# A New Method For Encoding MPLS Segment Routing TE Paths

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/IMT Atlantique/Adopnet Team, Rennes, France, geraldine.texier@imt-atlantique.fr

***Abstract*—Segment Routing (SR) architecture has a great potential to replace the MPLS control plane. It simplifies considerably the operation and management of the MPLS networks. A Segment Routing path does not require signaling because it relays on the source routing paradigm, where the path description is directly encoded into the packet's header as a label stack. This has a direct consequence on the size of the label stack which increases linearly with the length of the path. Unfortunately, such approach runs into the routers physical limitation known as the Maximum Stack Depth (MSD), that bounds the maximum number of labels a router can push onto packets. Consequently, it prevents traffic to flow on some of the network paths, leading to underutilization of network resources. Therefore, the MSD restrains the adoption of Segment Routing as it impacts the service provider ability to perform traffic engineering. Several algorithms have been proposed to mitigate the impact of the MSD. They usually rely on an optimization of the SR paths encoding. However, none of them eliminates the impact of the MSD limitation.**

**In this work, we propose a path segmentation approach to definitively eliminate the impact of the MSD. Accordingly, all the possible paths in the network may be considered to forward traffic. This approach is based on the introduction of a new type of Segment Identifiers (SID)s called Targeted SID (TSIDs). We detail the architectural requirements and propose an optimization algorithm to reduce the introduced overhead.**

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

## I. Introduction

With its great potential to simplify considerably the operation and the management of MPLS networks, the Segment Routing (SR) architecture appears as a serious alternative to the MPLS control plane. SR leverages the source routing paradigm, where the path description is carried in the packets header. Therefore, in MPLS networks implementing SR, signaling protocols such as LDP and RSVP-TE are not required anymore. In fact, SR relies on adding extensions to already deployed routing protocols such as OSPF and ISIS. Consequently, Service Providers (SP) can deploy SR without extra investment on new hardware, as it can be enabled on the routers that are currently in production with a simple software update. The Segment Routing (SR) architecture, as standardized by the IETF SPRING working group, can be instantiated over the MPLS (SR-MPLS) and IPv6 (SR-IPv6) data planes and offers a simple control plane. With SR, per-flow states have to be maintained only by the ingress nodes, relieving the transit nodes that solely maintain SR information. A segment, the main building block of the SR architecture, is associated with a forwarding plane instruction. In SR-MPLS, a segment is a 20 bits label and therefore it is manipulated using the MPLS forwarding plane operations (i.e., POP, PUSH, SWAP).

A Segment Routing Path (SRP) is composed of a succession of Segment Identifiers (SID)s. In SR-MPLS, the SRP is encoded as a label stack. In Traffic Engineering (TE), multiple Quality of Service (QoS) constraints are used for path computation. The computed path to carry clients traffic may not follow the shortest path. Consequently, the size of the label stack growth linearly with the longer of the path. However, currently available routers have a limitation on the number of labels that they can push onto a packet header. As a consequence, it reduces the number of paths potentially operable in the network, leading to poor network resources utilization. This limitation is known as the Maximum Stack Depth (MSD). It comes from the fact that in order to reach wire-speed packet processing, vendors implement MPLS forwarding operations in Application Specific Integrated Circuits (ASIC)s. Processing packets in hardware is more efficient than software, at the expense of capacity. Today, this limitation is low *e.g.*a maximum of 5 to 10 labels is currently supported by some routers.

In this paper, we study the use of SRPs fragmentation to tackle the MSD limitation. For that purpose, we define the Targeted SID (TSID) a new segment type that is attached/assigned to a slice of the SRP. TSIDs role is to reduce the size of the label stack to express a SRP. The underlying idea is to replace multiple labels in the initial stack by a TSID label. Then, when a packet reaches a specific node, the TSID label on the top of the label stack is substituted by the sequence of labels it has replaced initially. Consequently, TSIDs have to be pre-installed in the network before traffic is forwarded on the SRP. In this article, we prove that SRPs fragmentation is an effective method to bypass the MSD limitation. This reinstates the possibility to consider any available topological path and thence empowers a better network resource utilization. To achieve our goals, we propose an optimization algorithm to reduce the number of installed TSIDs, then we compare the proposed algorithm to the results of the offline linear programming model.

978-1-5386-0554-7/17/$31.00 ©2017 IEEE

In the proposed architecture, TSIDs may be installed anywhere in the network with the help of a Path Computation Element (PCE) server. Therefore, the Service Provider have to enable Path Computation Clients (PCC)s on transit nodes in addition to Provider Edge (PE)s nodes. However, this increases the number of Path Computation Element Protocol (PCEP) sessions the PCE has to maintain. For that purpose, we also propose an optimization algorithm to reduce the number of PCEP sessions. We compare the proposed algorithm to the results of the offline linear programming model.

