Demonstration of Dynamic Restoration in Segment Routing Multi-layer SDN Networks

A. Giorgetti[1], **A. Sgambelluri**[1], **F. Paolucci**[1], **F. Cugini**[2], **and P. Castoldi**[1]
1: Scuola Superiore Sant'Anna, Pisa, Italy; 2: CNIT, Pisa, Italy
e-mail a.sgambelluri@sssup.it

Abstract: Dynamic traffic recovery is designed and validated in a multi-layer network exploiting an SDN-based implementation of Segment Routing. Traffic recovery is locally performed from the node detecting the failure up to the destination node without involving the SDN controller. Experimental results demonstrate recovery time within 50 ms.

OCIS codes: 060.0060; 060.4250; 060.4261.

1. Introduction

Segment Routing (SR) is a novel traffic engineering (TE) technique compatible with traditional Multiprotocol Label Switching (MPLS) data plane [1]. SR relies on label stacking to steer traffic using the source routing paradigm. Specifically, traffic flows are enforced through a given path by applying, at the ingress node, a specifically designed stack of MPLS labels (i.e., the *segment list*). Each packet is then forwarded along the shortest path toward the network element represented by the top label in the segment list. For instance, a label can represent an Interior Gateway Protocol (IGP) prefix identifying a specific node (i.e., IGP-Node Segment).

Unlike traditional MPLS networks, SR maintains per-flow state only at the ingress node, where the segment list is applied. No signaling protocol (e.g., Reservation Protocol with traffic engineering extensions - RSVP-TE) is required. Thus, scalability of transit nodes is greatly improved, since state information is not required.

Several SR use cases have been recently proposed. As an example, SR can be effectively used to steer traffic flows on paths characterized by low latency values and to avoid link congestion using a very limited number of tunnels. Moreover, SR can be effectively used upon network failures to dynamically perform traffic recovery without involving the controller in the recovery process [2]. However, SR may suffer from other potential issues. Indeed, deployed MPLS equipments typically support a limited number of stacked labels. Therefore, it is important to minimize the the segment list depth.

Even though standardization of SR is rapidly evolving, there has been a limited research work within the academic community. Authors of [3] proposed to combine the benefits of SR with those of a Software Defined Networking (SDN) architecture. The work in [4] implemented SR in a Carrier Ethernet networks including experimental and simulation studies. An algorithm to compute the segment list of a given path based on a graph model is proposed in [5]. Our previous work [2] described a SR procedure to re-route disrupted traffic flows around a faulted network element, however the proposed method may lead to long segment lists. Finally, the work in [6] proposed a dynamic restoration scheme for SDN-based networks, but it does not consider the utilization SR.

This study proposes, implements and experimentally validates a SR scheme (i.e., SR-FAILOVER) to dynamically recover traffic flows disrupted by link or node failures minimizing the segment list depth. The central controller is not involved upon failure occurrence so that the traffic is locally recovered from the node detecting the failure up to the destination node. First, the SR-FAILOVER scheme is evaluated in a simulated environment in terms of segment list depth considering a number of network topologies. Second, it is experimentally implemented in a SDN-based testbed and validated in terms of achievable recovery time.

2. Segment routing recovery

In the SR-FAILOVER scheme each network node is properly configured during network initialization so that when a node physically detects a failure of a connected interface it is able to locally redirect the traffic on a proper backup path. The initialization can be performed in a distributed way using a properly extended IGP or using a centralized SDN controller.

978-1-943580-07-1/16/$31.00 ©2016 Optical Society of America

NODE A	
IN Label	Action
B	Out 1
C	Out 2
D	Out 1
E	Out 2

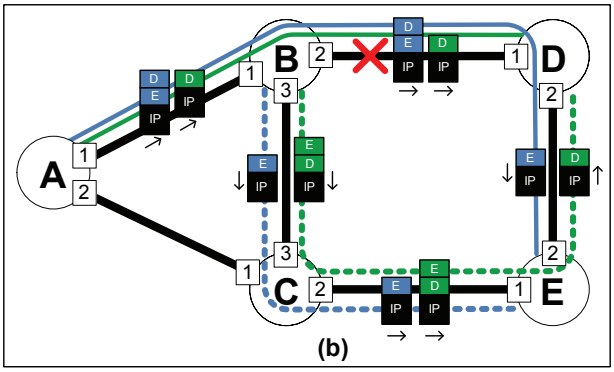
NODE B	
IN Label	Action
A	Out 1
C	Out 3
D	Out 2
E	Out 2, Out 3

