# Label Encoding Algorithm for MPLS Segment Routing

Rabah Guedrez* Olivier Dugeon* Samer Lahoud† Géraldine Texier‡
*Orange Labs, Lannion, France,
rabah.guedrez@orange.com, olivier.dugeon@orange.com
†*IRISA/Université de Rennes 1/Adopnet Team, Rennes, France, samer.lahoud@irisa.fr
‡IRISA/Télécom Bretagne/Adopnet Team, Rennes, France, geraldine.texier@telecom-bretagne.eu

***Abstract*—Segment Routing is a new architecture that leverages the source routing mechanism to enhance packet forwarding in networks. It is designed to operate over either an MPLS or an IPv6 control plane. SR-MPLS, its instantiation over MPLS, encodes a path as a stack of labels inserted in the packet header by the ingress node. This overhead may violate the Maximum SID Depth (MSD), the equipment hardware limitation which indicates the maximum number of labels an ingress node can push onto the packet header. Currently, the MSD varies from 3 to 5 depending on the equipment manufacturer. Therefore, the MSD value considerably limits the number of paths that can be implemented with SR-MPLS. The consequence may be an inefficient network resource utilization and may also lead to congestion. We propose and analyze SR-LEA, an algorithm for an efficient path label encoding that takes advantage of the existing IGP shortest paths in the network. The output of SR-LEA is the minimum label stack to express SR-MPLS paths according to the MSD constraint. Therefore, SR-LEA substantially slackens the impact of MSD and restores the path diversity that MSD forbids in the network.**

***Index Terms*—Segment Routing, MPLS, SR-MPLS, label stack, traffic engineering.**

## I. Introduction

Segment Routing (SR) is a new architecture standardized by IETF SPRING working group [1]. It can be instantiated over two existing data plane MPLS (SR-MPLS) [1] [2] and IPv6 (SR-IPv6) [3]. In SR packet are forwarded using the source routing mechanism: the path the packet has to go through is encoded in its header. SR-MPLS is the central focus of the IETF working groups, mainly because of the important implications of service providers (SPs).

The major advantage of SR is that it eliminates the per-flow states from the SP's core routers. In fact, a path is directly usable by any router; no prior setup/signalization is required, unlike MPLS-TE where a tunnel has to be signaled and maintained using protocols such as the Resource Reservation Protocol Traffic Engineering (RSVP-TE). In SR, only the ingress node has to maintain per-flow states. Also, SR architecture adds extensions to already deployed IGP protocols: Open Shortest Path First (OSPF) [4], Intermediate System to Intermediate System (IS-IS) [5] and Border Gateway Protocol Link State [6] to exchange SR information. Therefore, SR-MPLS revokes the need for a label distribution protocol such as LDP or RSVP-TE.

A SR Path (SRP) is encoded as list of segments identifiers (SIDs), each SID associated with a data plane forwarding instruction *e.g.*, forward the packet down the IGP shortest path or forward to a specific exit interface.

In the SR instantiation over the MPLS data plane (SR-MPLS), a SID is represented by a 20-bit label. The SID is processed using the three standard MPLS operations POP, PUSH, and SWAP. A SRP is encoded as a stack of labels that the ingress router pushes onto the packet header. In fact, pushing more than one was supported since the early version of MPLS standards [7], the label stack has been used for multiple use cases: hierarchical tunnels, Layer 2 Virtual Private Network (L2VPN), and Layer 3 VPN. However, those use cases require a small number of labels, for example, a scenario of L2VPN or L3VPN requires only simultaneously two labels: the tunnel's label and VPN's label. To take full advantage of SR's potential, a router has to be able to push a larger number of labels. Unfortunately, current hardware suffers from physical limitation of the number of labels that can be used simultaneously [8].