## II. Related Work

In the literature, several algorithms have been proposed to efficiently encode SRPs [1][2][3]. Their focus is to minimize the number of labels used to encode a SRP, mainly by the combination of different SID types. Indeed, in Segment Routing each SID corresponds to a forwarding behavior. For example, using a Node-SID forces the traffic to use the shortest path to reach a designated node whereas using an Adjacency SID constrains the traffic through a specific interface on a node.

Encoding algorithms slacken the impact of the MSD limitation. However, none of the proposed algorithms solves totally the MSD problem. In particular, all the proposed algorithms produce a label stack that expresses the SRP as a loose path. Indeed, those algorithms consider that it is not necessary to express in detail all the path if parts of the path follow the default route computed by the Shortest Path First (SPF) algorithm of the routing protocol. However. expressing a SRP as loose makes the SRP very sensitive to the network nodes routing tables changes triggered by events that engender default routes recomputation. For example, a link weight modification, a link or node failure. etc. In such events, to continue to express correctly the SRPs, those algorithms must be re-run for all the paths. Such behavior is not sustainable especially in large networks where changes are frequent continuously triggering SPF computations.

For all these reasons, we propose a new approach for reducing the SRP label stack while keeping the possibility to continue to express a strict path. To the best of our knowledge, this work is the first to use the path segmentation approach to eliminate totally the MSD limitation. This is a standalone approach yet it can be combined with an efficient encoding algorithm such as those previously discussed.

## III. Path Segmentation

In SR, each SID identifies a topological path, but the SR architecture offers several types of SIDs to compose an end to end SRP: a Nodal Segment (Node-SID) is used to forward a packet along the IGPs shortest path towards the associated node, whereas the Adjacency Segment (Adj-SID) forces the packets forwarding through a specific interface. Traffic Engineering, clients QoS requirements enforcement, path diversity, etc. May require the selection of paths that are usually not preferred by the IGP. However, the corresponding label stack to implement such SRP in SR-MPLS may be greater than what is allowed by the ingress nodes MSD.

We propose the path segmentation approach, where the initial label stack to express SRP is fragmented into multiple stacks, each sub-stack is replaced with a new type of segment named the Targeted SID (TSID). We create as much TSIDs as required to obtain a label stack size less than or equal to the MSD. A TSID is related to a specific label stack which encodes a topological path and is installed on specific network nodes. The TSID, like the Adj-SID, is local to a node, and takes its value outside the Segment Routing Global Block (SRGB). The TSID is assigned to a push operation which replaces the TSID label by a specific label stack. When the packet reaches the node that owns the TSID, (*i.e.*the top label is equal to the TSID), the TSID gets popped and the associated stack is pushed.

For illustration purposes, let us consider the network depicted in Fig. 1. A client requests a connection of 100 MB of bandwidth to connect two of its sites *CE1* and *CE2*, the ingress edge router for the requested path is PE1 and PE2 is the egress. The computed path that satisfies the requested bandwidth is *Pth1*: P1,P7,P12,P13,P14,P11,P10,P4. Moreover, the service provider implements the SRP strict encoding where all the intermediate node's Node-SIDs are listed in the label stack. Consequently, the *Pth1* get encoded with the following label stack: [1,7,12,13,14,11,10,4]. If PE1 has a MSD of 5, then PE1 would not be able to push*Pth1* stack onto the client packets. In our approach, a TSID can be used to replace a slice of Pth1. For example, replace the slice *Pth*:[12, 13, 14, 11] with *TSID1*. Therefore, *Pth1* is encoded as follows: [1, 7, TSID1, 10, 4]. As shown in Table. I, a new entry in *P7*s Label Forwarding Information Database (LFIB) has to be pre-installed before *Pth1* is installed on PE1 to avoid that packets get dropped by *P7*.

TABLE I: P7's LFIB

| Incoming label | Operation | Exit Interface |
|---|---|---|
| TSID1 | POP(TSID1) & PUSH([12,13,14,11]) | 7-12 |

### A. Targeted SID Architecture

The SR architecture main design guidelines' goals are to install fewer states in the network nodes and keep the control plane very light. In fact, no signaling is needed for the SRP as they are carried in the packets headers. Therefore, the signaling protocols RSVP-TE and LDP are removed. The IGPs (OSPF and ISIS) have been extended to exchange SR information. Consequently, no additional protocols have been defined in order to enable SR. However, the RSVP-TE in traditional MPLS is responsible for updating network resources such as available and reservable bandwidth on the network links. Without such signaling, SR is not able to trigger updates of network resources availability. However, an up to date network resources capacity is essential to to do Traffic Engineering. To overcome this limitation, Service Provider could used a

