Enabling traffic engineering over segment routing

Rabah Guedrez

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Rabah GUEDREZ

Enabling Traffic Engineering Over Segment Routing

Rendre possible l'ingénierie de trafic dans les réseaux avec routage par segment

Thèse présentée et soutenue à Rennes, France, le 12/12/2018
Unité de recherche : Institut de recherche en informatique et systèmes aléatoires (IRISA)
Thèse N° : 2018IMTA0116

Rapporteurs avant soutenance :

Béatrice PAILLASSA Professeure au laboratoire IRIT, ENSEEIHT
Jean Louis ROUGIER Professeur à Telecom ParisTech

Composition du Jury :

Président :	Stefano SECCI	Professeur au CNAM
Examinateurs :	Béatrice PAILLASSA	Professeure au laboratoire IRIT, ENSEEIHT
	Jean Louis ROUGIER	Professeur à Télécom Guillaume
	URVOY-KELLER	Professeur à l'Université Nice Sophia Antipolis
Encadrants :	Samer LAHOUD	Professeur associé à ESIB, Université Saint-Joseph de Beyrouth
Dir. de thèse :	Olivier DUGEON	Ingénieur de recherche à Orange Labs
Co-dir. de thèse :	Géraldine TEXIER	Maître de conférences à IMT Atlantique

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

This thesis was conducted in collaboration with

Orange Labs and IMT Atlantique.

"My Lord! Increase me in knowledge"

This thesis is dedicated to
My beloved parents
My wife Sara and my son Adam
My country

Acknowledgements

My deepest and infinite gratitude goes to the Almighty. Without his guidance and support in times of doubt, I'm certain that I would never be able to achieve my goals.

I would like to express my sincere gratitude to my advisors: Dr. Olivier DUGEON, Prof. Géraldine Texier and Prof. Samer LAHOUD, you have been excellent mentors for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. I would also like to thank my jury members, Prof. Stefano SECCI, Prof. Béatrice PAILLASSA, Prof. Jean Louis ROUGIER and Prof. Guillaume URVOY-KELLER for accepting to review my work. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

My sincere thanks also go to my team at Orange Labs for the wonderful years that I passed among them. In particular, I am grateful to Jean-Marc COROLLEUR who provided me an opportunity to join his team.

My heartful thanks go to all my friends that were always there to support me: Younes, Yazid, Imed, and Raouf.

This moment represents the biggest achievement of my life yet. So, I would like to thank the people who shaped me to the person that I am now. My family. My deepest gratitude and love to my wife Sara, my sisters Wahiba, Soulef, Wissem and my brother Nacer Eddine, I would never be able to thank enough my beloved parents who sacrificed everything to help me get to where I am now. Thank you for giving me the courage and motivation to push forward. Thank you for making me look back and be proud of where i came from. You have always been my North Star when I feel lost.

Resumé

L'internet est devenu une partie intégrante dans la vie des personnes et des entreprises. Cette démocratisation de la connexion a permis une incroyable effervescence de nouveaux services et applications. Les entreprises en profitent pour s'étendre et accéder à de nouveaux marchés. Actuellement le trafic internet comporte des flux originaires d'un très grand nombre d'applications telles que les appels VOIP, flux vidéo, jeux interactifs, trading à haute fréquence, transactions bancaires, etc. chacune avec des contraints réseaux particulière en terme de délai, de perte ou de bande passante. Cela a donné naissance à deux types de réseaux le réseau Internet public et les réseaux étendus privées : (i) Internet, le réseau public mondial où IP est la technologie de réseau prédominante pour le transport de trafic. Bien qu'avec le temps la qualité s'est améliorée et le débit a beaucoup augmenté, les applications n'ont pas de garantie ni de contrôle sur la qualité de service (Quality of Service - Qos) qu'elles reçoivent. (ii) Les réseaux étendus ou WAN (Wide Area Network) permettent de relier des sites distants à travers des connexions point à point dédiées. Au cours des deux dernières décennies, la commutation multiprotocole par étiquette (MPLS) est devenue la technologie plus répandue pour les réseaux WAN d'entreprise. Grâce à sa capacité à prendre en charge les exigences de qualité de service et à attribuer différentes classes de service (CoS) selon les besoins des applications.

Les opérateurs tels qu'Orange font face à des défis économiques et technologiques considérables causés par la croissance exponentielle du trafic généré par leurs clients ainsi que par la baisse des prix causée par un marché de plus en plus compétitif. En effet, les clients opérant dans des secteurs sensibles tels que le multimédia, la finance ou la santé comptent sur la fiabilité des connexions WAN dédiées pour connecter leurs sites distants et fournir leurs services (par exemple la voix sur IP, la vidéo à la demande). Les applications des clients sont de plus en plus sensibles aux dégradations réseau qui peuvent se produire suite à des pannes, la congestion, ou des problèmes de routage. Ce qui se traduit par des contrats de niveau de service (Service Level Agreement - SLA) très strict. Afin de générer des profits, les opérateurs doivent revoir leur modèle économique et réduire les coûts d'investissements et d'opérations. Cela commence par une simplification du fonctionnement et de la gestion de leurs réseaux (complexes) en s'appuyant sur des plans de contrôle simples et agiles tels que le routage par segment et l'automatisation à l'aide de briques logicielles.

Pour ces raisons, le groupe de travail SPRING de l'Internet Engineering Task Force (IETF), l'organisme de standardisation des protocoles de l'Internet, a proposé

l'architecture de routage par segments (Segment Routing - SR). Son objectif principal est de développer une architecture avec un plan de contrôle simple, léger et facile à gérer. Cette architecture peut être instanciée sur deux plans de transfert existants : MPLS et IPv6. Le composant principal de cette architecture est le concept de routage par la source, dans lequel un paquet transporte (dans son en-tête) les indications du chemin qu'il doit suivre pour atteindre sa destination. Cette architecture a suscité beaucoup d'enthousiasme chez les opérateurs tels qu'Orange, ce qui se traduit par leur forte implication dans le processus de standardisation. Dans le cadre de cette thèse effectuée au sein d'Orange, nous nous sommes plus particulièrement intéressés à l'instanciation de l'architecture Segment Routing sur le plan de transfert MPLS (SR-MPLS).

L'un des principaux concepts de l'architecture Segment Routing est la notion de segments, ils représentent les différents composants du réseau : physiques (nœud, lien, etc.) ou logiques (service/application). Un identificateur appelé identificateur de segment ou SID est attribué à chaque segment. Le type et le format du SID dépendent du plan de données sous-jacent (une étiquette MPLS ou une adresse IPv6). L'empilement d'une liste de segments dans l'en-tête du paquet permet l'expression de chemins topologiques. Dans SR-MPLS, un chemin est encodé sous la forme d'une pile d'étiquettes MPLS, et qui par la suite est insérée dans l'en-tête du paquet. Un SID est alors représenté par une étiquette de 20 bits et est traité en utilisant les trois opérations habituelle de MPLS : POP pour empiler une nouvelle étiquette, PUSH pour dépiler l'étiquette courante et SWAP pour en changer la valeur.

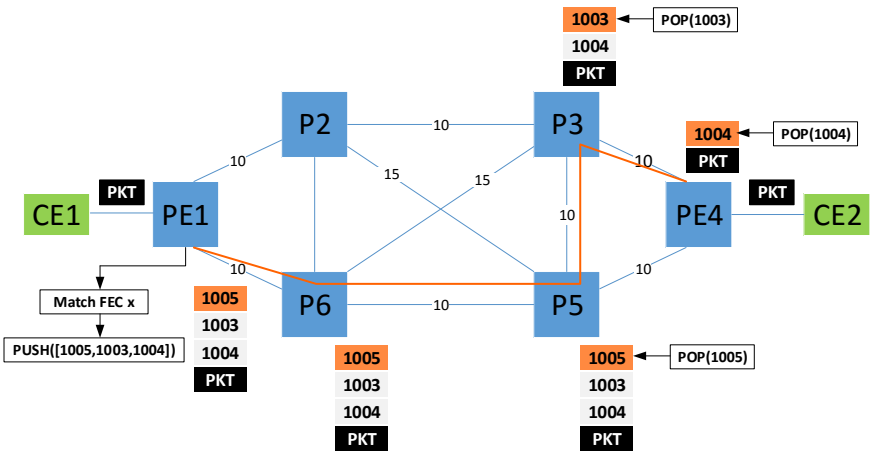


Figure 1: Transfert de paquets dans un réseau SR-MPLS

Un des avantages de l'instanciation de l'architecture Segment Routing sur le plan de transfert MPLS (SR-MPLS) est qu'elle nécessite moins de protocoles de contrôle que l'architecture MPLS classique. En effet, le développement incrémental de MPLS et l'apparition de nouvelles applications ont conduit aux ajouts successifs de fonctionnalités et de protocoles, ayant parfois les mêmes objectifs. Le résultat est une architecture complexe, non optimisée et très coûteuse en ressources. Un exemple concret de cette complexité peut être perçu lors de l'établissement d'un tunnel de réseau privé virtuel (VPN) pour relier un site client à un data center à travers le WAN avec une garantie de bande passante. Ceci nécessite la collaboration des protocoles de routage avec un protocole de signalisation et de réservation de ressource tel que Resource Reservation Protocol - Traffic Engineering (RSVP-TE). Dans l'architecture SR-MPLS, aucun pré-établissement de tunnels n'est nécessaire, car les paquets suivent le chemin qu'ils transportent dans leur entête. Ainsi, les états des tunnels ne sont maintenus qu'en bordure du réseau. Par conséquent, le nombre d'états maintenus dans le réseau est considérablement réduit. Le revers de la médaille réside dans le fait que l'élimination de la signalisation perturbe le processus de réservation de la bande passante, qui est un composant essentiel pour faire de l'ingénierie de trafic dans un réseau MPLS classique. Pour remédier à ça, une nouvelle approche a été adoptée pour permettre la mise en œuvre de l'ingénierie de trafic avec du Segment Routing. Cette nouvelle approche s'intègre bien dans l'évolution actuelle vers la "softwarization" des réseaux. Dans l'architecture de Software Defined Networking (SDN), un contrôleur logiciel pilote un plan de contrôle SR-MPLS et assure en même temps la gestion ainsi que la réservation des ressources réseau, alors que le plan de contrôle assure le transfert des paquets tout en respectant les chemins calculés par le contrôleur logiciel.