In fact, in order to achieve wire-speed packet processing, hardware vendors use Application-specific integrated circuit (ASIC)s. They are designed to perform specific tasks very efficiently compared to general purpose processors. Consequently, they are limited in the size and the type of the operations they can perform. For example, the PUSH operation is implemented using dedicated ASICs that limit the number of labels they can push onto the packet header, this limitation in SR is known as the Maximum SID Depth (MSD). Therefore, an efficient label encoding able to reduce the labels stack size is essential to alleviate the MSD impact. In addition, reducing the label stack saves space and enables to carry other types of labels such as the entropy labels [9].

In this paper, we propose two label encoding algorithms for SR-MPLS paths. Both algorithms compute the minimum number of labels to express a SRP. We evaluate their performances over several real-world network topologies. The results are presented in term of the average number of labels to express a set network paths. In addition, we study their efficiency in alleviating the impact of the MSD limitation.

## II. Related works

In [10], Giorgetti *et al.* propose two SRP encoding algorithms that produce for the same SRP two label stacks of the equal size. Both algorithms use only Node-SIDs to encode a SRP. Unfortunately, in some cases, the resulting label stack may not correspond to the initial path. In fact, the proposed algorithms work well in a network where the shortest path between two directly connected nodes is a direct link, However, in real networks, a network administrator may choose to attribute higher costs to particular links that can lead up to the direct link not being the shortest path between two nodes.

In [11], a new network graph is constructed based on the initial network by adding a virtual link is added for every pair of nodes in addition to the existing physical links. The virtual links represent the Equal-Cost Multiple Paths (ECMP)s between two nodes. The path computation is performed over the new graph. The proposed label encoding algorithm replaces a virtual link by the tail's end node Node-SID whilst the physical link is replaced by an Adj-SID. This proposition suffers mainly from scalability issues. In fact, the new graph is much bigger that the initial network graph, with a number of links equal to the number of combinations of network nodes. For example, a network composed of 1000 nodes results in a new graph with approximately half million links. Several problems arise with such huge graphs, like for example slower response time due to the memory requirements, a strong computation complexity, and a considerable amount of processing required to update the network traffic engineering database.

## III. Segment Routing Path

In this section, we explain the main concepts of the SR architecture and detail how different types of SID are used to forward packets through the network.

In SR-MPLS, a node advertises a Segment Routing Global Bloc (SRGB). The SRGB is the range of labels allocated for SR (*e.g.*, [1000, 2000]). A SID can be global or local to the node that advertises it. A global SID takes its value within the SRGB (*e.g.*, 1100), all the SR nodes install a forwarding instruction associated with each global SIDs. A local SID takes its value outside the SRGB (*e.g.*, 3000); it is advertised in the SR domain but only the node advertising it possesses an associated forwarding instruction.

A SRP can be encoded using any combination of SIDs (*i.e.*, local or global). As long as the nodes that the packet traverses own a forwarding instruction to reach the egress node. In this work, we focus on the use of two SIDs types: Node-SID and Adj-SID, we do not consider other SID types such as service SID, BGP peering SIDs, etc.

The two proposed SRP encoding algorithms produce a label stack composed of two SID types: Node-SID and Adj-SID. Each SID has a pre-installed forwarding plane instruction associated with it as follows:

- Node-SID: it is a label associated with the SR node *i.e.*, attached to the loopback address. When a SR node receives a packet with a Node-SID as a top label, it forwards it on the IGP shortest path to reach the node that owns that Node-SID, the node owning the Node-SID pops the label before inspecting the next label in the stack.
- Adjacency SID (Adj-SID): it is the label attached to an IGP adjacency *i.e.*, the interface to reach the neighbor router. It is used to enforce packet forwarding through a specific exit interface. By default, an Adj-SID is advertised as a local segment, it can also be advertised as global if desired.

## IV. Segment Routing path Encoding

The SRP length varies depending on the network diameter, QoS requirements, and network resources availability. Accordingly, the label stack to express a SRP can be very big, resulting in a label stack size that may exceed the ingress MSD. This causes the incapacity of using such a path, and other side effects such as preventing the use of other label types. Therefore, an efficient encoding algorithm is required to minimize the size of the label stack.

