Path Encoding in Segment Routing

Alessio Giorgetti, Piero Castoldi
Scuola Superiore Sant'Anna
Pisa, Italy
Email: a.giorgetti@sssup.it

Filippo Cugini
CNIT
Pisa, Italy

Jeroen Nijhof, Francesco Lazzeri
and Gianmarco Bruno
Ericsson
Genova, Italy

***Abstract*—Segment Routing (SR) is emerging as an innovative traffic engineering technique compatible with traditional MPLS data plane. SR relies on label stacking, without requiring a signaling protocol. This greatly simplifies network operations in transit nodes. However, it may introduce scalability issues at the ingress node and packet overhead. Therefore, specific algorithms are required to efficiently compute the label stack for a given path.**

This study proposes two algorithms for SR label stack computation of strict routes that guarantee minimum label stack depth. Then, SR scalability performance is investigated. Results show that, in most of the cases, SR uses label stacks composed of few labels and introduces a limited packet overhead. However, relevant scalability issues may arise in specific cases, e.g., large planar topologies.

I. Introduction

Segment Routing (SR) has been recently proposed as an innovative technique to provide traffic engineering (TE) by simplifying control plane operations [1], [2]. SR relies on the source routing paradigm and can be deployed in packet networks supporting Multiprotocol Label Switching (MPLS). Indeed, according to SR, packet flows are enforced through a specific path by applying, at the ingress node, a specifically designed stack of segment identifiers (SIDs) fully compatible with the MPLS data plane.

The stack of SIDs is named *segment list* and corresponds to the *stack of labels* in the MPLS architecture. Only the top SID in the list is processed during packet forwarding by transit nodes. In particular, each packet is forwarded along the shortest path toward the network element represented by the top SID. For instance, a SID can represent an Interior Gateway Protocol prefix (i.e., IGP-Prefix) which identifies a specific router (i.e., IGP-Node Segment [1]) or a local adjacency (i.e., IGP-Adjacency Segment). Unlike traditional MPLS networks, SR maintains per-flow state only at the ingress node, where the segment list is applied. Therefore, no signaling protocol (e.g., Reservation Protocol with traffic engineering extensions - RSVP-TE) is required to populate the forwarding table of transit nodes. In this way, a simplified control plane is employed, just relying on the IGP that is properly extended to advertise SIDs [3]. Scalability of transit nodes is greatly improved, since MPLS Label Switch Paths (LSPs) state information is not required.

An interesting SR use case is its application to steer traffic flows on paths characterized by low latency values. As an example, it has been demonstrated on real network topologies that by applying a single SIDs it is possible to significantly reduce the latency overhead with respect to the minimal latency path [4]. In a second use case, it is shown that with the utilization of SR it is possible to avoid link congestion using a number of tunnels that is 1000 times less with respect to the utilization of RSVP-TE.

However, SR may suffer from other potential issues [5]. First, currently deployed MPLS equipments typically support a limited number of stacked labels, called *segment list depth* in the context of SR. Therefore, the segment list depth may be constrained to a limited value to guarantee SR backward compatibility. This aspect is particularly relevant when equal-cost multiple paths (ECMPs) are available toward a destination. Indeed, SR has to be capable of either load balancing among ECMPs or enforcing traffic flows along a strict path. As it will be shown later, the latter case may lead to a large number of labels in the segment list. Second, adding a segment list to each packet, SR introduces additional packet overhead with respect to traditional IP/MPLS networks not employing label stacking (i.e., relying on a single label per packet). To address these issues, specific algorithms are required to efficiently compute the segment list encoding a given path minimizing both the segment list depth and the introduced packet overhead.