978-1-5386-0554-7/17/$31.00 ©2017 IEEE

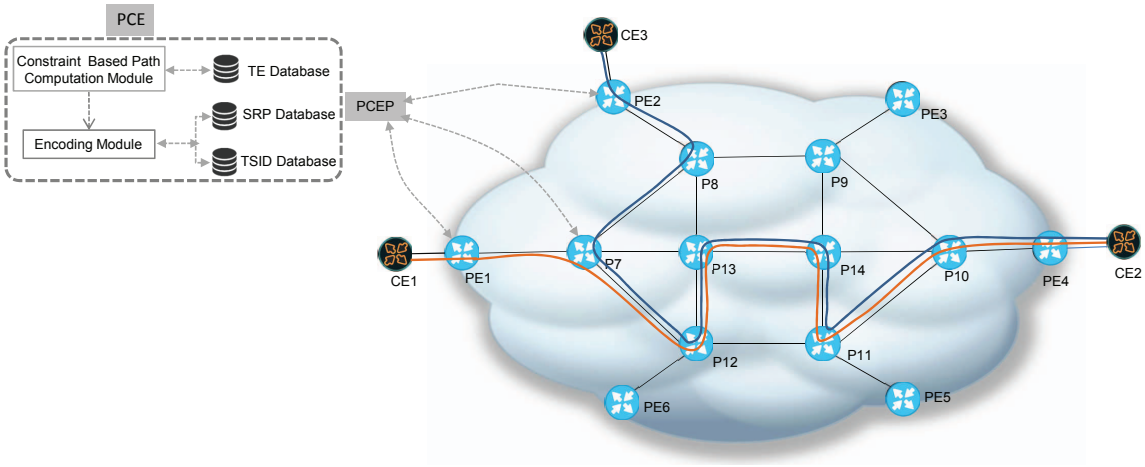



Fig. 1: Network Topology.

centralized controller that maintains the network resources in its local TE Database (TED). The architecture depicted in Fig. 1 details such solution that is based on a Path Computation Element (PCE) server.

Several architectural components need to be defined to enable path segmentation. To address how the TSIDs are computed and how they can be installed into the network. It makes sense that the PCE installs also the TSIDs, in addition to the SRP. Consequently, the PCE has to maintain a TSID database in order to be able to reuse previously installed TSID for future SRPs. In this proposed architecture, all network nodes implement the Path Computation Element Clients (PCC). When a request reach the PCE, the constraint based path computation (*e.g.*Constraint Shortest Path First, CSPF) module computes the path based on the requested parameters and the information contained in the TED. As depicted in 1, the computed path is then sent to the encoding module which decides if a TSID is required or not.

The TSID approach requires the standardization of some of its components in order to ensure inter-vendor interoperability. Recently, PCE protocol (PCEP) has been extended to support SR. in fact, new SR Type Length Values (TLV) have been defined in [4]. The Explicit Route Object (ERO) could carries the label stack to express a SRP. We propose to reuse the defined TLVs to carry the TSID label stack. Because the installation of TSIDs has to be initiated by the PCE, we propose to extend the mechanism described in [5] to add the support of PCE initiated TSIDs. The TSID value is a local label. Therefore, it is up to the node that installs the TSID label stack to allocate the TSID value. Indeed, as the TSID label is taken outside the SRGB, it is easier to let the node pick its value inside its label pool instead of letting the PCE allocate a label value that could be outside the local label pool or already in use by another protocol. However, no reporting mechanism is currently defined to let the PCC reports to the PCE the label value it has associated to the TSID. Accordingly, that requires standardization efforts to detail such mechanism.

In addition, service providers may choose to advertise the TSIDs in the network. Therefore, we propose to define a new sub-TLV similar to the SID/Label Binding Sub-TLV defined in [6]. In this scenario, the PCE may not be the only entity responsible for SRPs computation. For example, network nodes may have their own CSPF computation engine. Consequently, the TSIDs need to be advertised in IGP so that other nodes can use them. Also, in case of PCE failure, the advertisement of TSIDs help to recover the state of the network by listening to the IGP. However, this approach adds new states in the network which segment routing precisely tries to reduce. This scenario should be avoided in Software Defined Networks (SDN).

## IV. Offline TSID Placement Models

SR-MPLS nodes maintain considerably fewer states compared to traditional MPLS. However, the proposed path segmentation approach adds an overhead to the SR architecture. TSIDs are additional entries in the node's forwarding table. Each node may have to maintain TSID database if the TSIDs are advertised in the IGP. Also, in the proposed architecture, TSIDs are installed via the PCEP protocol that increases the number of PCEP sessions that the PCE have to maintain. Indeed, in a traditional IP/MPLS networks, the PCEP sessions are established between the PCE and the edge nodes *i.e.*PEs. In our approach, additional PCEP sessions must be established between the PCE and core *i.e.*PE nodes in order to install TSIDs. In this work, in addition to the proposition of the TSID mechanism and the architecture that enables it, we aim to solve two following optimization problems:

- To reduce the global number of installed TSIDs,
- To reduce the number PCEP sessions the PCE has to maintain.