A source routed path may be strict or loose. A path is strict if all the links and nodes that the packet has to go through are listed in the packet header, whereas a path is loose if only a subset of the links and nodes that packet has to go through are listed in its header. SRPs can be expressed exclusively with Node-SIDs, local Adj-SIDs, Global Adj-SIDs or a combination of those SID types. In this paper, a SRP is strict if it is encoded using only Adj-SIDs. Otherwise, it is considered as loose:

- A SRP encoded exclusively with Node-SID or a combination of Node-SIDs and Adj-SIDs is a loose path. Two successive Node-SIDs in the label stack can be separated by one or more network nodes. The label stack expresses the initial path in the current state of the network. However, if the IGP metric between two SR nodes changes, the label stack will not represent the initial path anymore.
- A SRP encoded with local Adj-SID is a strict path, because the Adj-SIDs are local to the nodes advertising them have the forwarding entries associated with them. Therefore, the packet has to go through only the nodes that own the Adj-SIDs.
- A SRP encoded with global Adj-SIDs can be a strict or a loose path: strict if all the links that the packet has to go through are listed in the label stack, loose only if a subset of the links is listed.

We propose two SRP path encoding algorithms. The computed label stack is a combination of Node-SIDs and Adj-SIDs. Recall that in current SR deployments, Adj-SIDs are advertised as local segments. In that case, we use the SR paths Label Encoding Algorithm (SR-LEA) to compute the minimum label stack. However, as stated in the standards, Adj-SIDs can also be advertised as global segments. Therefore, the minimum label stack is computed using the SR-LEA-A Algorithm.

Let us consider the topology detailed in Fig. 1. All the nodes allocate the same SRGB: [1000, 2000]. The computed path to satisfy the Quality of Service (QoS) requirements for the traffic sent by CE1 to CE2 is *P*: $PE1 \rightarrow P2 \rightarrow P3 \rightarrow P7 \rightarrow P6 \rightarrow PE5$. *P* has to be encoded as a stack of labels then pushed by $PE1$ onto flow's packets. In what follows we detail the different encoding algorithms:

### A. Strict Encoding

A strict encoding of the SRP is the worst case scenario, as it generates the maximum label stack to encode a SRP. Two approaches may be applied:

- Using exclusively Node-SIDs to encode a SRP. Replace each node in the SRP by its Node-SID. This approach suffers from the same problem as in [10]. In fact, the resulting label stack expresses the requested SRP only if the shortest path between all the neighbors in the path is via the direct link. For example, a strict encoding of path *P* results in the following label stack: {*Node-SID PE1, Node-SID P2, Node-SID P3, Node-SID P7, Node-SID P6, Node-SID PE5*}. However, this label stack does not express path *P*. The packets at $P3$ will not be sent to $P7$ over the direct link because it is not the shortest path.
- Using exclusively Adj-SIDs to encode a SRP. At each node the exit interface is replaced with the associated Adj-SID, this produces a label stack that corresponds to requested path. As shown in Fig. 2, a strict encoding of path *P* results in the following label stack: {*Adj-SID PE1-P2, Adj-SID P2-P3, Adj-SID P3-P7, Adj-SID P7-P6, Adj-SID P6-PE5*} $= [5012, 5023, 5037, 5076, 5065]$. Each node pops the Adj-SID that it owns before forwarding the packet through the chosen interface.

Strict encoding can be essential to accomplish certain tasks such as Operations, Administration, and Maintenance (OAM) [12]. For example, to monitor a specific path when ECMPs exist. The reference topology Fig. 1 is composed of 8 nodes. However, a SP network can be composed of hundreds or even thousands of nodes. Consequently, using strict encoding especially for long paths is not always possible as it violates the MSD constraint, also it adds a considerable overhead to packets.