Dans cette thèse, nous avons identifié et proposé des solutions aux problèmes pour faire de l'ingénierie de trafic dans les réseaux SR-MPLS (SR-TE). Ce travail est divisé en deux parties principales : (i) l'identification des défis techniques et la résolution du problème lié au cas d'utilisation de l'ingénierie de trafic. (ii) définition des exigences architecturales et construction d'une preuve de concept fonctionnelle.

0.1 Algorithme native d'encodage de chemins SR

Un chemin SR est encodé comme une pile d'étiquettes que le routeur d'entrée inséré (PUSH) sur l'en-tête du paquet, comme le montre la figure 1. 1. En fait, insérer plus d'une étiquette a été supporté depuis la première version de MPLS [1] pour des

cas d'utilisation comme : hiérarchisation [2], réseau privé virtuel de couche 3 [3], etc. Comme on peut remarquer, ces cas d'utilisation nécessitent une pile d'étiquettes relativement petite (deux a trois étiquettes.). Par exemple, un scénario de L3VPN ne nécessite que deux étiquettes simultanément : l'étiquette du tunnel et celle du VPN. Cependant, un chemin SR, en fonction de la taille du réseau, peut-être composé de plusieurs dizaines segment (SID) ce qui produit une grande pile d'étiquettes. Par conséquent, les routeurs doivent pouvoir insérer un plus grand nombre d'étiquettes afin de profiter pleinement du potentiel du SR.

Topologies	Nombre de Noeuds	Nombr de liens	Nombre de demades
Geant	22	36	431
Albilene	12	18	131
Brain	161	166	9045
Germany50	50	80	1270
Nobel-germany	17	26	248

Table 1: caractéristiques des topologies utilisées dans les simulations.

L'encodage des chemins Segment Routing sous la forme d'une pile d'étiquettes MPLS engendre une surcharge que les équipements actuels ne peuvent pas supporter. En effet, les équipementiers n'ont pas anticipé l'insertion d'un grand nombre d'étiquettes MPLS dans les en-têtes des paquets. Chaque routeur est conçu pour supporter une profondeur maximale de pile d'étiquettes MPLS appelé MSD (Maximum Stack Depth). Le MSD est une limitation matérielle des équipements, qui représente le nombre maximum d'étiquettes qu'un nœud d'entrée peut insérer sur l'en-tête d'un paquet [4]. Actuellement, le MSD varie de 3 à 5 selon l'équipementier. Le MSD a un impact fort sur la capacité d'un réseau SR-MPLS d'exprimer ses chemins. Un chemin est inutilisable en Segment Routing s'il ne peut pas être exprimé par un nombre de label suffisamment petit pour que la profondeur de pile d'étiquettes soit inférieure ou égale au MSD du routeur chargé d'insérer la pile d'étiquette sur l'en-tête du paquet. Une valeur de MSD petite (comme celle des routeurs actuels) ne permet d'employer qu'un sous-ensemble des chemins du réseau SR-MPLS, ce qui concentre le trafic et peut engendrer des pertes et entraîner de la congestion.

Les équipementiers utilisent des circuits intégrés spécifiques aux applications (ASIC) pour accélérer les fonctions de traitement et de transfert des paquets afin d'atteindre des vitesses de transfert importantes. Dans les équipements assurant le transfert des paquets MPLS, la limitation MSD provient de l'implémentation de l'opération PUSH dans les ASICs [5]. Cette limitation doit être prise en compte dans le processus de

calcul du chemin, car elle peut rendre inutilisables les longs chemins qui sont exprimés avec une pile d'étiquettes supérieure à la MSD. Par conséquent, les flux de trafic sont forcés sur des chemins relativement courts, ce qui entraîne une répartition inefficace du trafic ou une congestion aggravée du réseau. D'où la nécessité de développer des algorithmes efficaces pour l'encodage des chemins SR.

Nous avons identifié le problème d'encodage des chemins Segment Routing comme un verrou technologique qui doit être surmonté pour faire de l'ingénierie de trafic dans des réseaux Segment Routing. Nous avons développé des algorithmes qui permettent de calculer une pile d'étiquettes minimale pour l'expression de chemins SR. Les algorithmes proposés fonctionnent nativement dans l'architecture de routage par segment. Ils calculent le nombre minimum d'étiquettes pour exprimer un chemin SR.

Pour évaluer la performance de nos solutions, nous avons en premier lieu opté pour des simulations proches de scénario de production. Pour cela, nous avons sélectionné un ensemble de topologies réelles [6] avec leurs matrices de trafic, comme on peut voir dans le tableau 1. Nous avons commencé par formuler le problème de répartition de la matrice des demandes sur la topologie sous-jacente sous la forme d'un programme linéaire. Par la suite, nous avons résolu ce programme linéaire qui a permis de calculer le chemin optimal pour chaque demande. L'ensemble des chemins généré par le programme linéaire est ensuite utilisé pour évaluer la performance des deux algorithmes d'encodage. Les résultats de simulation montrent une augmentation considérable du nombre de chemins qui sont exprimés avec une profondeur de pile inférieure au MSD. Par la suite, nous avons démontré l'efficacité de nos algorithmes avec une implémentation dans le contrôleur SDN OpenDayLight pilotant une topologie composée d'une douzaine de routeurs de production et notre implémentation Open Source d'un routeur SR-MPLS.

Ces algorithmes permettent de réduire la taille de la pile d'étiquettes nécessaire pour encoder un chemin SR. Bien que les résultats soient satisfaisants, ils ne permettent pas d'exprimer la totalité des chemins du réseau, le MSD restant trop limitant. Pour cela, nous avons proposé une méthode de fragmentation de chemins afin de résoudre la limitation MSD (le but est de pouvoir se rapprocher de 100% des chemins exprimés). Nous avons ainsi proposé un nouveau type de segment cible appelé Target SID (TSID) . Ce type de segment permet de créer des raccourcis dans le réseau physique pour outrepasser MSD.

0.2 Nouveau type de Segment: Target-SID

Les algorithmes d'encodage natifs que nous avons proposés permettent de réduire considérablement l'impact du MSD. Cependant, l'impact de cette limitation persiste pour de très longs chemins, qui ne peuvent être exprimés avec une profondeur de pile inférieure au MSD même avec l'optimisant l'encodage. Ces chemins peuvent être des chemins de supervision ou des chemins dans des topologies de grand diamètre. Nous avons donc proposé une nouvelle méthode d'encodage des chemins Segment Routing, cette méthode de base sur un nouveau type de segment appelé le Targeted SID (TSID). L'idée est de remplacer plusieurs étiquettes dans la pile initiale par une seule étiquette TSID. C'est dans ce but qu'un chemin est fragmenté en plusieurs sous-chemins. Ces sous-chemins pourront potentiellement être remplacés par des TSIDs pendant la phase d'encodage. De plus, les TSID peuvent être partagés entre plusieurs chemins pour réduire le nombre de TSID créés dans le réseau. L'utilisation des TSID permet de réduire la taille de la pile d'étiquettes pour exprimer un chemin Segment Routing dans la limite du MSD et est encore plus efficace lorsqu'elle est asssociée aux algorithmes d'encodage proposés précédemment. Cependant, les TSID doivent être installés au préalable dans le réseau avant que le trafic ne soit acheminé sur le chemin SR. Quand un paquet atteint un nœud spécifique sur son chemin, l'étiquette TSID en haut de la pile sera remplacée par la séquence d'étiquettes correspondant à la portion du chemin qu'il a remplacé . Nous avons validé l'efficacité de cette méthode sur les topologies présentées dans le tableau 1. L'encodage avec des TSID a permis de limiter l'impact du MSD y compris sur les chemins longs.