In this section, we present two offline Linear Programming (LP) models. Both models take a set of paths in input and require the existence of a traffic matrix. In fact, a realistic set of paths is generated by solving the multi-commodity flow problem for a given network and a given traffic matrix. The proposed models have been used as a benchmark for the more practical online algorithms with unknown traffic matrices. In addition, if a Service Provider has the traffic matrix and wants

978-1-5386-0554-7/17/$31.00 ©2017 IEEE

to migrate its network to Segment Routing, the two offline models may be used to assist the transition.

### A. Offline Optimization of TSIDs Placement

The first offline LP model (1) computes the minimum number of TSIDs to install for a given set of paths. For simplicity purposes, we suppose that all the network nodes have the a MSD of 5 labels, this MSD value represents what is currently supported by some routers. This model can still be extended for per-node MSD case at the expense of increased computation.

We denote the set of paths that satisfy the traffic matrix and that are encoded with a label stack greater than the MSD by $\mathcal{P}$. $\mathcal{T}$ denotes the set of all possible TSIDs without duplication generated from $\mathcal{P}$. It is worth mentioning that it is possible to have two or more different TSIDs associated with the same label stack, because they are installed on different nodes. For example, in our sample network, the label stack composed of three Node-SIDs [P13,P14,P11] can be installed on different nodes: P7,P8 and P12 and therefore considered as three different TSIDs and not just one. $\mathcal{T}_p$ denotes the set of TSID that can be used for the path $p$. $\alpha_{lt}$ equals to 1 if the label $l$ is used in TSID $t$ and 0 otherwise. $s_p$ denotes the size of path's $p$ label stack. $s_t$ denotes the size of the TSID's $t$ label stack. $f_{pt}$ is a binary variable, it takes the value 1 if the path $p$ uses TSID $t$ and 0 otherwise. $\hat{f}_t$ is a binary variable, it takes the value 1 if the TSID $t$ is chosen to reduce at least one path and 0 otherwise.

$$\textbf{Minimize} \sum_{t \in \mathcal{T}} \hat{f}_t \tag{1a}$$

**Subject to :**

$$\hat{f}_t \geq f_{pt} \qquad \forall p \in \mathcal{P}, \forall t \in \mathcal{T} \tag{1b}$$

$$s_p - \sum_{t \in \mathcal{T}_p} f_{pt} * (s_t - 1) \leq MSD \qquad \forall p \in \mathcal{P} \tag{1c}$$

$$\sum_{t \in \mathcal{T}_p} f_{pt} * \alpha_{lt} \leq 1 \qquad \forall p \in \mathcal{P}, \forall l \in p \tag{1d}$$

The objective function (1a) minimizes the sum of $\hat{f}_t$ (*i.e.*, the total number of used TSIDs). The Equation (1b) ensures that a TSIDs is computed once even it is used to reduce multiple paths. Equation (1c) ensures that the TSIDs used for a path results in a label stack size less than the MSD, keeping in mind that a TSID reduces the size of the path by its size plus 1. For example, a SRP label stack composed 8 labels can not be reduced by a TSID stack of 3 labels because the resulting stack would be 6 labels, as an additional label has to be added to identify the TSID. Equation (1d) ensures that no label appears more than once in the TSIDs used to reduce a path. In fact, the intersection of a solution's TSIDs must be avoided as it leads to the creation of traffic loops.

### B. Offline Minimization of PCEP sessions

The TSID architecture as depicted in Fig, 1 requires that the all the network nodes become PCCs (*i.e.*, edge and core routers). Thus, all the nodes are able to install TSIDs. However, SP tends to enable PCCs only on the border of the network, *i.e.*, PE routers. The increase in the number of PCEP sessions a PCE has to maintain could lead to scalability issues. Accordingly, the performance of the proposed architecture needs to be evaluated not only based on the number of installed TSIDs but also on the required number of PCEP sessions. A SP may estimate that it is more important to reduce the number of PCEP sessions instead of minimizing the number of TSIDs. We encourage this approach for large networks, where number of core nodes is greater than the edge nodes, especially if the TSIDs are not advertised by the IGP. A side effect of this approach is that TSIDs may be concentrated at certain network nodes. Consequently, in the case of a node failure, a considerable amount of path will be affected especially that no fast re-route like mechanism is defined for the TSID approach.