### B. SR-LEA Algorithm

We propose the SR-LEA algorithm, for computing the minimum label stack to encode a SRP. It is used if the Adj-SID are advertised as local segments. The algorithm takes the initial path expressed as a list of IP addresses. The initial path can be imposed manually or computed by a centralized entity such as a Software Defined Network (SDN) controller [13] [14] or by a Path Computation Element (PCE) [15] [16]. SR-LEA makes use of existing IGP shortest paths, which are installed as forwarding instructions by the SR-MPLS control plane. The resulting label stack is a combination of Node-SIDs and local Adj-SIDs. It represents exactly the initially computed path in the current state of the network.

**Algorithm 1** Efficient Label Encoding algorithm

**INPUT:** The SRP expressed as a list of IP addresses
**OUTPUT: labelStack** the SRP minimum label stack.
**Initialization:**
$G$: Graph of the network topology
$A = \{\ \}$: Holds the list of the SRP subpaths.
$B = [\ ]$: Used to construct a subpath, when no IP addresses can be added it is moved to $A$.
$SPF = Dijkstra(SRP[1],\ SRP[end])$: The shortest path between the source and destination of the SRP.
$labelStack = [\ ]$

**STEP 1:** Computation of the SRP subpaths.

```
1:  $i = 1$: Points to the current node of the SRP.
2:  $k = length(SRP)$ : Points to the last node of the candidate subpath.
3:  while $i <= length(SRP)$ do
4:      $push(B,\ SRP[i])$
5:      if $i == length(SRP)$ then
6:          $push(A,\ B)$
7:      else if $B \nsubseteq SPF$ then
8:          if $length(B) == 2$ then
9:              if $k > i$ then
10:                 $k--$
11:                 $B = B[1]$
12:                 $SPF = Dijkstra(G,\ B[1],\ SRP[k])$
13:                 $continue$
14:             else
15:                 $push(A,\ B)$
16:                 $B = B[end]$
17:                 $SPF = Djikstra(G,\ B[1],\ SRP[k])$
18:             end if
19:         else
20:             $push(A,\ B[1 : end - 1])$
21:             $SPF = Djikstra(G,\ B[end - 1],\ SRP[k])$
22:             $B = [\ ]$
23:             $i--$
24:             $continue$
25:         end if
26:     end if
27:     $i++$
28:     $k = length(SRP)$
29: end while
```

**STEP 2:** The construction of the label stack.

```
30: for $i \leftarrow 1$ To $Size(A)$ do
31:     if $length(A[i]) > 2$ then
32:         $push(labelStack,\ NodeSID(A[i][end]))$
33:     else
34:         $push(labelStack,\ AdjSID(A[i]))$
35:     end if
36: end for
```

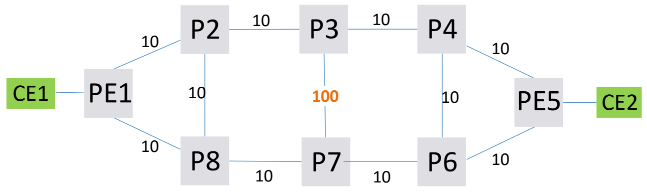


Fig. 1: Reference network topology, all the links costs is 10 except the link P3-P7 is attributed cost 100.

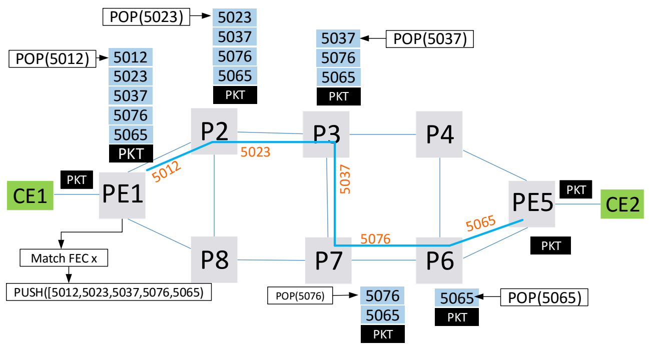


Fig. 2: The SRP to connect CE1 and CE2 is expressed as a label stack using the strict algorithm.