So far, these aspects have not been adequately investigated. Even though standardization is advanced and the main network equipment vendors are currently involved in the implementation phase (commercial deployment of SR products has been announced for the next few months [6]), there has been a limited research work within the academic community. The work in [7] considers the SR application in Carrier Ethernet networks. In particular, the authors propose to combine the benefits of SR with those of a software defined networking (SDN) architecture [4]. In [5], [8], the authors proposed a SR implementation for Carrier Ethernet networks including both a testbed evaluation and a simulation study. The proposed solution is able to reduce the required segment list depth but implies integrations of the segment list at some intermediate nodes (i.e., the swap nodes), therefore it is not compliant with the source routing paradigm. However, all aforementioned works do not propose algorithms for segment list computation and do not evaluate the achievable segment list depth and packet overhead.

Our previous work in [9] proposes an algorithm to compute the segment list using an auxiliary network graph. Finally, the works in [10]–[12] propose two experimental implementations of SR respectively based on an OpenFlow-based controller and a PCE-based controller.

This study proposes two algorithms for computing the segment list encoding a given path. Both algorithms are compliant with the on-going standardization and provide the segment list of minimum depth when a unique path has to be identified. Then, using the two proposed algorithms, SR scalability is assessed in several network scenarios in terms of segment list depth and introduced packet overhead. Obtained results confirm that the two algorithms provide the same segment list depth for each encoded path and demonstrate that one of them guarantees lower packet overhead.

Algorithm 1 *SR-D*$(\overline{SL}, \bar{p}, d)$ iteratively fills the segment list $\overline{SL}$ representing the target path $\bar{p}$, d is the iteration step.

```
 1: if $d \neq 0$ then
 2:     $SL_{d-1} \leftarrow p_0$
 3: end if
 4: for $i = 1$ to $i < |\bar{p}|$ step $1$ do
 5:     for $j = 0$ to $j \leq i$ step $1$ do
 6:         $s_j \leftarrow p_j$
 7:     end for
 8:     if $\bar{s}$ is not unique shortest path from $p_0$ to $s_{|\bar{s}|-1}$ then
 9:         break
10:     end if
11: end for
12: if $i < |\bar{p}|$ then
13:     for $j = i - 1$ to $j < |\bar{p}|$ step $1$ do
14:         $k \leftarrow j - (i - 1)$
15:         $p'_k \leftarrow p_j$
16:     end for
17:     SR-D$(\overline{SL}, \bar{p}', d + 1)$
18: else
19:     $SL_d \leftarrow p_{|\bar{p}|-1}$
20: end if
21: return
```

II. Segment Routing

SR is typically associated with a centralized control plane implementation (e.g., SDN [4], [7], [13], [14]). In this scenario, when a new traffic flow has to be established, a request is issued to the centralized controller that determines the traffic path based on its vision of the network (e.g., retrieved by listening to IGP advertisements). Specifically, upon reception of the request, the controller first performs the path computation and then encodes the computed path using a sequence of SIDs. This way, it determines the segment list to be applied to all packets belonging to that traffic flow. In particular, the segment list is determined assuming that all the nodes in the network will forward the packet using MPLS rules, i.e., looking at the top SID and forwarding the packet on the interface associated with that SID in the forwarding table.