The offline LP model (2) minimizes the number of the PCEP sessions required to install TSIDs. We used this model to benchmark the online PCEP minimization algorithm. The objective function depicted in (2a) minimizes the network nodes that have to be a PCC. In (2a) $k_n$ is a binary variable, it is equal to 1 if the node $n$ is used to install TSIDs and 0 otherwise. In addition to the constraint depicted in (2b), the LP model (2) is subject to the same constraints as the LP model (1). $\zeta_{n,t}$ denotes where the TSID $t$ has to be installed, it is equal to 1 if the node n is used to install the TSID t and 0 otherwise.

$$\textbf{Minimize} \sum_{n \in \mathcal{V}} k_n \tag{2a}$$

**Subject to:** (1b),(1c),(1d)

$$k_n \geq \hat{f}_t * \zeta_{n,t} \qquad \forall p \in \mathcal{P}, \forall t \in \mathcal{T} \tag{2b}$$

### C. Online Algorithms

Delivering QoS using segment routing requires the use of a centralized controller (*e.g.*, PCE or SDN controller). In an online environment, the SP does not have the full demand matrix. Therefore, the connection demands are treated by the controller one by one, each demand is received as a source, destination and the QoS requirements. A path that respects those requirements is computed by the optimization engine and then passed to the encoding engine. If the path is encoded with a label stack greater than the demand's source node MSD, it gets invalidated. Consequently, the computation of another

path is triggered, in absence of other paths the demand is rejected.

In this section, we present two online algorithms, referred in the next sections as OTO for Online TSIDs Optimization:

- OTO for TSID minimization, favors the reutulization of existing TSIDs and creates new ones only if there is no solution to reduce the requested path with the available TSIDs in the TSIDs Database.
- OTO for PCEP session minimization, favors the solutions that require the installation of TSIDs on the nodes that maintain an active PCEP session with PCE, also by reusing exiting TSIDs.

The OTO algorithm is composed of 6 steps, its pseudo-code is detailed in Algorithm 1. In step 1, for a requested SRP, the function generateTSIDs (SRP) generates a set of candidate tsids. Each candidate TSID has a size of at least 2 and not more than the MSD. A candidate TSID reduces the SRP stack size as follows: $length(SRP) - length(TSID) + 1$. In step 2, from the set of candidate TSIDs, function $generateSolutions(tsids)$ generates all the possible solutions to reduce the label stack of the SRP, a solution generates a label stack size less than the MSD. Additionally, a solution may be composed of one or multiple TSIDs depending of the MSD value and the longer of the SRP. The TSIDs that constitute a solution must not intersect. In step 3, a weight is assigned to each candidate solution, depending on the objective set by the operator. A solution's weight is equal to the number of new TSIDs that has to be created or number of new PCEP sessions it requires, hence preferring the re-utilization of already existing TSIDs or established PCEP sessions. In step 4, the solution with the lowest weight is chosen. In step 5, the best solution may require the creation and installation of new TSIDs. In this case, function $matchTSIDToPCEPNode$ identifies the node that has to install the new TSID, then the function $establishedPCEPSession(nodePCEP)$ checks if there is an active PCEP session with that node. If no session was found, the function $establishPCEPSession(nodePCEP)$ triggers the establishment of the PCEP session. This can be performed by a node configuration protocol such as NETCONF. The function $PCEPinstallTSID$ uses PCEP to install the TSID on the identified node. In step 6, in the initial SRP label stack, we replace the TSIDs with the labels reported by the PCC nodes for that TSIDs. Finally, the OTO algorithm returns the $labelstack$ to install.

The OTO algorithm can be implemented as a module of the encoding engine depicted in Fig. 1. The encoding engine triggers the installation of SRP and TSIDs, also maintains the TSID Database.

TABLE II: Entries for the linear programming models

| Topology | Nodes | Path Set | Possible TSIDs |
|---|---|---|---|
| Nobel-germany | 17 | 136 | 423 |
| Geant | 22 | 162 | 566 |
| Albilene | 12 | 41 | 109 |
| Brain | 161 | 2571 | 2073 |
| Germany50 | 50 | 991 | 3141 |

**Algorithm 1** Online TSIDs Optimization (OTO)

**INPUT:** *SRP* The SRP expressed as a list SIDs
**OUTPUT:** *labelStack* the SRP label stack MSD.

**STEP 1:** Generation of all the TSIDs for the SRP.

tsids = generateTSIDs(SRP)

**STEP 2:** Generation of possible solution.

*solutions = generateSolutions(tsids)*

**STEP 3:** compute the weight of each solution.

```
solWeight An array that holds the weight of each solution
for sol in solutions do
    weight = weightSolution(sol)
    push(solWeight,weight))
end for
```

**STEP 4:** Find the best solution.

bestSolution = minWeightSolution(solutions, solWeight)

**STEP 5:** Install required TSIDs.

```
for ts in bestSolution do
    if existTSID(tsid) then
        continue
    else
        nodePCEP = matchTSIDToPCEPNode(ts) node where to install the TSID
        if !establishedPCEPSession(nodePCEP) then
            establishPCEPSession(nodePCEP)
        end if
        installTSIDPCEP(ts,nodePCEP)
        addTSID(ts) Add the ts to TSID Database
    end if
end for
```

**STEP 6:** Compose the label stack.

```
labelStack = SRP
for tsid in bestSolution do
    replaceTSID(labelStack,tsid)
end for
Return labelStack
```

## V. Experimental Results

We performed several experiments to measure the performance of the two variations of the OTO algorithm. Mainly we compare the two variations of the OTO algorithm to the offline LP's models in terms of the number of installed TSIDs and required PCEP sessions. We also consider the case where the TSID mechanism is coupled with the Segment Routing Label Encoding algorithm (SR-LEA) presented in

[1]. The experiments use network topologies provided by SNDlib [7][8] and their demand matrices. We fixed the MSD to 5 labels, which is the value announced currently by the major equipment vendors. Table. II, details for each topology, the number of paths to encode and the number of possible TSIDs.