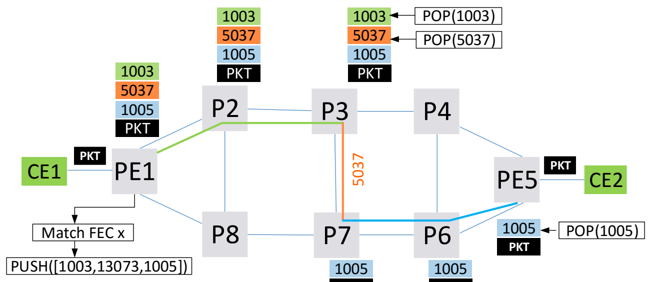


Fig. 3: The SRP to connect CE1 and CE2 is expressed as a label stack computed using the SR-LEA algorithm.

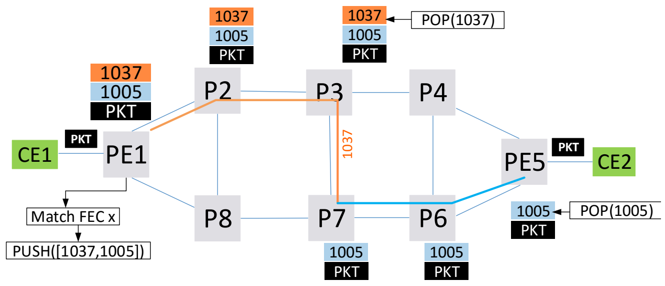
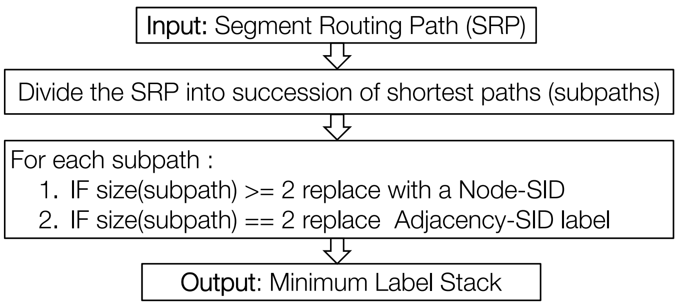


Fig. 4: The SRP to connect CE1 and CE2 is expressed as a label stack computed using the SR-LEA-A algorithm.



Input: Segment Routing Path (SRP)
Divide the SRP into succession of shortest paths (subpaths)
For each subpath :
1. IF size(subpath) >= 2 replace with a Node-SID
2. IF size(subpath) == 2 replace Adjacency-SID label
Output: Minimum Label Stack

Fig. 5: SR-LEA flowchart.

SR-LEA has two main steps, as shown in Fig. 5 and detailed by the pseudocode in Algorithm 1. In the first step, the SRP is spliced into a succession of shortest paths (subpaths), container $A$ holds the SRP splices, whereas the container $B$, will hold the potential SRP splice. In the second each subpath composed of three or more nodes is replaced by its tail's end node Node-SID, whilst if it is composed of two nodes it is replaced by the Adj-SID between those two nodes. The best case is that the requested SRP follows the shortest path SPF. Consequently, SR-LEA output a label stack composed of one label: the egress node's Node-SID.

In order to encode the path $P$ using SR-LEA, we follow the two steps of the algorithm. First, the subpath that compose the path $P$ are computed and saved in the list $A$: $\{(PE1,P2,P3),(P3,P7),(P7,P6,PE5)\}$ . Finally each subpath in $A$ is replaced with the appropriate SID:

- The subpath $\{PE1,P2,P3\}$ is composed of three nodes. Therefore, it is replaced by $P3$'s Node-SID = 1003.
- The subpath$(P3,P7)$ is composed of two nodes. Therefore, it is replaced by the Adj-SID $P3$-$P7$ = 5037.
- The subpath $\{P7,P6,PE5\}$ is composed of three nodes. Therefore, it is replaced by $PE5$'s Node-SID = 1005.