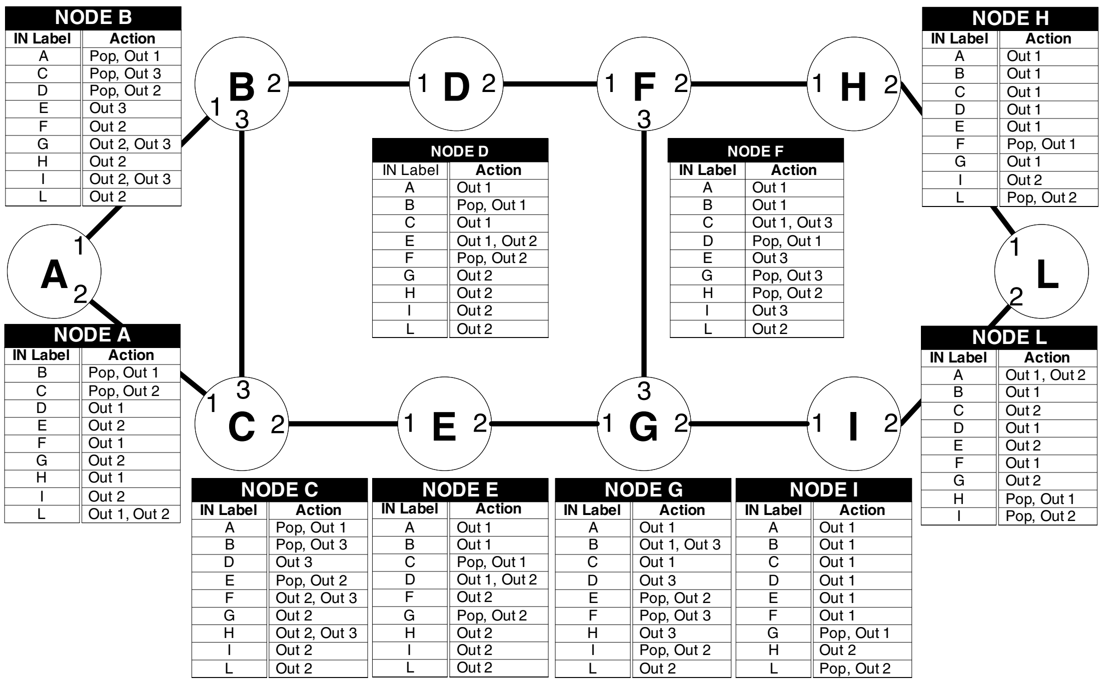
Many types of segment identifiers can be considered. Among them, node and adjacency identifiers are typically utilized when the target is to enforce a specific path for a traffic flow. This work focuses on the utilization of node identifiers. Adjacency segments are left for future work since they are required only to discriminate among multiple links between two adjacent nodes.

Fig.1 illustrates an example where all the MPLS forwarding tables are detailed supposing that Open Shortest Path First with traffic engineering extensions (OSPF-TE) is used as IGP to advertise the node identifiers, as proposed in [3]. Let assume that a new traffic flow is issued to the controller from node A to node I and the controller selects the path $\bar{p}_1 = \{A, C, E, G, I\}$. Since $\bar{p}_1$ is the *unique shortest path* from A to I in terms of number of hops (i.e., there is no alternative path having equal or lower cost in the network graph between A and I), the segment list $\overline{SL}^{\bar{p}_1}$ used for coding $\bar{p}_1$ only includes one label (i.e., $\overline{SL}^{\bar{p}_1} = \{I\}$). The packets are then forwarded along $\bar{p}_1$ without modifying the segment list up to node G where the label I is popped (i.e., penultimate hop popping) and the packet is forwarded to node I.

Alternatively, if the controller selects the path $\bar{p}_2 = \{A, B, D, F, G, I\}$ a segment list composed of at least two labels is required because this path is not a shortest path to node I. Specifically, a possible segment list is $\overline{SL}^{\bar{p}_2} = \{F, I\}$. In this case the packets are forwarded up to node D without modification to the segment list. At node D the label F is popped so that node F will receive packets with the single label I and will forward it through the unique shortest path toward node I. At node G, the label I is removed and the packet is finally forwarded to node I.

A. Equal Cost Multi Path aware routing

SR natively implements ECMP-aware routing, where in case of multiple shortest paths toward the destination the traffic is load balanced on a per-flow



NODE B

IN Label	Action
A	Pop, Out 1
C	Pop, Out 3
D	Pop, Out 2
E	Out 3
F	Out 2
G	Out 2, Out 3
H	Out 2
I	Out 2, Out 3
L	Out 2

NODE A

IN Label	Action
B	Pop, Out 1
C	Pop, Out 2
D	Out 1
E	Out 2
F	Out 1
G	Out 2
H	Out 1
I	Out 2
L	Out 1, Out 2

NODE D

IN Label	Action
A	Out 1
B	Pop, Out 1
C	Out 1
E	Out 1, Out 2
F	Pop, Out 2
G	Out 2
H	Out 2
I	Out 2
L	Out 2