### A. *OTO for TSIDs minimization*

In the OTO algorithm for TSIDs minimization, the weight function attributes weights to all the possible solutions to reduce the size of the label stack. A solution that does not require new TSIDs has weight equal to zero whereas solutions that require the installation of new TSIDs are penalized by higher weights. The chosen solution is the one with the minimum weight. The performance of the OTO for TSIDs optimization is evaluated on the number of TSIDs created.

In order to evaluate the impact of the OTO $weight$ function, we consider an online worst-case scenario. Demands arrive sequentially, and for a given SRP there is no prioritization between solutions. The first found solution that reduces the SRP's label sack is chosen. As a result, new TSIDs are created more frequently. As seen in Fig. 2, for all the topologies, the OTO algorithm generates fewer TSIDs than to the worst-case scenario. We observe that the OTO gain against the worst-case scenario in terms of the number of TSIDs correlates with the number of possible TSIDs shown in Table II. The more TSIDs there are, the better the OTO performs. The weight function considers all the possible TSIDs combinations and favorites the reuse of TSIDs. In other words, the more paths OTO minimizes, the higher is the chance to reuse a TSID.

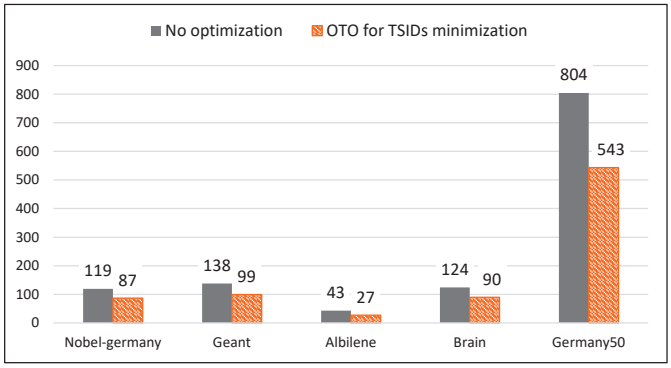



Fig. 2: OTO for TSIDs minimization compared to the worst-case scenario *i.e.*, online TSID installation with no optimization.

The LP model (1), computes the minimum number of TSIDs required for a given path set. It is the ultimate benchmark for the OTO algorithm. As it can be seen in Fig. 3, The OTO algorithm performs very well especially for small path sets. The number of TSIDs install by OTO algorithm is very close to the LP's solution for the first four topologies. However, we notice an increase in the gap between OTO and LP for topology *Germanny50*, this is due to the large TSIDs set.

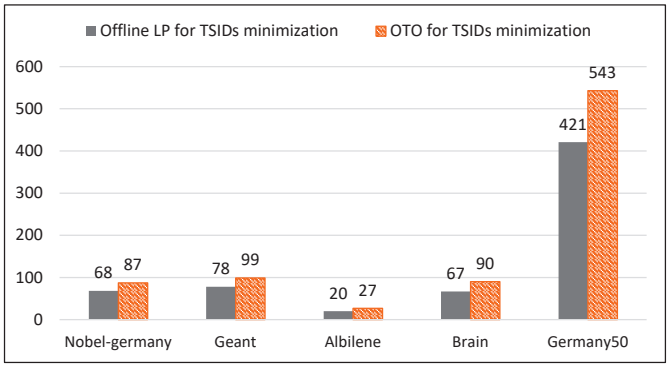



Fig. 3: OTO for TSIDs minimization compared to the Offline LP for TSIDs minimization

The path segmentation approach causes two problems 1) the creation of new states in the network (*i.e.*, TSIDs) and 2) the establishment of additional PCEP sessions. The OTO algorithm minimizes one of the two problems. We find it interesting to evaluate the impact of OTO optimizing one problem over another. Therefore, we evaluate the impact of the OTO algorithm when minimizing the number of PCEP sessions over the number of installed TSIDs. As seen in Fig. 4, optimizing the number of PCEP sessions increases considerably the number of installed TSIDs. Minimizing PCEP sessions leads to concentrating the TSIDs on certain nodes, which cause a weak TSIDs reuse factor. Additionally, this causes an important overhead to the control plane if the TSIDs are advertised via the IGP.