Certes, la méthode du fractionnement des chemins grâce aux TSIDs permet de résoudre le problème du MSD. Cependant, elle nécessite de créer et de maintenir de nouveaux états dans les réseaux. Un grand nombre de chemins Segment Routing peut générer un nombre très important de TSIDs. Pour remédier à ce problème, nous avons développé un algorithme en ligne d'optimisation de nombre TSID. Dans cet algorithme, nous encourageons par le biais d'une fonction d'incitation à réutiliser les TSID existants pour satisfaire chaque nouvelle demande d'encodage. Pour évaluer les performances de cette solution, nous avons formulé le problème d'installation d'un nombre minimale de TSID sous la forme d'un programme linéaire. Les résultats obtenus par l'algorithme en ligne sur les majorités des topologies sont proches des résultats optimaux obtenus par le programme linéaire.

0.3 Conclusion et Perspectives

Notre travail s'inscrit dans le contexte de l'instanciation MPLS de l'architecture de Segment Routing. Cette thèse a proposé des solutions pour faire de l'ingénierie de trafic dans un réseau Segment Routing, et ainsi permettre aux opérateurs de profiter de l'architecture de SR. Nos contributions ont apporté des solutions algorithmiques pour résoudre le problème de la limitation MSD des équipements réseau. Il est essentiel de surmonter cette limitation pour rendre possible une large adoption de Segment Routing par les opérateurs. Afin de démontrer l'efficacité des algorithmes proposés, nous les avons mis en œuvre dans une architecture SDN pour faire l'ingénierie de trafic dans un réseau SR-MPLS. Nos algorithmes d'encodage, ont été implémentés dans le contrôleur SDN OpenDayLight.

Si nos solutions permettent d'utiliser Segment Routing dans les réseaux actuels, elles offrent également des perspectives intéressantes pour des cas d'utilisation future tels que le chaînage de fonctions réseau virtualités ou la protection contre les pannes.

Nous travaillons actuellement sur l'adaptation de nos algorithmes d'encodages pour les appliquer aux chemins de protection calculés sur le graphe de topologie post-convergence. Les algorithmes doivent éviter le problème de boucle qui peut survenir à cause d'un mauvais encodage de la pile d'étiquette ou d'une erreur de calcul sur le chemin poste convergence. Ce travail est guidé par le standard TI-LFA (Topology-indépendant Loop Free Alternate) [çois-spring-segment-routing-ti-lfa-02].

Dans un second temps, nous envisageons l'utilisation de Segment Routing dans le cadre de la virtualisation des fonctions réseau afin de permettre l'encodage des chemins de chaîne de service (NFV Service Chaining). En effet, il est nécessaire de pouvoir imposer à un flux de trafic de passer une suite ordonnée de plusieurs fonctions réseau virtualisées implantées dans des nœuds du réseau. Dans SR-MPLS, chaque fonction réseau est associée à une étiquette MPLS ce qui augmente la taille de la pile pour exprimer le chemin de service. De plus, une fonction réseau peut modifier la classe de service du flux qui la traverse. Ces caractéristiques des chemins de chaîne de service doivent donc être prises en compte dans la conception d'un nouvel algorithme d'encodage.

Contents

List of Figures

Chapter 1

Introduction

Service Providers are facing extreme challenges to keep pace with the exponential growth of their client's traffic. Moreover, clients are requesting more strict Quality of Service (QoS) requirements for their sensitive applications such as medical, financial, real-time videos calls and streaming application resulting in tightened Service Level Agreements (SLA). Consequently, service providers need to meet those requirements while reducing cost. One of the solutions they are adopting is simplifying their complex networks and rely more on software to reduce the operational expense and capital expenditure.

Organizations operating in highly demanding environments such as banking, trading, medical, etc. rely on dedicated Wide Area Network (WAN) connections to connect their remote sites. Mainly because of the guaranteed QoS requirements such as bandwidth, protection, delay, etc. that the service providers deliver. For that, service providers invest in building and maintaining specialized WANs to satisfy the increasing demands. The majority of these networks run on the Multiprotocol Label Switching (MPLS). However, over the years and with the increase of use cases the MPLS control plane grow increasingly complex, which required a variety of interconnected protocols built by different standardization working groups, thus making it hard to manage, troubleshoot and evolve.

For the reasons mentioned above, the Internet Engineering Task Force (IETF) SPRING working group has proposed the Segment Routing (SR) architecture. Its main objective is to have a simple and easy to manage control plane. It relies on an old networking paradigm known as source routing, where a packet carries in its header the path to reach its destination. This architecture has generated a lot of enthusiasm among service providers such as Orange, due to the simplification that it brings to their IP/MPLS networks. In fact, the instantiation of SR architecture over the MPLS data plane requires less control plane protocols: There is is no need

to pre-establish tunnels and the per-flow states are maintained only at the edges of the network. Therefore, no signaling protocols such as LDP and/or RSVP-TE are required. Consequently, the number of states maintained in the network is considerably reduced. However, the elimination of signaling disrupts the bandwidth reservation process which is an important tool for traffic engineering. The lack of such an important use case makes the service providers hesitant to migrate their networks to SR. In this thesis, we identify and solve the problems that must be overcome to do traffic engineering over SR (SR-TE) in IP/MPLS networks. Our work is divided into two main parts: Identify and address the technical challenges to deliver such use case, define architectural requirements, and build a working proof of concept.

In Chapter 2, we introduce the SR architecture and detail its building blocks. We detail the implementation specifics SR-MPLS. We finish by detailing few use cases where SR can have an important impact.

In Chapter 3, we first focus on defining a reference architecture to achieve traffic engineering over SR. Interestingly, Software Defined Networks (SDN) has the capacity to fill the gap of resource reservation and management in the SR architecture. In fact, SR does not rely on signaling. Therefore, the resources availability does not get updated throughout the network, which may lead to overbooking and poor network utilization. Consequently, resource reservation and management mechanisms functionalities are developed into an SDN controller who will maintain a centralized Traffic Engineering Database (TED). Additional functionalities can leverage the TED such as path computation and network optimization, etc. we build the proposed architecture using the OpenDayLight SDN Controller coupled with a testbed composed of commercial routers running SR control plane. Secondly, we use the proposed architecture to address the Maximum Segment Identifier Depth (MSD) hardware limitation. Which limits the number of labels a router can push onto a packet's header. This prevents the expression of a considerable number of network paths causing a sub-optimal network resources utilization. The MSD may slow down the adoption of SR. To solve this problem, we propose two label encoding algorithms that slacken the impact of the MSD by reducing the size of the label stack to express segment routing paths. Both algorithms work natively without any additions to the SR architecture. We prove the effectiveness of both algorithms through simulation over real network topologies. In addition, we develop into OpenDaylight a label encoding engine that leverages both algorithms.

In Chapter 4, we propose a new method to optimize the encoding of SR paths and therefore reduce the impact of the MSD limitation. In this approach, we define

a new segment type called Targeted Segment Identifier (TSID). It is used in the encoding of SR paths, where a single TSID is used to substitute multiple labels in SR path description. We propose an SDN based architecture to implement the TSID mechanism. Additionally, we propose and compare several optimization algorithms to reduce the overhead introduced by TSID architecture.

Finally, in Chapter 5 we present the general conclusions of this thesis. We emphasize the major contributions of this work, point out the open issues and the interesting future directions.

Chapter 2

Segment Routing

2.1 Introduction

In September 2013, the Internet Engineering Task Force (IETF) started a new Working Group (WG) called "Source Packet Routing in Networking" (SPRING) to standardize the Segment Routing (SR) architecture. One the main goal of this new architecture is to address the complexity of Multi-Protocol Label Switching (MPLS) networks that are currently used by Service Providers to transport data in their core networks.

Over the years, MPLS has gained acceptance as the de facto technology to deploy large IP networks. Thanks to the efforts made by the IETF standardization working groups, MPLS control plane has continuously evolved in order to improve the performance, scalability and provide support for new services. The introduced enhancements required the definition of new protocols *e.g.*, Resource Reservation Protocol Traffic Engineering RSVP-TE, the Label Distribution Protocol (LDP) and the extension of others : Open Shortest Path First (OSPF) [7], Intermediate System to Intermediate System (IS-IS) [8] and the Border Gateway Protocol Link State [9].

Traffic engineering represents an essential task for service providers. It optimizes the use of the network resources (*e.g.*, send traffic through less congested links), makes sure that client Quality of Service (QoS) requirements are met and enforces network resiliency by bypassing link and node failures through fast reroute. An MPLS network with traffic engineering capabilities is referred to as MPLS-TE.

The MPLS-TE control plane consists mainly of the two protocols: RSVP-TE to establish and maintain Label Switched Paths (LSPs), and an IGP protocol to advertise the network resources. Large networks with a large number of MPLS-TE LSPs suffer from scalability issues [10] due to RSVP-TE, which consumes a significant amount of node resources: memory storage to maintain millions of states, processor

cycles to process these huge state tables and to synchronize the control protocols. Indeed, after a node restart, the number of messages exchanged over the links to repopulate the state tables can cause node congestion, which involves packets being dropped and/or retransmitted and increases network convergence time.