NODE F

IN Label	Action
A	Out 1
B	Out 1
C	Out 1, Out 3
D	Pop, Out 1
E	Out 3
G	Pop, Out 3
H	Pop, Out 2
I	Out 3
L	Out 2

NODE H

IN Label	Action
A	Out 1
B	Out 1
C	Out 1
D	Out 1
E	Out 1
F	Pop, Out 1
G	Out 1
I	Out 2
L	Pop, Out 2

NODE L

IN Label	Action
A	Out 1, Out 2
B	Out 1
C	Out 2
D	Out 1
E	Out 2
F	Out 1
G	Out 2
H	Pop, Out 1
I	Pop, Out 2

NODE C

IN Label	Action
A	Pop, Out 1
B	Pop, Out 3
D	Out 3
E	Pop, Out 2
F	Out 2, Out 3
G	Out 2
H	Out 2, Out 3
I	Out 2
L	Out 2

NODE E

IN Label	Action
A	Out 1
B	Out 1
C	Pop, Out 1
D	Out 1, Out 2
F	Out 2
G	Pop, Out 2
H	Out 2
I	Out 2
L	Out 2

NODE G

IN Label	Action
A	Out 1
B	Out 1, Out 3
C	Out 1
D	Out 3
E	Pop, Out 2
F	Pop, Out 3
H	Out 3
I	Pop, Out 2
L	Out 2

NODE I

IN Label	Action
A	Out 1
B	Out 1
C	Out 1
D	Out 1
E	Out 1
F	Out 1
G	Pop, Out 1
H	Out 2
L	Pop, Out 2

Fig. 1. Test network (TNet) topology detailing Segment Routing examples. The MPLS forwarding table is reported for each node.

basis. As an example, in Fig.1 traffic from node C to node F and labeled with the single label F will be load balanced on the two equal cost shortest paths $\bar{p}_3 = \{C, E, G, F\}$ and $\bar{p}_4 = \{C, B, D, F\}$.

If the ECMP load balancing has to be avoided, that is a strict path has to be identified, additional SIDs in the segment list are required. As an example, if the target path is $\bar{p}_3$ at least two labels are required in the segment list to discriminate between the two shortest paths (e.g., $\overline{SL}^{\bar{p}_3} = \{G, F\}$).

III. Label stack computation

This section describes two proposed algorithms for generating the segment list for a pre-determined path (i.e., the *target path*). Both algorithms have been implemented using an iterative approach that navigates the target path and performs several comparisons with shortest paths locally stored or dynamically computed by the controller. Alg. 1 defines the Segment Routing Direct (*SR-D*) scheme; Alg. 2 defines the Segment Routing Reverse (*SR-R*) scheme. *SR-D* navigates the target path starting from the source node toward the destination, whereas *SR-R* navigates the target path in the opposite direction.

In the algorithms description, the following notation is used: $\overline{SL}$, $\bar{p}$, $\bar{p}'$, and $\bar{s}'$ are vectors of nodes in the general form $\bar{v} = \{v_0, v_1, \ldots, v_{n-1}\}$ including $|\bar{v}|$ elements. Specifically, $\overline{SL}$ is the output of the algorithm and represents the computed segment list; $\bar{p}$ is the target path; $\bar{p}'$ is the target path generated for the next algorithm iteration; $\bar{s}$ is the *target segment*, dynamically built to navigate the target path.