The resulting label stack is [1003, 5037, 1005]. As shown in Fig. 3, a packet follows the IGP shortest path to reach $P3$ using label 1003 (*i.e.*, P3's Node-SID). At P3, the Adj-SID 5037 is used to enforce the packet through the link *P3-P7*. At $P7$, label 1005 (*i.e.*, PE5's Node-SID) is used to forward the packet down the IGP shortest path to reach $PE5$. At PE5, label 1005 is popped and the IP packet is forwarded to $CE2$.

## C. *SR-LEA-A*

In the segment routing architecture, it is possible to advertise an adjacency (*i.e.*, an interface) as a global segment, rather than advertising it as a local segment. Accordingly, the adjacency becomes routable in the SR domain. In comparison to the local Adj-SID, all the SR nodes forward the packet using the IGP shortest path to reach the node that advertises the global Adj-SID, then the node that owns the adjacency forwards the packet to the exit interface associated with the global Adj-SID. To take advantage of this possibility, we propose SR-LEA with global Adj-SIDs (SR-LEA-A). When Adj-SIDs are advertised as global segments it is the SR-LEA-A that computes the minimum label stack.

In SR-LEA-A, we suppose that the Adj-SIDs are advertised as global segments, the resulting label stack is either smaller or equal to the SR-LEA's one. Both algorithms share step 1 detailed in Algorithm 1. In SR-LEA-A, as detailed by the pseudocode in Algorithm 2: a subpath of $size \geqslant 3$ followed by one of $size = 2$ are encoded using one label: the global Adj-SID between the last node in the first path and the first node in the second one. Compared to SR-LEA, two labels are used to encode the two subpaths.

**Algorithm 2** Efficient Label Encoding algorithm with global Adj-SIDs

**STEP 1** Same as for SR-LEA

**STEP 2**

```
for $i \leftarrow 1$ To $Size(A)$ do
  if $length(A[i]) > 2$ then
    if $length(A[i+1]) == 2$ then
      $push(labelStack, GlobalAdjSID($
      $A[i][end],\ A[i+1][1]))$
      $p+ = 2$
      $continue$
    end if
    $push(labelStack, NodeSID(A[i][end]))$
  else
    $push(labelStack, AdjSID(A[i]))$
  end if
end for
```

In the example described in Fig. 4, P3 advertises its adjacency with P7 as the global SID 1037, the list $A$ contains the following subpaths: $\{(PE1, P2, P3)$, $(P3, P7)$, $(P7, P6, PE5)\}$. Accordingly, the two subpaths $\{(PE1, P2, P3), (P3, P7)\}$ are encoded using the global Adj-SID $P3 - P7$ : 1037. Consequently, the label stack for the path *P* is [1037, 1004]. At PE1 and P2, based on 1037 the packet is forwarded down the shortest path to reach P3. At P3, the top label 1037 is popped and the packet forwarded through the interface that connects P3 to P7. At P7, based on the $PE5$'s Node-SID (*i.e.*, 1005) the packet is forwarded through the shortest path to reach $PE5$.

## V. Simulation Results

In order to better evaluate the performance of the proposed algorithms, we experimented on several SNDlib network topologies [17] [18]. To get a representative set of paths, for each topology, we consider a sample bandwidth demand matrix D. As detailed in Table I, we solve the multicommodity flow problem [19]. The result is the optimal set of paths to satisfy the demand matrix. The paths are then encoded using the strict Adj-SID, SR-LEA and SR-LEA-A.

The two proposed algorithms, compute the minimum label stack to express a SRP. SR-LEA is used when the Adj-SIDs are local segments whilst SR-LEA-A is used when they are global.

| Topology | V | E | D |
|---|---|---|---|
| Geant | 22 | 36 | 431 |
| Albilene | 12 | 18 | 131 |
| Brain | 161 | 166 | 9045 |
| Germany50 | 50 | 80 | 1270 |
| Nobel-germany | 17 | 26 | 248 |

TABLE I: V is the number of nodes. E the number of links. D: number of demands in the demand matrix.