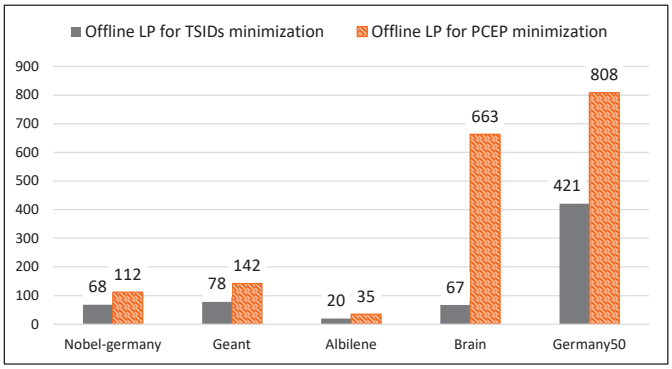
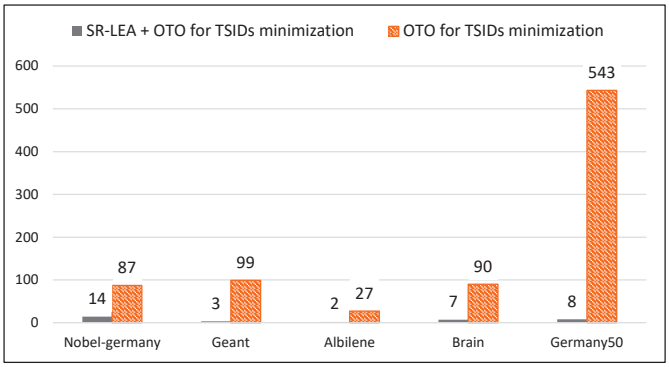



Fig. 4: Comparison of the number of TSIDs created by the two offline LPs

The PCE's encoding engine can use solely the OTO algorithm to reduce the size of all the label stacks. However, we noticed that when the OTO algorithm is coupled with SR-LEA encoding algorithm presented in [1]. The number of the installed TSIDs is drastically reduced as it can be seen in Fig. 5. In fact, the OTO algorithm is called only if the SR-LEA algorithm fails at computing a label stack with a size less than the MSD.

978-1-5386-0554-7/17/$31.00 ©2017 IEEE




3

Fig. 5: Comparison of the number of TSIDs created solely by OTO and SR-LEA encoding algorithm combined with OTO

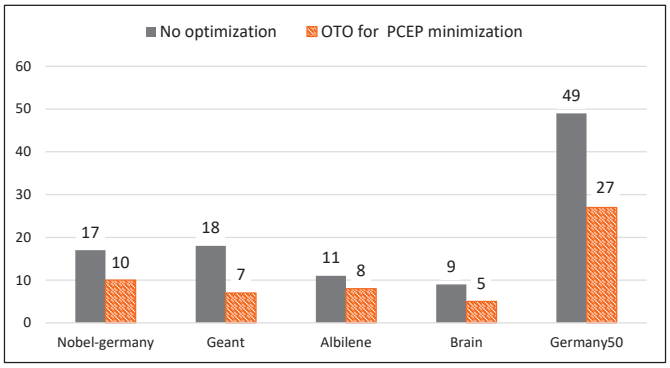



Fig. 6: OTO for PCEP sessions minimization compared to the worst-case scenario of an online TSIDs installation with no PCEP optimization.

### *B. OTO for PCEP sessions minimization*

Service Providers are moving toward the network softwarization era, where having a logically centralized controller is essential. Particularly, Traffic Engineering in Segment Routing network requires a centralized resource allocation and path computations. The traditional way SPs use the PCE with RSVP-TE is to establish PCEP sessions with only the network's border routers. However, the proposed path segmentation approach installs TSIDs on transit routers, which requires to maintain additional PCEP sessions with core nodes. Unfortunately, maintaining an active PCEP session with all the network nodes may rise scalability issues. Reducing the number of PCEP session with transit routers can be a priority for the SP especially for large networks.

In an online scenario, connection demands arrive sequentially. Therefore, anticipation the establishment of PCEP session with some set of the network nodes is not possible. In the path segmentation architecture, we propose that the PCEP sessions establishment be triggered only if required. For each SRP, a set of solutions composed of TSIDs to reduce stack size gets generated. The weight function penalizes solution that requires the establishment of new PCEP sessions, a solution that reuses already established PCEP session has a weight of 0. The solution with the minimum weight gets chosen. Several experiments have been conducted to evaluate the performance of OTO with PCEP sessions minimization as follows.

In a worst-case scenario, the first available solution is chosen. If the required TSIDs does not exist in the TSIDs Database and no PCEP session with the node that has to install the TSID is established then a PCEP session is initiated with the network node using network configuration protocols such AS NETCONF. As it can be seen in Fig. 6, the weight function of the OTO algorithm allows to reduce the number of PCEP sessions.