Algorithm 2 *SR-R*$(\overline{SL}, \bar{p}, d)$ iteratively fills the segment list $\overline{SL}$ representing the target path $\bar{p}$, d is the iteration step.

```
 1: $SL_d \leftarrow p_{|\bar{p}|-1}$
 2: for $i = 1$ to $i < |\bar{p}|$ step 1 do
 3:     for $j = 0$ to $j \leq i$ step 1 do
 4:         $s_j \leftarrow p_{|\bar{p}|-1-i+j}$
 5:     end for
 6:     if $\bar{s}$ is not unique shortest path from $s_0$ to $p_{|\bar{p}|-1}$
        then
 7:         break
 8:     end if
 9: end for
10: if $i < |\bar{p}|$ then
11:     for $j = 0$ to $j < |\bar{p}| - (i-1)$ step 1 do
12:         $p'_j \leftarrow p_j$
13:     end for
14:     SR-R$(\overline{SL}, \bar{p}', d+1)$
15: end if
16: if $d = 0$ then
17:     reverse the order of $\overline{SL}$ elements
18: end if
19: return
```

In Alg. 1, lines 1-3 insert the source node of the target path in the segment list, this operation is not required on the first iteration step. Lines 4-11 implement a loop to generate target segments by gradually including nodes taken from the target path. This way,

TABLE I
SEGMENT ROUTING RESULTS SUMMARY

	N	H	SLD		Avg. Oh		
			Avg.	Max	*SR-D*	*SR-R*	*MPLS*
Fig. 1	10	2	1.7	3	3.2	2.7	2.2
PanEU	27	3.8	3.3	7	8.2	8.1	4.1
NSF	14	2.3	1.8	3	2.8	2.4	2
MultiD	75	6.8	3.8	8	17.5	16.9	7.5
BRITE	50	3.3	2.5	7	5.4	5.1	3.2
BRITE	100	3.9	2.4	6	6.3	5.8	3.7
BRITE	150	4.1	2.5	7	6.9	6.3	4.1
BRITE	200	4.3	2.5	7	7.3	6.6	4.3
Grid	25	5.2	3.4	8	8.8	8.8	4.2
Grid	49	8.6	5.1	12	22.5	22.5	7.6
Grid	64	10.5	6	14	22.5	22.5	9.5
Grid	100	14.4	8	18	56.3	56.3	13.3

the first evaluated target segment only includes the first two nodes of the target path, the second one includes the first three nodes and so forth. Specifically, if the generated target segment is a unique shortest path, an additional node is included from the target path. If not, the loop is broken. Lines 12-17 implement the iteration condition. If the loop has been broken before reaching the last node of the target path, a new target path is built starting from the penultimate node of the current target segment (lines 13-16). Then the algorithm is iterated (line 17). Lines 18-20 implement the condition to terminate the iteration. Specifically, these lines are executed only if the last considered target segment is equal to the target path: in this case the last node of the target path is inserted in the segment list.

Alg. 2 has the same structure of Alg. 1 but generates target segments by navigating the target path in the opposite direction (lines 2-9). The first evaluated target segment only includes the last two nodes of the target path, the second one includes the last three nodes and so forth. In this case, the segment list is generated in reverted order, therefore it is required to re-order it before exiting the algorithm (lines 16-18).

As an example, with reference to Fig. 1, if both algorithms are used to encode the target path $\bar{p}_5 = \{A, C, E, G, F, H\}$, the two obtained segment lists are $\overline{SL}^{\bar{p}_5}_{SR\text{-}D} = \{G, H\}$ and $\overline{SL}^{\bar{p}_5}_{SR\text{-}R} = \{E, H\}$, respectively. SR-D and SR-R respectively tend to insert in the segment list nodes closer to the destination and to the source.

Both algorithms provide a segment list of minimum depth. This is proved in the following for *SR-D* algorithm, a symmetric proof can be provided for *SR-R*. Note that the segment list provided by *SR-D* produces a partition of $\bar{p}$ in a sequence of M adjacent segments $\bar{s}^i$ such that $\bar{p} = \{\bar{s}^1, \bar{s}^2, \ldots, \bar{s}^M\}$ where the generic j-th element of $\bar{s}^i$ is an element of $\bar{p}$, i.e., $s^i_j = p_{j+\sum_{k=1}^{i-1}|\bar{s}^k|}$. Thus, the segment list computed for path $\bar{p}$ is composed