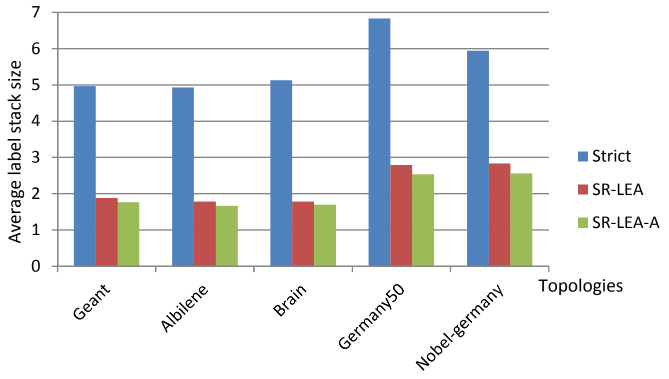


Fig. 6: Comparison of the average label stack size generated using a strict encoding, SR-LEA and SR-LEA-A algorithms.

The comparison is made between the strict encoding, the SR-LEA and the SR-LEA-A algorithms. For each topology, using the three encoding algorithms detailed previously, we compute the average label stack size and the percentage of network paths encoded with a label stack $size \leq MSD$.

Fig. 6 illustrates the per-topology average label stack size variation depending on the topology and the encoding algorithm.

- We observe that the strict encoding always produces a large label stack. This was expected because no optimization on the label stack size is performed, rather a one to one mapping of the physical links to the label stack. We note that for some paths the label stack noticeably reaches up to 14 labels.
- SR-LEA reduces the size of the label stack by 52% to 65% compared to the strict encoding; the observed gain varies depending on the network design and diameter.
- SR-LEA-A gives the best results. Notably, compared to the strict encoding, the average label stack size is reduced by 57% to 67%.

The MSD corresponds to the maximum number of labels a router can push onto packet header, it is a local characteristic of a router, it varies from one equipment vendor to another. In an architecture where the path computation is delegated by the SR node to a centralized entity such as a SDN controller or a PCE. The node's MSD is learned via the Path Computation Element Protocol (PCEP) extensions for SR [8]. Hence, this limitation is taken into consideration in the path computation process. This limitation makes long paths in the network unusable. Consequently, it forces the network traffic to follow only short paths which cause inefficient traffic distribution or worse network congestion. For this study, we fixed the MSD

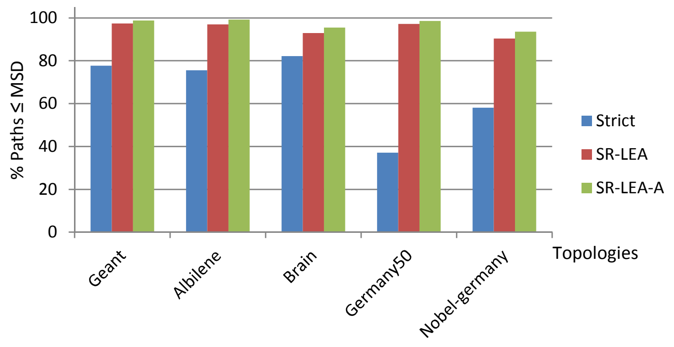


Fig. 7: Paths expressed with a label stack size $\leq MSD$ $(MSD = 5)$.

to 5 labels, which is the value announced currently by the major equipment vendors.

Fig. 7, illustrates the variation of the percentage of the useable paths in each topology. With a strict encoding, the percentage of useable paths can be very low *e.g.*, 37% for $Germany50$ topology. Using SR-LEA, increases considerably the amount of useable paths *e.g.*, from 37% to 97% for $Germany50$ topology. However, encoding the label stack using SR-LEA-A gives the best results, as it increases the number of usable paths from 37% to 99%, a gain of 2% to 4% more than SR-LEA. We expect the difference to be more considerable on topologies with bigger diameters.

We conclude that the proposed algorithms are very efficient in reducing the label stack size, also in minimize considerably the impact of the MSD limitation. However, both algorithms do not completely eliminate the MSD problem, as we still have paths that can not be expressed with a label stack smaller than the MSD.