NODE C	
IN Label	Action
A	Out 1
B	Out 3
D	Out 2, Out 3
E	Out 2

NODE D	
IN Label	Action
A	Out 1
B	Out 1
C	Out 1, Out 2
E	Out 2

NODE E	
IN Label	Action
A	Out 1
B	Out 1, Out 2
C	Out 1
D	Out 2

(a)



NODE B		
IN Label	Action	Backup Action
A	Out 1	Pop, go to table fail 1
C	Out 3	Pop, go to table fail 3
D	Out 2	Pop, go to table fail 2
E	Out 2, Out 3	Pop, go to table fail 2 or 3

B - fail 1	
IN	Action
A	Out 3
C	Out 3
D	Out 2
E	Out 3

B - fail 2	
IN	Action
A	Out 1
C	Out 3
D	Push E, Out 3
E	Out 3

B - fail 3	
IN	Action
A	Out 1
C	Out 1
D	Out 2
E	Out 2

(c)

Fig. 1: Scheme description and experimental testbed : (a) simplified primary forwarding table of each node; (b) testbed topology with two traffic flows; (c) detailed primary forwarding table and failover forwarding tables of node B.

The target of the SR-FAILOVER scheme is to re-route the disrupted traffic from the node detecting the failure up to the destination node (i.e., the node indicated in the last label of the segment list). To this extent we propose to use a *primary forwarding table* in each node and a number of *failover forwarding tables*, i.e., one failover table is required for each interface of the node. When the output port indicated in the primary table for a specific ingress label is down the backup actions in the primary table are executed. Following the backup actions the node first pops all the labels in the segment list except the bottom label of the stack, then the packet processing is passed on the proper failover table, see Fig. 1(c). The actions to be executed within the failover tables depend on the value of the bottom label. Specifically, for each label the failover table enforces the utilization of the shortest path computed on the topology without the failed link. To do this, a number of push actions may be required before forwarding the packet.

As an example, Fig. 1(a) illustrates all the primary tables of the nodes belonging to the network topology illustrated in Fig. 1(b). A traffic flow traversing the path $\bar{p}_1 = \{A,B,D,E\}$, i.e., blue solid line in Fig. 1(b), is established and encoded with the segment list $\overline{SL}^{\bar{p}_1} = \{D,E\}$. Thus, node B receives the traffic with label D and forwards the traffic toward node D using interface 2. In case of failure of this interface, label D is popped then the switch is redirected to failover table 1 that implements the routing on a topology where the link $B-D$ is not included. Specifically, Fig. 1(c) illustrates all the failover tables configured at node B, in the case of $\bar{p}_1$ the backup path is $\{A,B,C,E\}$, i.e., blue dashed line in Fig. 1(b). Since the bottom of the stack is label E, traffic is simply forwarded on interface 3 without requiring other actions. Conversely, for path $p_2 = \{A,B,D\}$, i.e., green solid line in Fig. 1(b) the backup path is $\{A,B,C,E,D\}$, i.e., green dashed line in Fig. 1(b). In this case the destination is D and a new label is required to be pushed by node B (i.e., label E) before forwarding it on interface 3. So that, at node C, the traffic matches the desired backup path in the primary table and it is correctly forwarded using interface 2.

The SR-FAILOVER scheme as described in the previous paragraph considers single link failures. However, it can be easily extended to node failures. Indeed, it is sufficient to initialize the failover tables computing the backup paths on the topology without all the links connected to the failed node. As an example, if node A detects a failure on interface 1 the backup paths should be computed assuming the failure of node B, i.e., excluding bidirectional links $A-B$, $C-B$, and $D-B$ from the network topology.

3. Simulative and experimental validation

To assess the scalability of the SR-FAILOVER scheme, in terms of segment list depth, two network topologies are considered in a simulated environment: a Pan European network including 27 nodes and 55 bidirectional links, and a topology generated with BRITE [7] including 150 nodes and 300 bidirectional links. In both topologies, the segment