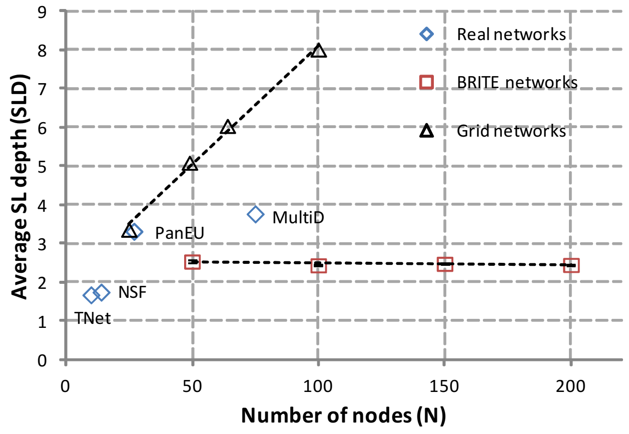


Fig. 2. SL depth vs. number of nodes. BRITE topologies results are reported with confidence interval at 99% confidence level.

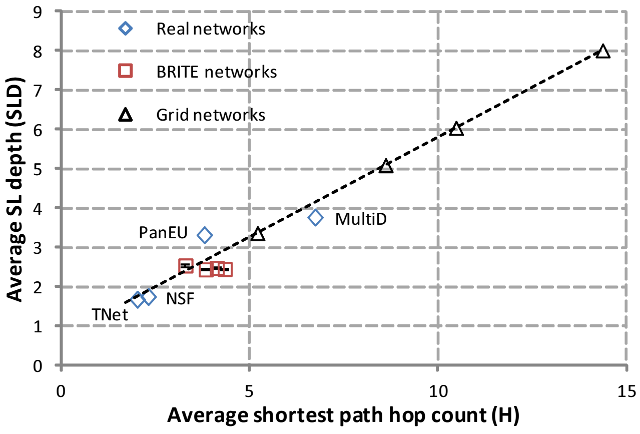


Fig. 3. SL depth vs. average hop distance. BRITE topologies results are reported with confidence interval at 99% confidence level.

by the SIDs identifying the last node of each segment, that is, $\overline{SL}^{\bar{p}} = \{s^1_{|s^1|-1}, s^2_{|s^2|-1}, \ldots, s^M_{|s^M|-1}\}$.

Lemma 1: SR-D provides a partition of $\bar{p}$ that includes the minimum number K of segments such that, for all $i = 1, \ldots, K$, the i-th segment $\bar{s}^i$ is a unique shortest path from s^i_0 to $s^i_{|\bar{s}^i|-1}$.

Proof: By construction, each one of the M segments $\bar{s}^i$ provided by *SR-D* is a unique shortest path. Moreover, the segment $\bar{r}^i = \{s^i_0, s^i_1, \ldots, s^i_{|\bar{s}^i|-1}, s^{i+1}_0\}$ composed by $\bar{s}^i$ plus the following node in $\bar{p}$ is not a unique shortest path between s^i_0 and s^{i+1}_0 (i.e., lines 8-10 in Alg. 1).

Assuming that M is not the minimum number of unique shortest paths to partition $\bar{p}$, there must exist a partition of $\bar{p}$ composed of K unique shortest paths with $K < M$. In this case, at least one of this N segments must totally contain one of the segments s^i plus at least one following node in $\bar{p}$. But if a segment is composed of s^i plus one or more additional nodes of $\bar{p}$, then it is not a unique shortest segment. ■