As an example, let us consider a network of a WAN network composed of 1,000 edge routers and 60 core routers. In order to establish a full-mesh of MPLS-TE LSPs between the edge routers, each edge router has to establish 999 LSPs. Consequently, the core routers have to manage and maintain a total of 999,000 LSPs, this number may be multiplied in the case of establishing dedicated TE LSPs for each service or QoS class, this phenomenon is known as the N-Squared problem [11]. Therefore, the network performance is severely impacted by the RSVP-TE refresh mechanism, used to maintain up all the instantiated LSPs in the network. Some techniques such as grouping the refresh messages of multiple LSPs in one or increasing the refresh period are used to overcome the RSVP scalability issues [12]. However, these are fixes that do not address the core problem. Also, using a hierarchy of LSPs reduces the number of LSPs to maintain in the core routers. Or, RSVP-TE may be used to only deliver node and link protection [13] while LDP is used to deliver VPN services without bandwidth reservation.

Finally, the complexity of the management and the configuration of traffic engineering information on the routers prevents their efficient exploitation. Service Providers have become aware of the problems with MPLS and reached a point where a global simplification of its control plane is necessary.

SR was introduced in order to overcome these issues and to simplify the use of MPLS. SR leverages the source routing paradigm, where the path taken by the packet to reach its destination is encoded in its header. The most important building block of this architecture is the concept of segments [14]. Basically, a segment can represent a topological path (*e.g.*, the shortest path to a node) or service (application). Combining two or more segments allows the expression of any network path. Instead of using a dedicated protocol, segments are advertised using routing protocols. SR architecture [15] [16] can be instantiated over two data planes: MPLS, without any data plane modification and an IPv6-based solution that requires a hardware upgrade. SR over MPLS data plane (SR-MPLS) simplifies the deployment and configuration of services such as fast reroute (link and node protection) and VPNs, by eliminating the need for protocols like LDP and/or RSVP-TE. Equipment vendors have already started delivering the support for SR in their recent software releases. The first interoperability tests have been conducted internally by service providers

or by independent entities such as the European Advanced Networking Test Center (ENTC) [17].

In this chapter, we take a deep dive into the SR architecture. First, We start by introducing source routing in section 2.2. Second, we detail the different building block of SR in section 2.3. Third, we detail the MPLS in in section 2.4 implementation. Finally, we detail several SR use cases in section 2.5.

2.2 Source Routing

The source routing paradigm was first introduced in the early versions of the Internet Protocol IPv4 [18]. Since then various network architectures and protocols have adopted it, such as MPLS [19], IPv6 [20] and wireless networks protocols: Dynamic Source Routing DSR [21] and Source Demand Routing Protocol (SDRP) [22]. Additionally, source routing is a main component of the SR architecture. Understanding source routing is essential for a better grasp of SR. For that purpose, we detail in this section the functioning of source routing and its implementation in different protocols.

Source routing enables the source node (ingress) to specify explicitly the path that packets have to follow in order to reach their destination. The Source-Routed Path (SRP) consists of a sequence of nodes and links the packets pass through. The components of the SRP (nodes, links) are carried in every packet header. The SRP can express any topological paths. Consequently, it allows forwarding of traffic on paths that are not the IGP shortest paths. Such characteristic is interesting for multiple use cases such as traffic engineering, path monitoring and troubleshooting. We distinguish two types of Source Routing:

1. Strict Source Routing: in strict source routing, all the intermediate hops (nodes, links, etc.) between the source and the destination are listed in the packet header. In this case, the packet must pass through exclusively the listed hops. Two successive hops in the packet header are adjacent. Intermediates nodes do not have to determine the next hop because the forwarding decision is only based on the information carried in the packet header. In Fig. 2.1, service provider connects two Customer Edge (CE) routers $CE1$ and $CE2$, and decides that the traffic sent from $CE1$ to $CE2$ follows the path A: $PE1 \rightarrow P2 \rightarrow P3 \rightarrow PE4$. Hence, $PE1$ receives packets from $CE1$ then encodes into each packet header the list of all the intermediate hops identifiers. When the packet reaches a given node, this node looks up the reference of the next hop in the list in order to

determine to which hop it must forward the packet. Finally, $PE4$ removes the hop list before forwarding the packet to $CE2$.

2. Loose Source Routing: in loose source routing, the packet carries only a subset of the hop identifiers that constitute the full path. The packet goes through all the hops listed in the header but not only *i.e.*, packets may go through hops that are not present in their header. This happens when two successive hops in the packet header are physically separated by one or more intermediate nodes. Going back to our example shown in Fig. 2.1, $CE1$ sends its traffic to CE2 following path B: $PE1 \rightarrow P3 \rightarrow PE4$, only $P3$ is specified as an intermediate node. In this case, $PE1$ determines that $P6$ is the next hop to reach $P3$ by looking up its forwarding table. This is considered as a *loose path* because $PE1$ and $P3$ are not direct neighbors, consequently packets have to go through $P6$, which is not present in the hop list of the source path.

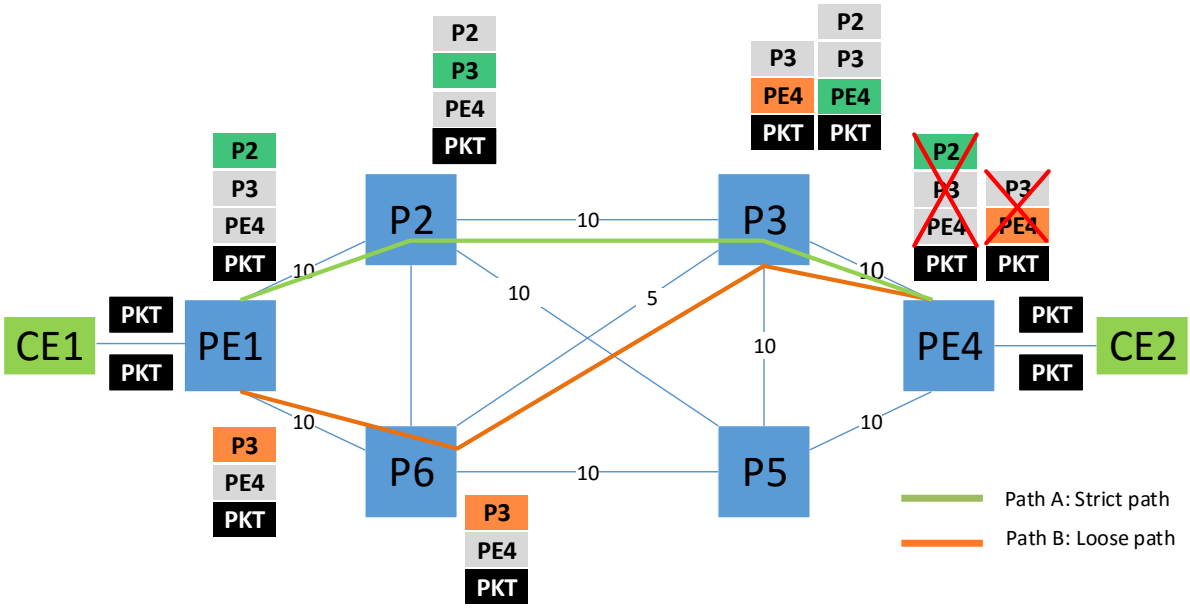


Figure 2.1: Path A is a strict path where all the intermediate nodes are carried in the packet header. Path B is a loose path because only a subset of the intermediate nodes is carried in the packet header

2.2.1 Source Routing with IPv4

As mentioned above, source Routing was introduced in the early versions of the IPv4 standards [18], it was implemented as an option in the IPv4 header, for which two types were defined: type 131 for Loose Source and Record Route (LSRR) and type 137 for Strict Source and Record Route (SSRR).

IPv4 standard limits the length of the SRP to a maximum of 9 IPv4 addresses because it is transported as an option in the IPv4 packet header, which has a maximum size of 40 Bytes. Each node crossed by the packet gets recorded, which allows the destination to build its own loose or strict route in order to respond via the same path.

IPv4 source routing was originally used by network administrators to perform tasks such as network discovery, measurements and debugging. Due to the discovered vulnerabilities and its exploitation for malicious purposes [23], such as conducting Denial of Service attacks, spoofing attacks, firewall bypassing and other attacks. Thus, the majority of network administrators have disabled the support of these options in their networks. As a result, the IETF advice now that packets with LSRR and SSRR options should be dropped [24].

2.2.2 Source Routing with IPv6

The source routing behavior from IPv4 was reproduced in IPv6 extension header Type 0 Routing header (RH0), copying almost the same concepts as in IPv4. Contrarily to IPv4, the RH0 is an extension header and its size is limited only by the maximum transmission unit. For this reason, an RH0 can carry longer paths in comparison to IPv4. For example, with a maximum transmission unit of 1500 Bytes, an SRP can be composed of 90 IPv6 addresses, with the possibility to include the same address multiple times. However, big SRPs have made things worse from a security point of view. For example, a Denial Of Service attack can be carried out via traffic amplification on a specific path between two nodes, this attack is more powerful with RH0 than with IPv4 source routing, due to the fact that the number of IP addresses carried in the RH0 is much greater than IPv4 source routing option: the RH0 can carry up to 90 addresses compared to only 9 for IPv4, this gives the attacker the possibility of oscillating up to 44 times between two nodes, which would eventually cause a congestion on that path. As a result, the IETF deprecated the support of RH0 [25].