The offline LP model for PCEP session minimization (2), serve as a reference to evaluate the performance of the OTO algorithm. As it can be seen in Fig. 7, on all the tested topologies, the gap between the offline LP and OTO is very small. Hence, we conclude that OTO performs very well.

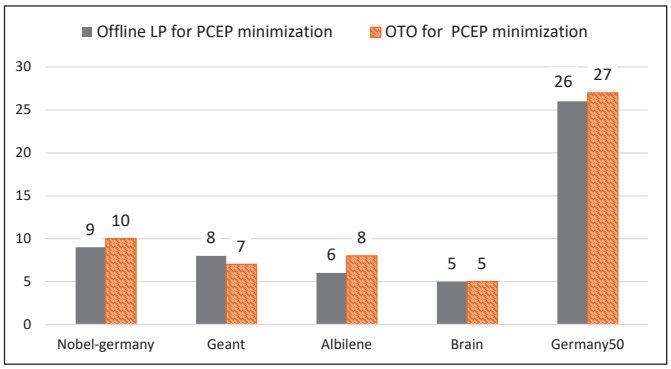



Fig. 7: OTO for PCEP sessions minimization compared to the offline LP for PCEP sessions minimization

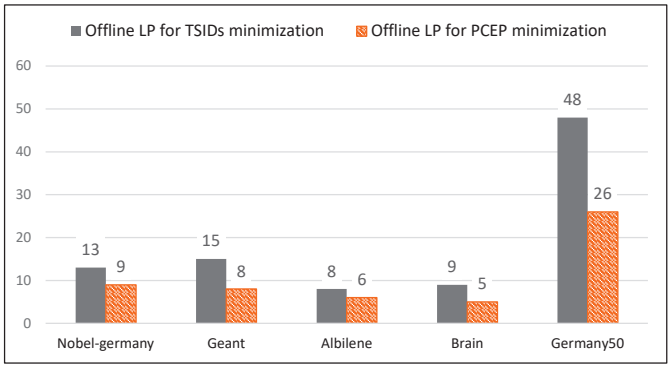



Fig. 8: Comparison of the number of PCEP sessions required by the two offline LPs

Minimizing the number of TSIDs comes at a price of augmenting the number of PCEP sessions. As it can be seen in Fig. 8 when comparing the results of the two LPs (1)(2) in terms of PCEP sessions, increasing the number of PCEP sessions augments the number of created TSIDs, for all the topologies minimizing the number TSIDs. Accordingly, minimizing the PCEP sessions concentrates the installation of TSIDs on certain network nodes, which in the case of a

node failure could impact more paths, especially when no fast recovery mechanism is defined.

## VI. CONCLUSION

In this work, we proposed a path segmentation approach to solve the Maximum Stack Depth (MSD) limitation in Segment Routing networks. We detailed its implementation and architectural requirements. We addressed the two optimization problems identified for this architecture. Namely, minimizing the number of created TSIDs and minimizing the number of PCEP sessions a PCE has to maintain with transit nodes. We proposed the Online TSID Minimization (OTO) algorithm, it addresses the two optimization problems. The defined weight function is adapted to each optimization problem *i.e.*, penalizes the creation of new TSIDs or the establishment of new PCEP sessions. The experimental results show that the two variations of the OTO algorithm perform very well, as their results are close to the reference offline LP models. Coupling the OTO algorithm with the Segment Routing Label Encoding Algorithm (SR-LEA) gave the best experimental results. Therefore, it is the recommended approach.

In future work, we will focus on solving challenges that face the proposed architecture, such as defining a fast-recovery mechanism for TSIDs similar to the classical MPLS fast-reroute. In addition, we will explore the possibility to extend the TSID mechanism to address the inter-AS segment routing problem.

## REFERENCES


[1] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier, "Label encoding algorithm for mpls segment routing," in *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*. IEEE, 2016, pp. 113–117.

[2] 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.

[3] 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.

[4] S. Sivabalan, J. Medved, C. Filsfils, V. Lopez, W. Henderickx, J. Hardwick, J. Tantsura, R. Raszuk, and E. Crabbe, "PCEP Extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-pce-segment-routing-08, Oct. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-segment-routing-08

[5] S. Sivabalan, E. Crabbe, I. Minei, and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model," Internet Engineering Task Force, Internet-Draft draft-ietf-pce-pce-initiated-lsp-07, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-pce-initiated-lsp-07

[6] P. Psenak, S. Previdi, C. Filsfils, J. Tantsura, W. Henderickx, H. Gredler, and R. Shakir, "OSPF Extensions for Segment Routing," Internet Engineering Task Force, Internet-Draft draft-ietf-ospf-segment-routing-extensions-10, Oct. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ospf-segment-routing-extensions-10

[7] 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

[8] ——, "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


978-1-5386-0554-7/17/$31.00 ©2017 IEEE