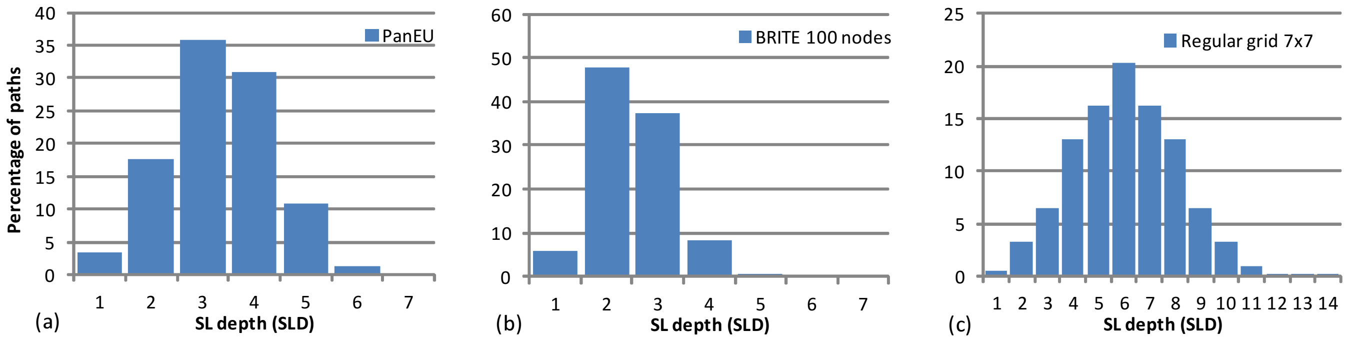


Fig. 4. Percentage of paths coded with each specific SLD value (a) PanEU; (b) BRITE topologies with 100 nodes; (c) 7×7 regular grid.

IV. Numerical Results

This section evaluates the two algorithms on several network topologies. Two performance metrics are considered: the segment list depth (i.e., ***SLD***) and the packet overhead (i.e., ***Oh***) introduced by SR. ***SLD*** is defined as the number of labels configured at the source node to route the packets along the desired path. For instance, with reference to Fig. 1, $\overline{SL}^{\bar{p}_2} = \{F, I\}$ and $SLD^{\bar{p}_2} = 2$. Oh is defined as the cumulative number of labels used at each hop along the desired path. Given a path $\bar{p}$ including $n-1$ links and the relative $\overline{SL}^{\bar{p}}$ of depth $SLD^{\bar{p}}$ the following definition holds $Oh = \sum_{i=0}^{n-1} SLD_i$ where SLD_i is defined as the number of labels included in the segment list at the i-th hop considering the label popping along the path. Therefore, two different Oh values can be obtained from two segment lists of the same depth, i.e., the segment list in which the labels are popped closer to the destination node introduces a higher Oh value. For instance, $Oh^{\bar{p}_2} = 6$, because on the first two hops the segment list contains 2 labels, on the third and the forth hops it contains 1 label, and on the last hop the segment list is totally removed. Considering that each MPLS label is composed of 4 bytes, one more unit of Oh means four more bytes that have to be processed by network nodes.

The considered topologies can be divided in three groups: (i) real topologies; (ii) topologies generated with BRITE [15]; (iii) regular grid topologies. Tab. I summarizes the network topology details and the results obtained by both algorithms. Specifically, for each topology, Tab. I reports: the number of nodes N; the average shortest path hop count H; the average and the maximum SLD obtained with the algorithms encoding, for all node pairs, all the paths within one hop from the shortest path. Since the two algorithms provide segment lists of minimum depth, the third and the forth columns hold for both algorithms. Tab. I finally includes the average overhead Oh obtained with *SR-D* and *SR-R* algorithms, and in traditional MPLS networks using a single label per flow (i.e., not exploiting label stacking). The values reported for the BRITE generated topologies are averaged over 100 different random topologies. The detailed representation of PanEU, MultiD and NSF networks can be found in [16] and [17]. BRITE typically generates not planar topologies where H is almost independent of N.

Tab. I shows that the ***SR-R*** scheme provides lower Oh values with respect to ***SR-D*** since the computed segment lists typically include nodes that are closer to the source node, as explained in Sec. III. However, both algorithms provide Oh values that are significantly higher with respect to the MPLS case, especially for planar topologies with high H values (e.g., MultiD and regular grid topologies). Finally, the table shows that the maximum SLD is limited to a value of 8 for real and BRITE networks, whereas it assumes significantly higher values for the grid topologies.