The deprecation of RH0 was intended to stop its exploitation for malicious purposes, but not to prevent totally the use of source routing in IPv6 networks. Indeed, new secure routing headers were defined to deliver source routing for different networks types. For example, an IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks [26] and the Segment Routing extension header [27] (see Section A).

2.2.3 Source Routing with MPLS

MPLS networks also took advantage of source routing. Using the PUSH operation multiple labels get added to the packet's header, this is known as label stacking. Each label from the stack identifies a unique LSP. These LSPs are pre-established either by RSVP-TE or by LDP or BGP. Each intermediate node installs a state in its forwarding table for each LSP that goes through it.

Unlike IPv4/IPv6 implementation of source routing where an IP address identifies a unique node. In MPLS, a label is local to the node and it determines the path to a hop (either direct or over an LSP). Additionally, MPLS implementation does not suffer from the same security issues as in IPv4/IPv6, because a service provider drop traffic originated from untrusted sources.

In the next section, we go into more detail on how SR architecture leverages the source routing paradigm and the implementation specifics for each data plane: MPLS and IPv6.

2.3 Segment Routing Generic Concepts and Terminology

As mentioned in the previous section, source routing is used since the early days of internet protocols. Although, it constitutes a core routing mechanism in environments such as ad-hoc wireless networks [21, 22]. It was always implemented as extensions for the IP protocols to solve specific use cases. The classical shortest path stayed the de facto routing forwarding mechanism. The lightweight and the flexibility of source routing in the expression of topological paths led the IETF to create the Source Packet Routing in Networking (SPRING) working group, that works to define and standardize the segment routing architecture that leverages source routing. In this section, we define and detail the main concepts of SR.

In fact, SR is not a new protocol, but rather an architecture that defines a set of requirements to implement source routing over IP networks. The SPRING working

group focuses on the definition and the standardization of SR architecture and its use cases. Additionally, other IETF working groups work on the definition of protocol extensions: Open Shortest Path First (OSPF), IS-IS for IP Internets (ISIS), Inter-Domain Routing (IDR) for BGP, Path Computation Element (PCE), and IPv6 Maintenance (6man).

Service provider core networks are experiencing limitations due to the growing number of services and protocols deployed, which creates a barrier towards achieving its optimal exploitation [10]. Therefore, in order to support the ever-growing number of protocols, services and policies, network nodes are forced to maintain and manage large state tables, which consumes a large amount of the node's memory and processing resources. It also becomes very difficult for network administrators to manage and troubleshoot. All these points were taken into consideration by the SPRING working group when defining the SR architecture. The SR architecture is focused on the simplification of the node's control plane and the reduction of states maintained in forwarding tables. For instance, in an MPLS network with SR deployed there is no need to pre-establish tunnels using RSVP-TE because the instructions to forward the packets are carried in their headers. This leads to a considerable reduction in the number of states maintained by the intermediate nodes. Finally, the SR architecture does not require additional protocols: it extends those already deployed, such as the Internal Gateway Protocols (IGP) *e.g.*, OSPF [7] and ISIS [8] have been extended to exchange SR information. SR main architectural concepts can be summarized in the following points:

- Avoid as much as possible deploying new protocols dedicated for SR-MPLS,
- Extend already deployed protocols used by the IP/MPLS networks (OSPF, ISIS, BGP-LS),
- Maintain per-flow states only at the network boundaries and reduce the flow states maintained by the intermediate nodes.

SR is a generic architecture that can be instantiated over existing data planes; this reduces its deployment cost and accelerates its mastering by the service provider's engineers. Depending on the existing hardware, SR deployment requirement varies from a software upgrade to new hardware. Sections 2.4 and appendix A we explain in more detail the deployment challenges of SR-MPLS and SR-IPv6 respectively. In addition, SR can be deployed gradually because the control plane protocol extensions are designed to be backward compatible. This enables SPRING nodes to interoperate with other nodes that are not SR enabled.

2.3.1 Segments and Segment Identifiers

One of the main concepts in SR architecture is the notion of segments. They represent different network components: physical (node, link, etc.) or logical (service/application). An identifier called *Segment Identifier* or SID is attributed to each segment. The SID type and format depend on the underlying data plane (an MPLS label or an IPv6 address as explained in Section Sections 2.4 and A). An SID can have a global significance in the domain or local to the node advertising it. Stacking a list of SIDs into the packet header allows the expression of (strict or loose) topological paths. For the rest of the manuscript the terms *segment* and *SID* are interchangeable.

In SR, a source route is encoded by the ingress node as a sequence of segments in the packet header. Each individual segment has a corresponding data plane forwarding instruction. The interpretation of the segment list creates an end-to-end path. Intermediate nodes may modify the path by adding or removing one or more segments from the packet header. A segment can represent a node, a link, a Border Gateway Protocol (BGP) peering adjacency, an LSP, or even a service.

SR flexibility in expressing topological or service-based paths makes it very attractive for data centers or service provider networks. For example, a segment can be interpreted by an intermediate node as *send packet to node N via the IGP shortest path*, *send this packet to a Deep Packet Inspection (DPI) virtual machine* or *forward packet through this node's interface X*. Moreover, the possibility of deploying this new architecture over existing data planes with no or minor changes favors its wide adoption. Currently, two data planes are considered for its instantiation [10]: an MPLS data plane without any changes and IPv6 with a new type of Routing Extension Header.

2.3.1.1 Segment Routing Global Block SRGB

Service providers may consider deploying SR in a gradual manner. For example, one may choose to deploy SR over only a subset of the network nodes (*e.g.*, PEs), or use it to enable specific use cases such as Fast Reroute (FRR). Consequently, SR control plane has to co-exist and interoperate with other control plane protocols on the same node or over the network. For that, mechanisms are needed in order to avoid conflicting or duplicated entries in the forwarding plane. Therefore, each SPRING node reserves an SID block only for SR use: in the case of SR-MPLS a block of labels or a block of IPv6 addresses for SR-IPv6. This block is named the *Segment Routing*

Global Block (SRGB). Reserving the same SRGB throughout the network is highly recommended because it simplifies operations and troubleshooting. SR information such as the SRGBs and SIDs are advertised inside a domain using the IGP protocol.

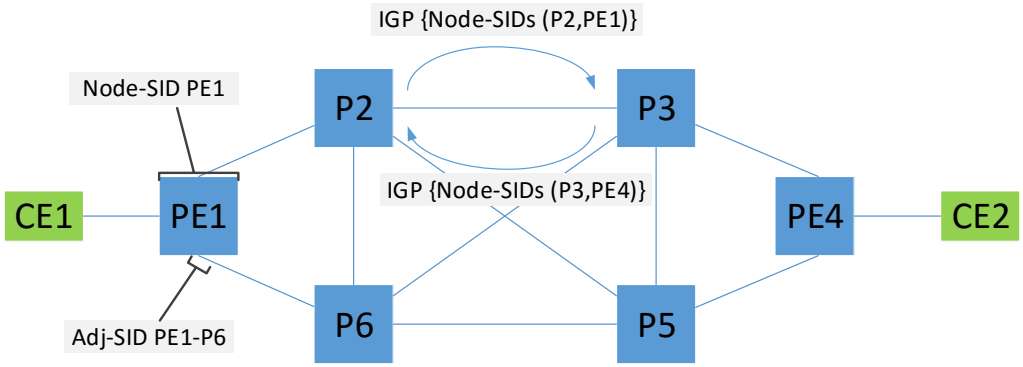


Figure 2.2: Reference network topology

2.3.1.2 Segments Global And Local Scope

The scope of an SID may be domain wide or local to a particular node. An SID is global (*i.e.*, unique in a SPRING domain) if it takes its value within the SRGB. In addition, a global SID has an associated entry in the forwarding tables of all the SPRING nodes, this is detailed in Section 2.4.2. The SID can also be local to the node advertising it. Consequently, the same SID may be reused and therefore advertised by other nodes. A local SID takes its value outside the SRGB. The local SID is readvertised by all the SPRING nodes but only its owner installs a forwarding entry associated with it. The entry in the forwarding table refers to the next hop, which has been determined based on the IP forwarding table.

2.3.1.3 Active Segment

An active SID is the one that the current node uses to make a forwarding decision. It can be the top label in an MPLS stack or the IPv6 address in the Destination Address (DA) of the IP header. Every SID the encode the SR path becomes at least once active before the packet reaches its destination. A global SID may stay active and span several nodes. For example, the routers IPv6 loopback address in SR-MPLS. A local SID must become active only on the node that advertises it. For example, the SID attached to a specific router interface in SR-MPLS.