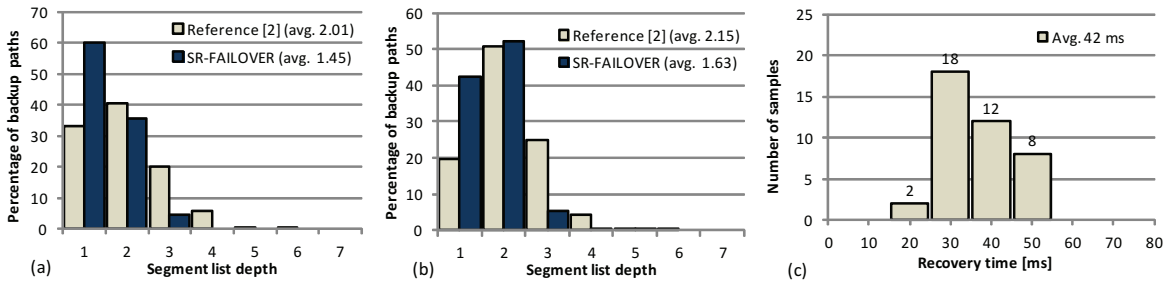


Fig. 2: Statistical distribution: (a) segment list depth in Pan European topology; (b) segment list depth in BRITE topology; (c) recovery time in the experimental testbed.

list depth is computed for a set of backup paths with SR-FAILOVER scheme and with the scheme described in [2] where backup paths are computed from the point of failure to the next label in the segment list. A single backup path is computed for each link failure that can affect every shortest path in the network. This way, 6332 and 198270 backup paths are considered for the Pan European and the BRITE topologies, respectively.

Fig. 2(a) and Fig. 2(b) show the obtained distribution of the segment list depth. In the Pan European topology the SR-FAILOVER scheme achieves an average segment list depth of 1.45 with respect to the average value 2.01 obtained using the reference scheme [2]. In the BRITE network the achieved segment list depth is 1.63 with respect to 2.15. Also it is shown that, for both topologies, using the SR-FAILOVER scheme 95% of the backup paths use a segment list of 1 or 2 labels, whereas this percentage is about 70% using the reference scheme.

The SR-FAILOVER scheme has been implemented in an experimental testbed using an SDN controller and five OpenFlow switches. In the switches, the primary forwarding table uses the *Group Table* OpenFlow feature to enable the monitoring of the interfaces and support a backup list of actions to be applied when the primary forwarding interface is down. Nodes have been implemented within Intel Core 4 servers (CPU 2.40 GHz, Kernel 2.6) equipped with four Gb/s Ethernet interfaces and running OpenvSwitch version 2.61, supporting MPLS-based forwarding. OpenFlow 1.3 has been utilized for the communication between the nodes and the controller. Commercially available ROADMs equipped with 10Gb/s OTN muxponders have been connected to nodes for implementing the optical transport plane of the multi-layer network. The controller has been implemented by extending the SDN RYU controller [8].

The aforementioned hardware has been utilized to realize the network topology represented in Fig. 1(b) where two traffic flows along paths $\bar{p}_1$ and $\bar{p}_2$ have been established. 20 failures of the link $B-D$ have been emulated by physically disconnecting interface 2 of node B; each failure disrupts both established traffic flows. Fig. 2(c) shows the distribution of the obtained recovery time (40 samples) where the average value is 42 ms.

4. Conclusions

Traffic recovery has been implemented and demonstrated in a multi-layer SDN network based on Segment Routing. Simulation results showed that using the SR-FAILOVER scheme the wide majority of backup paths can be encoded with segment list of one or two labels. Experimental measurements reported an average recovery time of 42 ms.

Acknowledgement: This work has been partially supported by H2020 project 5GEx.

References

1. C. Filsfils *et al.*, "Segment routing architecture," draft-filsfils-spring-segment-routing-05, Sep. 2015.
2. A. Giorgetti, A. Sgambelluri, F. Paolucci, and P. Castoldi, "Reliable segment routing," in *RNDM*, Oct. 2015.
3. D. Cai *et al.*, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. WoWMoM*, 2014.
4. S. Bidkar *et al.*, "Field trial of a software defined network (SDN) using carrier ethernet and segment routing in a tier-1 provider," in *Globecom*, Dec. 2014.
5. F. Lazzeri *et al.*, "Efficient label encoding in segment-routing enabled optical networks," in *Proc. ONDM*, 2015.
6. A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," *J. Opt. Commun. Netw.*, vol. 5, no. 9, pp. 1066–1075, Sep. 2013.
7. A. Medina *et al.*, "BRITE: an approach to universal topology generation," in *Proc. MASCOTS*, 2001.
8. "Ryu controller." [Online]. Available: http://osrg.github.io/ryu/