

[9] L. Csikor and G. Retvari, "On Providing Fast Protection with Remote Loop-Free Alternates: Analyzing and Optimizing Unit Cost Networks," *Telecommunication Systems*, 2015.

[10] L. Csikor, J. Tapolcai, and G. Retvari, "Optimizing IGP link costs for improving IP-level resilience with Loop-Free Alternates," *Computer Communications*, vol. 36, no. 6, pp. 645 – 655, Mar. 2013.

[11] G. Retvari, J. Tapolcai, G. Enyedi, and A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited," in *IEEE Infocom*, Apr. 2011.

[12] W. Tavernier *et al.*, "Self-configuring Loop-free Alternates with High Link Failure Coverage," *Telecommunications Systems*, vol. 56, no. 1, pp. 85–101, May 2014.

[13] A. Farrel and R. Bonica, "Segment Routing: Cutting Through the Hype and Finding the IETF's Innovative Nugget of Gold," *IETF Journal*, vol. 13, no. 1, Jul. 2017.

[14] A. Bashandy *et al.*, "Topology Independent Fast Reroute using Segment Routing," https://tools.ietf.org/html/draft-bashandy-rtgwg-segment-routing-ti-lfa, Jul. 2017.

[15] A. Bashandy *et al.*, "Loop Avoidance Using Segment Routing," https://tools.ietf.org/html/draft-bashandy-rtgwg-segment-routing-uloop, Jul. 2017.

[16] S. Nelakuditi *et al.*, "Fast Local Rerouting for Handling Transient Link Failures," *IEEE/ACM Trans. on Networking*, vol. 15, no. 2, Apr. 2007.

[17] A. Kvalbein *et al.*, "Fast IP Network Recovery Using Multiple Routing Configurations," in *IEEE Infocom*, Apr. 2006.

[18] A. Atlas *et al.*, "RFC7812: An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)," Jun. 2016.

[19] M. Menth *et al.*, "Performance Comparison of Not-Via Addresses and Maximally Redundant Trees (MRTs)," in *IEEE/IFIP IM*, Apr. 2013.

[20] K. Kuang, S. Wang, and X. Wang, "Discussion on the Combination of Loop-Free Alternates and Maximally Redundant Trees for IP Networks Fast Reroute," in *IEEE ICC*, June 2014.

[21] S. Cho, T. Elhourani, and S. Ramasubramanian, "Independent Directed Acyclic Graphs for Resilient Multipath Routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 153 –162, Feb. 2012.

[22] Y. E. Oktian *et al.*, "Distributed SDN Controller System: A Survey on Design Choice," *Computer Networks*, vol. 121, pp. 100–111, 2017.

[23] S. Sharma *et al.*, "OpenFlow: Meeting Carrier-Grade Recovery Requirements," *Computer Communications*, vol. 36, no. 6, 2013.

[24] K.-W. Kwong, L. Gao, R. A. Guerin, and Z.-L. Zhang, "On the Feasibility and Efficiency of Protection Routing in IP Networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, Oct. 2011.

[25] J. Kempf *et al.*, "Scalable Fault Management for OpenFlow," in *IEEE International Conference on Communications (ICC)*, 2012.

[26] N. L. van Adrichem *et al.*, "Fast Recovery in Software-Defined Networks," in *EWSDN*, Sep. 2014.

[27] R. M. Ramos *et al.*, "SlickFlow: Resilient Source Routing in Data Center Networks Unlocked by OpenFlow," in *LCN*, Oct. 2013.

[28] C. Cascone *et al.*, "SPIDER: Fault Resilient SDN Pipeline with Recovery Delay Guarantees," in *IEEE NetSoft*, June 2016.

[29] W. Braun and M. Menth, "Loop-Free Alternates with Loop Detection for Fast Reroute in Software-Defined Carrier and Data Center Networks," *Journal of Network and Systems Management*, vol. 24, no. 3, 2016.

[30] N. L. M. van Adrichem *et al.*, "Backup Rules in Software-Defined Networks," in *IEEE NFV-SDN*, Nov 2016.

[31] S. Cevher *et al.*, "Multi Topology Routing Based IP Fast Re-Route for Software Defined Networks," in *IEEE ISCC*, 2016.

[32] P. Francois *et al.*, "An Evaluation of IP-Based Fast Reroute Techniques," in *ACM CoNEXT*, 2005.

[33] OpenFlow Switch Consortium *et al.*, "OpenFlow Switch Specification Version 1.1.0," Decemeber 2011.

[34] OpenFlow Switch Consortium *et al.*, "OpenFlow Management and Configuration Protocol 1.2 (OF-Config 1.2)," April 2015.

[35] S. Li *et al.*, "Flexible Traffic Engineering: When OpenFlow Meets Multi-Protocol IP-Forwarding," *IEEE Comm. Letters*, vol. 18, no. 10, 2014.

[36] S. Knight *et al.*, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, Oct. 2011.

[37] S. Halabi, "OSPF Design Guide," Cisco Systems, Tech. Rep., Apr. 1996.