2.3.1.4 Segment types

The type of a segment depends on the way it is advertised and the network component that it identifies, as shown in Fig. 2.3. Consequently, the segment advertised in the IGP is named the IGP segment. In IP networks, network components are identified by IP addresses which are advertised as IGP prefixes. In SR the SIDs attached to these IGP prefixes are named IGP Prefix Segments (Prefix-SIDs). For example, an SID can be attached to a unicast IGP prefix (*e.g.*, router loopback or an adjacency) as shown in Fig. 2.2, or an IGP anycast prefix (*e.g.*, DNS, CDN, etc.). With the emergence of new use cases, it is likely that new SID types will be defined in the future. In this manuscript, we focus on the two main SID types: Node-SID and adjacency SID (Adj-SID) as shown in Fig. 2.2. Both segment are sufficient to encode any intradomain SR path. They are defined as follows:

- An *Node-SID* is a Prefix-SID, it is a unique identifier (global SID) in the SR domain. It is assigned to a specific node. Specifically, it is attached to one of the node's loopback addresses. Every SPRING node has an entry in its forwarding table for every Node-SID in the SPRING domain. When a node receives a packet with a Node-SID as the active SID, it forwards the packet down the path that results from the IGP path computation algorithm. For example, if the node uses the standard Shortest Path First SPF algorithm, it executes the following forwarding instruction: *forward this packet down the shortest-path to the node that has this SID.*

- An *Adj-SID* is an IGP segment, it is an identifier that has a local significance. It is used to identify an adjacency between two nodes. Only the node advertising it has a forwarding entry corresponding to that SID. When a node receives a packet with an Adj-SID as the active SID in the stack, it executes the following forwarding instruction: *send this packet out of a specific interface.*

A Prefix-SID is associated with a forwarding instruction derived from the routing table computed by node's path computation algorithm. In SR, it is possible to assign Prefix-SIDs per path computation algorithms. Consequently, a SPRING node advertises multiple Prefix-SIDs for one IGP prefix (*i.e.*, one per algorithm), the topological path varies depending on the path computation algorithm used. Currently, two algorithms are defined: type 0 which is a standard SPF based on the link metrics, type 1 which is the a strict SPF, it is identical to a standard SPF, but it requires that the nodes on the path respect the computed path even if it contradict local

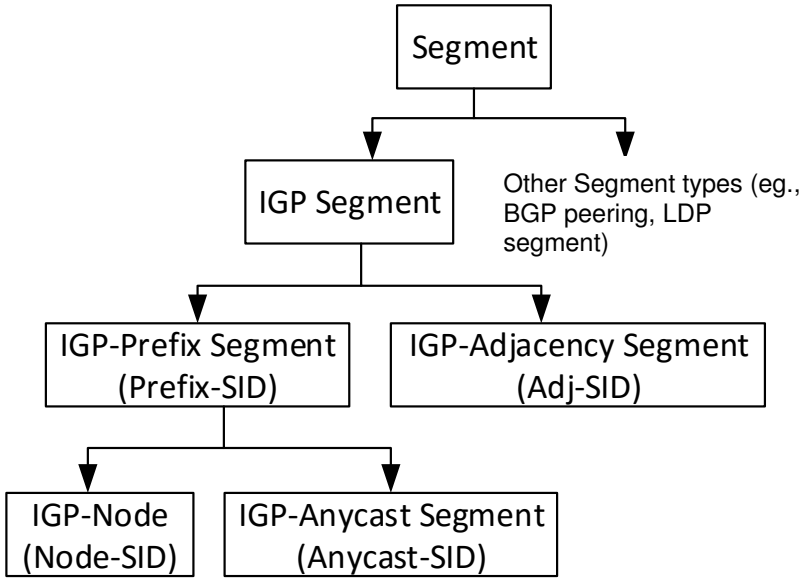


Figure 2.3: Hierarchy of the different segment types

policies. Other algorithms may be defined in the future. For example, algorithms that take into consideration metrics such as delay, jitter or packet loss, etc. However, the per algorithm SID assignment complicates the SR architecture, as it requires to maintain additional states in the network leading to difficulties in configuration and troubleshooting. It is up to the service providers to decide to adopt or not this possibility. For the remainder of the manuscript, we consider that the SIDs are associated with the standard SPF algorithm.

2.3.1.5 Segment Routing Paths

An SR path is composed of an ordered list of SIDs. The path can be automatically computed and instantiated or manually imposed by the network administrator. The path type depends on the type of SIDs used for its construction. For example, a topological path is composed of SIDs that are attached to network components (*e.g.*, node and link). It can be constructed using only multiple adjacency segments (Adj-SID), multiple node segments (Node-SID) or a combination of both depending on the forwarding requirements.

SR paths can be determined and provisioned by different mechanisms and entities, for example with the nodes IGP Shortest Path First (SPF) algorithm, by the network administrator via explicit configuration, or by the Path Computation Element (PCE)

[15]. The SR path is encoded as an SID list in into the packet's header. Each SID becomes active at one of the nodes crossed by the packet. The Active SID is associated with a set of instructions in the forwarding table, which determines what to do the active SID and how the packet is forwarded to the next hop. A global SID may stay active for multiple nodes as seen in Fig. 2.4. Node-SID $P5$ is the active SID at $PE1$, $P6$ and $P5$. The method of transition from one active SID to another depends on the instantiated data plane i.e., MPLS or IPv6, as explained in sections 2.4 and A respectively. If the SID is global, all the SPRING nodes have a set of instructions associated to it. If it is local, then only the node that owns that SID has an association between the local SID and the set of instructions.

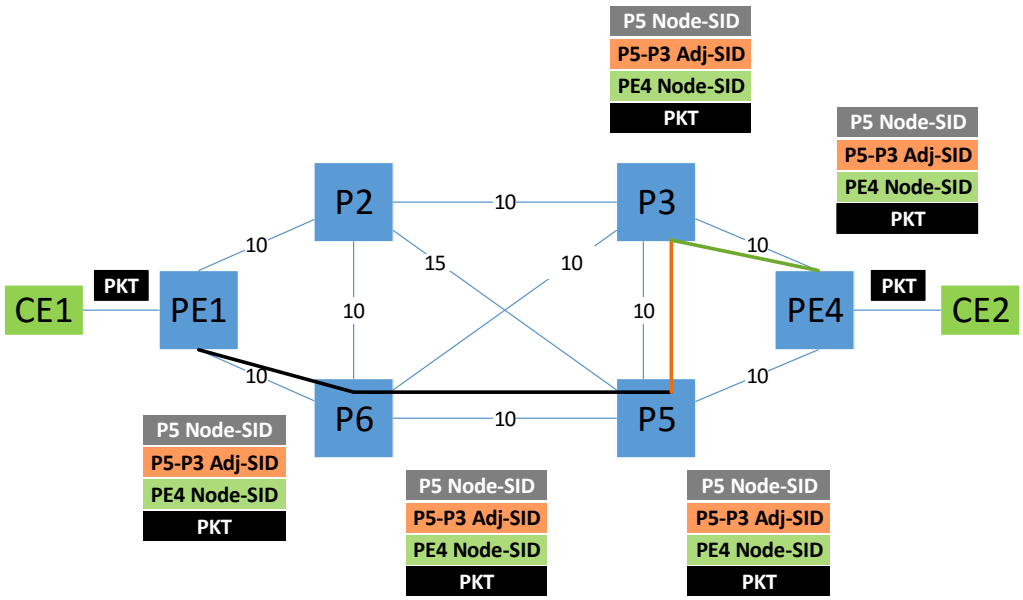


Figure 2.4: An example of Segment Routing Path

In the example shown in Fig. 2.4, the service provider decides to forward the client traffic from $CE1$ to $CE2$ through the path $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$. The selected SR path satisfies the client traffic requirements (*e.g.*, bandwidth, delay, jitter, etc.). The SID stack is pushed by $PE1$ on each packet that belongs to a specific client flow (*e.g.*, IP packets with destination $CE2$). $P5$'s Node-SID is the active SID at $PE1$ and $P6$. Once packets arrive at $P5$, the node recognizes the active SID as its own Node-SID. Applying the loose source routing paradigm required by the $P5$ Node-SID, $PE1$ forwards the packet to $PE6$. Then, $P5$ reads the next SID in the stack, *i.e.*, the Adj-SID $P5 - P3$, which corresponds to an instruction that forces the

packets through the link connecting $P5$ and $P3$, before forwarding the packet $PE4$ Node-SID becomes the active SID. At $P3$, $PE4$ Node-SID is the active SID; PE4 is a direct next-hop to reach PE4 because $P3$ and PE4 are neighbors. At PE4, the SR Path is removed and the packets are forwarded to $CE2$.

2.3.2 SPRING Node Configuration

A SPRING node has to be configured to be able to announce its SR capabilities and exchange SR information. Configuration can be done manually using the node Command Line Interface (CLI) or as shown in Fig. 2.5, via a centralized entity such as the Network Management System (NMS) using management protocols [15]. NETCONF [28] is foreseen as the de facto management/configuration protocol. Therefore, the SPRING working group is working on the definition of a YANG model for the configuration and management of SPRING nodes [29]. A SPRING node requires a minimum set of configuration to be able to communicate with other SPRING nodes. First SR has to be enabled on the node, then the set the SRGB and at least one prefix SID binding.