## VI. Conclusion

In this work, we proposed two SR-MPLS paths label encoding algorithms, namely SR-LEA and SR-LEA-A. Both algorithms compute the minimum label stack to express a segment routing path. Their performance has been evaluated over real topologies. In addition, we prove that they are efficient in alleviating the impact of the MSD. For future work, a PCE implementation of the proposed algorithms is underdevelopment. We are considering the possibility to use the two algorithms to encode Topology Independent Loop-Free Alternate (TI-LFA) Fast Reroute post-convergence paths.

## References

[1] C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-spring-segment-routing-09, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-segment-routing-09

[2] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[3] S. Previdi, C. Filsfils, B. Field, I. Leung, J. Linkova, E. Aries, T. Kosugi, E. Vyncke, and D. Lebrun, "IPv6 Segment Routing Header (SRH)," Internet Engineering Task Force, Internet-Draft draft-ietf-6man-segment-routing-header-01, Mar. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-01

[4] P. Psenak, S. Previdi, C. Filsfils, W. Henderickx, J. Tantsura, H. Gredler, and R. Shakir, "OSPF Extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-ospf-segment-routing-extensions-08, Apr. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ospf-segment-routing-extensions-08

[5] S. Previdi, C. Filsfils, A. Bashandy, S. Litkowski, J. Tantsura, B. Decraene, and H. Gredler, "IS-IS Extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-isis-segment-routing-extensions-06, Mar. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-isis-segment-routing-extensions-06

[6] S. Previdi, P. Psenak, C. Filsfils, H. Gredler, M. Chen, and J. Tantsura, "BGP Link-State extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-gredler-idr-bgp-ls-segment-routing-ext-01, Dec. 2015, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-gredler-idr-bgp-ls-segment-routing-ext-01

[7] Y. Rekhter, A. Conta, G. Fedorkow, E. Rosen, D. Farinacci, and T. Li, "MPLS Label Stack Encoding," RFC 3032, Mar. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc3032.txt

[8] S. Sivabalan, J. Medved, C. Filsfils, V. Lopez, J. Tantsura, W. Henderickx, E. Crabbe, and J. Hardwick, "PCEP Extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-pce-segment-routing-07, Mar. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-segment-routing-07

[9] K. Kompella, S. Sivabalan, S. Litkowski, R. Shakir, S. Kini, and jefftant@gmail.com, "Entropy labels for source routed tunnels with label stacks," Internet Engineering Task Force, Internet-Draft draft-ietf-mpls-spring-entropy-label-03, Apr. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-mpls-spring-entropy-label-03

[10] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[11] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *Optical Network Design and Modeling (ONDM), 2015 International Conference on*. IEEE, 2015, pp. 34–38.

[12] R. Geib, C. Filsfils, C. Pignataro, and N. Kumar, "A Scalable and Topology-Aware MPLS Dataplane Monitoring System," Internet Engineering Task Force, Internet-Draft draft-ietf-spring-oam-usecase-03, Apr. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-oam-usecase-03

[13] 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 Communications Conference and Exhibition (OFC), 2015*. IEEE, 2015, pp. 1–3.

[14] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano, "Traffic engineering with segment routing: Sdn-based architectural design and open source implementation," in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, 2015, pp. 111–112.

[15] J. Medved, I. Minei, E. Crabbe, and R. Varga, "PCEP Extensions for Stateful PCE," Internet Engineering Task Force, Internet-Draft draft-ietf-pce-stateful-pce-14, Mar. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-stateful-pce-14

[16] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Sdn and pce implementations for segment routing," in *Networks and Optical Communications-(NOC), 2015 20th European Conference on*. IEEE, 2015, pp. 1–4.

[17] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009. [Online]. Available: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz

[18] ——, "SNDlib 1.0–Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010. [Online]. Available: http://www3.interscience.wiley.com/journal/122653325/abstract

[19] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.