Fig. 2 illustrates the average SLD as a function of N, showing that there is not a clear relation between these two values. Conversely, Fig. 3 illustrates the average SLD as a function of H. This figure shows an almost linear relation between the average SLD and H and can therefore provide indications for larger topologies. Finally, Fig. 4 illustrates the percentage of paths coded with each specific SLD value considering PanEU, 7×7 grid, and 100 nodes BRITE topologies. For all the distributions we can conclude that the maximum SLD value is assumed only for a very few number of paths. As an example, for the BRITE topologies, more than 90% of the paths are coded with a segment list including less than four elements.

V. Conclusions

This paper introduced two algorithms (i.e., ***SR-D*** and ***SR-R***) for effective segment list computation in SR networks. Both algorithms provide the minimum segment list depth. However, algorithm SR-R guarantees lower packet overhead. The algorithms were applied on a number of network topologies to evaluate the scalability performance of SR. In most cases, low values of segment list depth and packet overhead have been demonstrated. However, under specific networking scenarios such as relatively wide planar topologies, larger values have been experienced. Even if they may occur rarely, these outsiders have to be considered to successfully implement the SR technology.

Acknowledgment

This work was partially funded by the European Community's through the H2020 project 5GEx.

References

[1] C. Filsfils *et al.*, "Segment routing architecture," *draft-filsfils-spring-segment-routing*, 2014.

[2] ——, "Segment routing with MPLS data plane," *draft-filsfils-spring-segment-routing-mpls*, 2014.

[3] P. Psenak *et al.*, "OSPF extensions for segment routing," *draft-psenak-ospf-segment-routing-extensions-05*, 2014.

[4] G. Swallow, "From tag switching to SDN and segment routing MPLS: an enduring architecture," in *Proc. MPLS SDN World Congress*, Mar. 2014.

[5] S. Bidkar, A. Gumaste, and A. Somani, "A scalable framework for segment routing in service provider networks: The omnipresent Ethernet approach," in *Proc. HPSR*, Jul. 2014.

[6] C. Filsfils, "Segment routing: Update and future evolution," in *Proc. MPLS SDN World Congress*, Mar. 2014.

[7] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. WoWMoM*, Jun. 2014.

[8] S. Bidkar, A. Gumaste, P. Ghodasara, S. Hote, A. Kushwaha, G. Patil, S. Sonnis, R. Ambasta, B. Nayak, and P. Agrawal, "Field trial of a software defined network (SDN) using carrier ethernet and segment routing in a tier-1 provider," in *Globecom Conf.*, Dec. 2014.

[9] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and C. P., "Efficient label encoding in segment-routing enabled optical networks," in *Proc. ONDM*, May 2015.

[10] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "SDN and PCE implementations for segment routing," in *Proc. NOC*, Jul. 2015.

[11] A. Sgambelluri, A. Giorgetti, F. Cugini, G. Bruno, F. Lazzeri, and P. Castoldi, "First demonstration of SDN-based segment routing in multi-layer networks," in *Optical Fiber Communication (OFC) Conference*, March 2015, pp. Th1A–5.

[12] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Experimental demonstration of segment routing," *J. Lightw. Technol.*, vol. PP, no. 99, early access on IEEE Xplore.

[13] 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.

[14] A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Dynamic restoration with GMPLS and SDN control plane in elastic optical networks [invited]," *J. Opt. Commun. Netw.*, vol. 7, no. 2, pp. A174–A182, Feb. 2015.

[15] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proc. MASCOTS*, 2001, pp. 346–353.

[16] A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow and PCE architectures in wavelength switched optical networks," in *Proc. ONDM*, Apr. 2012.

[17] A. Bukva, R. Casellas, R. Martinez, and R. Munoz, "A dynamic path-computation algorithm for a GMPLS-enabled multi-layer network," *J. Opt. Commun. Netw.*, vol. 4, no. 6, Jun. 2012.