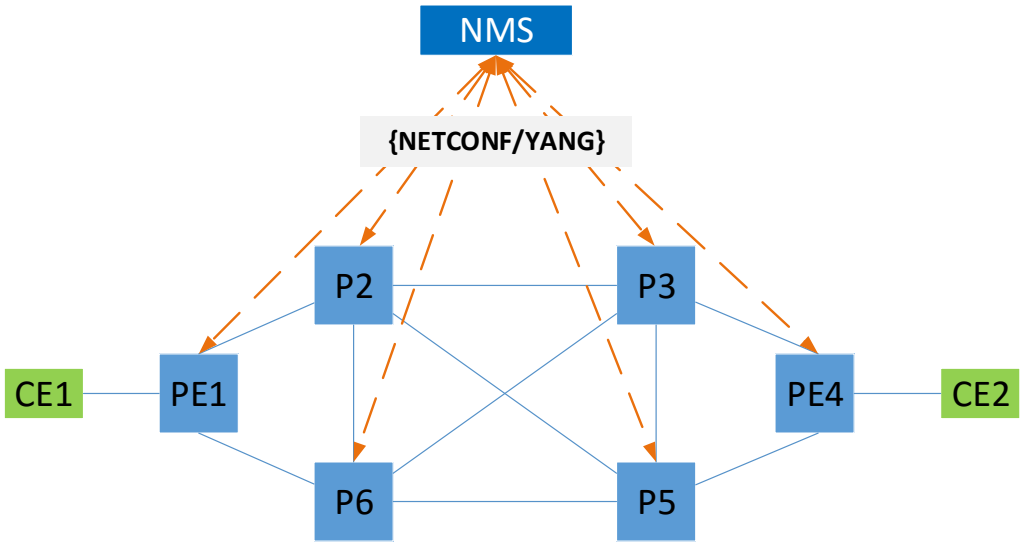


Figure 2.5: Configuration of Segment Routing on nodes using a Network Management System (NMS)

In order to simplify the configuration process and to allow the interoperability with non-SPRING nodes a SPRING nodes can act as Segment Routing Mapping Server (SRMS) [30]. Using a set of IGP extension, the SRMS centrally advertises mappings

between prefixes and and Segment Identifiers (SID) on behalf of other SPRING and non-SR-capable nodes. In Fig. 2.6, node *P*3 plays the role of SRMS. The SRMS node uses its SPRING IGP instance to advertise the bindings (IGP-Prefix, SID) of *P6* and *P5* into the SR domain. SPRING nodes handle the SRMS advertisements as if the Prefix-SIDs were advertised by the nodes themselves.

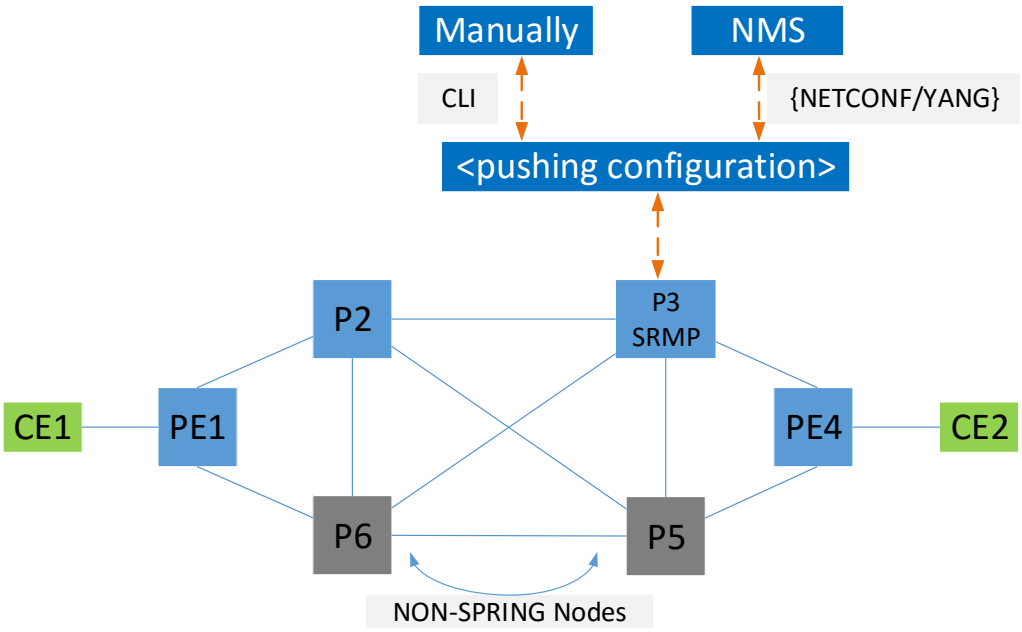


Figure 2.6: Configuration of Segment Routing on nodes using a Mapping Server

As stated before, segment routing does not introduce new protocols. Rather it defines new extensions to the already deployed ones. Once a SPRING node is configured with the SR information, it is ready to participate in the SR domain. It advertises its own SR information (SRGB, prefix SID bindings,) using SR protocol extensions and learns other SPRING node's information. In an intradomain scenario, SPRING nodes exchange their SR capabilities using one of the two widely deployed IGP protocols (OSPF and ISIS), which have been extended to support the advertisement of SR capabilities. In OSPF, SR information (*e.g.*, SRGB, SIDs, etc.) are encoded in a Type Length Value (TLV) format and carried in the opaque Link State Advertisement (LSAs) of type 9, 10 and 11 [31]

2.4 Segment Routing over the MPLS Data Plane

Multiprotocol Label Switching (MPLS) [32] plays a critical role in service provider's core networks in delivering high-performance next-generation services. Additionally, it is agnostic to the client's access links because it encapsulates packets in its own header. MPLS has simple data plane. It is based on a fixed size 20-bit labels and three basic operations: POP, PUSH and SWAP. An ingress MPLS node push labels into each packet that enters the MPLS domain. An intermediate MPLS node switches the packet based on his pre-installed forwarding entries and the label carried in the packet's header. Despite its simple data plane, MPLS's wide use and the increase of supported services has made its control plane complicated and difficult to troubleshooting and to evolve. Traditional MPLS control plane is composed of routing protocols, label distribution protocols and tunnel signaling protocols, causing an important portion of network resources to be consumed just to maintain the protocols soft states. Other issues arise such as synchronization problem between different protocols. SR is regarded as an evolution to MPLS, with its simple, lightweight and SDN-ready control plane. Service providers consider SR as a strong candidate to deliver traditional MPLS services such as Traffic engineering, failure protection (fast reroute), layer two and layer three VPNs.

MPLS's data plane meets all the requirements to instantiate the SR architecture without any hardware upgrade. A segment is represented by a label, the MPLS header already supports label stacking. Consequently, an SR path is encoded as label stack, and the three basic MPLS forwarding operations are sufficient to forward source-routed packets. Therefore, SR deployment is claimed to be straightforward. For example, to enable SR on one of the network nodes (*e.g.*, PE router), a simple software upgrade is sufficient and there is no need to deploy new hardware. Unfortunately, most of the MPLS Label Switching Routers (LSR) do packets processing using ASICs. ASICs have fixed capacities: for example, if the ASIC can only PUSH a small number of labels, this limits the node's capacity when expressing SR paths.

In this section, we discuss the MPLS instantiation of SR (SR-MPLS): how SR architectural components are implemented (*e.g.*, Local SID, Global SID, SRGB, etc.). How SR-MPLS control plane interacts and interoperates with other protocols such as LDP and RSVP-TE.

2.4.1 Segment Routing MPLS Terminology

SR instantiation over the MPLS data plane requires the mapping of the generic concepts defined in the SR architecture to the MPLS components and operations. In this section, we detail SR-MPLS specific terminology and how the SR architecture components are implemented in an MPLS data plane.

In SR-MPLS, an SID is an MPLS label. Therefore, for the remainder of the manuscript, the term SID and label are used interchangeably. The *SRGB* is one range or a concatenation of multiple ranges of local labels allocated by a given node for SR. The advertised SRGB should be large enough to encompass all the global segments. For example, if the network has 100 nodes, then the SRGB size must be greater or equal to 100 (*e.g.*, SRGB == [1000, 1100] or SRGB == [1000, 1049] $\cup$ [2000, 2049]). To avoid label allocation conflict, no other protocol (*e.g.*, LDP, RSVP-TE) is allowed to use a label inside the SRGB. Additionally, a global unique Index (32-bit integer) is attributed to every global segment. This *Index* is combined with the nodes SRGBs to compute the labels allocated for a global segments as explained in section 2.4.2.

In addition to the IGP Prefix Segment defined in the generic SR architecture, SR-MPLS defines new SID types that are specific to MPLS networks such as the LDP LSP segment and the RSVP-TE LSP segment. A *Global SID* is a label within the SRGB and the *Local SID* is a label outside the SRGB. For example:

- A *Prefix-SID* is a global SID, which is attached to an IGP-Prefix (*i.e.*, IP address). It is a label that take its values within the SRGB. The *Node-SID* is a special case of the Prefix-SID; it is a label that identifies a specific network node. It is attached to one of the node's loopback addresses.
- An *Adj-SID* is a label that identifies the adjacency between two nodes. Generally advertised as a local SID unless decided otherwise. The *Group-Adj-SID* is a special case of the *Adj-SID*. It is used for load balancing purposes over all the links/interfaces tagged with the same Group-Adj-SID.

2.4.2 SR-MPLS Global Segment Implementation

SR instantiation over the MPLS data plane translates an SID to a standard MPLS 20-bit label. SR paths are encoded as a stack of labels that get pushed into each packet header to construct the source-routed path. Having a consensus on how to

implement globally significant SIDs for SR-MPLS is one of the challenges that had to face community. Consequently, two main proposal were discussed as follow:

The first proposal is to use globally unique significant labels as SID, this proposition requires that all the LSRs support the entire label range (*i.e.*, $[1, 2^{20}]$), or at least the same sub-range. This proposition was rejected by IETF due to physical restrictions imposed by some vendors on their products. In fact, some LSRs support only a sub range of the entire label space, as the LSR is the one who chooses and advertises to its neighbors the labels to use for every Forwarding Equivalence Class (FEC). Consequently, the LSR is always certain that its neighbors will forward traffic to it with a top label that falls into its supported range. The supported label range is vendor specific. Therefore, a common label range in an operational network where routers come from different vendors may be empty or too small for the network size.

The second proposal, which was adopted by the SPRING WG, consists of allocating and advertising a locally chosen label range for SR called SRGB by all the SPRING nodes. The SRGB can change from one node to another. In addition, a unique index (*i.e.*, 32-bit integer) is associated with each global segment. This index represents the offset of the global segment label within the SRGB. Some service providers have expressed their intention to allocate the same SRGB on all SPRING nodes when possible. This decision reproduces the globally unique significant label concept, which will simplify configuration and troubleshooting.

2.4.3 SID Computation

SR-MPLS uses MPLS labels to identify segments. The different types of SIDs can be advertised as global or local:

A globally significant SID is mapped to a label that takes its value within the SRGB. Instead of advertising a globally unique label, a SPRING node advertises its SRGB and for each global SID, it advertises a globally unique 32-bit index. All the SPRING nodes compute the labels that identify global segments and install them in their Label Forwarding Information Base (LFIB). The local label attributed to the global segment is computed using (2.1), where the segment index is added to the lower bound of node's SRGB.

$$\begin{cases} Global\ SID = Index + SRGB\_lower\_bound \\ \\ Global\ SID \leq SRGB\_upper\_bound \end{cases} \tag{2.1}$$

We consider that all the nodes in the topology shown in Fig. 2.2, reserve the same SRGB $[1000, 2000]$. Each node advertises using the IGP the same SRGB $[1000, 2000]$ and a domain-wide unique index attached to its loopback address. The computation of the global SID (*e.g.*, Node–SIDs) is performed using (2.1). For example, $PE1$'s Node–SID $= 1000 + 1 = 1001$. The results of the Node–SIDs computation as shown in Table 2.1 is the same on all the network nodes.

Table 2.1: Node–SIDs computation with a single Global Block (SRGB) $[1000, 2000]$

Node	Loopback	Global-index	Node-SID
PE1	192.0.2.1/32	1	1001
P2	192.0.2.2/32	2	1002
P3	192.0.2.3/32	3	1003
PE4	192.0.2.4/32	4	1004
P5	192.0.2.5/32	5	1005
P6	192.0.2.6/32	6	1006

A SPRING node may advertise multiple separate labels blocks. Hence, the Node's SRGB is a concatenation of such blocks. For example, the node $PE1$ advertises the label ranges $[1000, 1500]$ and $[2000, 2500]$, thus $PE1$ SRGB is $[1000, 1500] \cup [2000, 2500]$. For the two global indexes 300 and 700 bound to two IGP prefixes, their associated labels at $PE1$ are respectively:

$$Prefix\text{–}SID1 = 1000 + 300 = 1300$$
$$Prefix\text{–}SID2 = 700 - (1500 - 1000 + 1) + 2000 = 2199$$

An SRGB is a local propriety of a SPRING node. Hence, every SPRING node may reserve a distinct SRGB. Consequently, a global segment gets associated with a different label values (*e.g.*, Node-SID) at each SPRING nodes. Now we consider that all the nodes in the topology shown in Fig. 2.2, reserve a distinct SRGB. Each node advertises using the IGP its SRGB and a domain-wide unique index attached to its loopback address. Then a SPRING node uses every node's SRGB to compute the different values of global SIDs (*e.g.*, Node-SID), the results are shown in Table 2.2.

Table 2.2: Node-SIDs computation with different Global Block for each node

PE1 SRGB [1000,2000]	
Node	Node-SID
PE1	1001
P2	1002
P3	1003
PE4	1004
P5	1005
P6	1006

P2 SRGB [3000,4000]	
Node	Node-SID
PE1	3001
P2	3002
P3	3003
PE4	3004
P5	3005
P6	3006

P3 SRGB [5000,6000]	
Node	Node-SID
PE1	5001
P2	5002
P3	5003
PE4	5004
P5	5005
P6	5006

PE4 SRGB [7000,8000]	
Node	Node-SID
PE1	7001
P2	7002
P3	7003
PE4	7004
P5	7005
P6	7006

P5 SRGB[9000,10000]	
Node	Node-SID
P1	9001
P2	9002
P3	9003
P4	9004
P5	9005
P6	9006

P6 SRGB[11000,12000]	
Node	Node-SID
PE1	11001
P2	11002
P3	11003
PE4	11004
P5	11005
P6	11006

For a local SIDs, no computation is performed because they are advertised as labels. A local SID takes its value outside the node's SRGB. A LFIB Forwarding entry for a local SID is only installed by the node that advertises it *i.e.*, only the owner of the local SID knows how to forward packets using it.

2.4.4 Forwarding Operations

SR-MPLS uses the MPLS forwarding plane. Consequently, SR packets get manipulated using the MPLS data plane operations PUSH, POP (NEXT) and SWAP (CONTINUE):

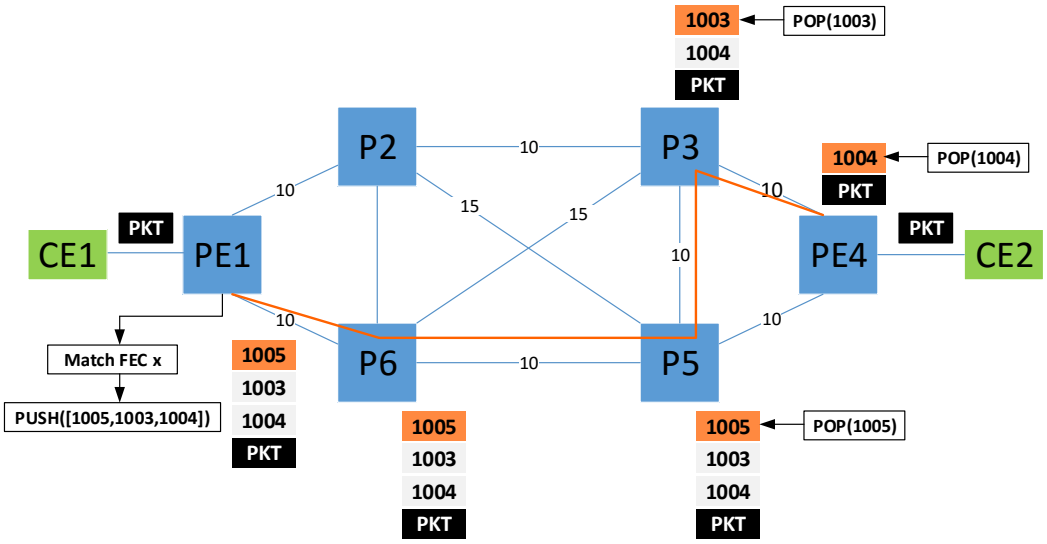


Figure 2.7: Segment Routing operations

- PUSH: the push operation is performed by the ingress nodes (LER), which encodes into the packet header the list of labels that compose the SR path. Node *PE1* in Fig. 2.7, uses the PUSH operation to encode the path description into the packet's header. It is also used by intermediate nodes to add one or more additional labels for rerouting and protection purposes.

- POP (NEXT): when the pop operation is performed, the active label is removed from the label stack. It is performed when the packet's active label belongs to the current node (*e.g.*, Node-SID. Adj-SID or Group-Adj-SID), or when the penultimate-hop-popping is enabled for the next segment. Node *P5* in Fig. 2.7, pops the labels *1005* because it is its own Node-SID. In SR implementation, the POP operation is called *NEXT* because popping the top-level label means that the next label will be pointing to the next segment in SR path.

- SWAP (CONTINUE): the swap operation is performed to replace the active label with a new one. In the SR-MPLS, it replaces the current node's local label of a global SID by the next node local label of the same global SID. It is called *CONTINUE* because the old and the new labels point to the same segment, they just belong to two different SRGBs.

In the next section, we detail how segments are used to populate the forwarding plane entries, the operations that are performed upon the reception of a packet and the forwarding decision-making mechanism.

2.4.5 SR-MPLS Forwarding Entries Installation

Forwarding entries for SR are derived from the routing table and SR information (*i.e.*, SIDs, SRGB), which is either configured or learned using the SR control plane protocols (OSPF, ISIS, BGP-LS). The process of installing forwarding entries into the forwarding table is depicted in Fig. 2.8. It explains the transition from the SR information to the installation of forwarding entries into the forwarding table of a SPRING node, different vendors may implement it with respect to their router's physical architecture.

A SPRING node is configured with its own SRGB and SIDs. Additionally, it receives the SRGBs and SIDs (local or global) received via the IGP from other nodes. Forwarding entry installation process starts by identifying the SIDs. There are SIDs that are advertised as labels (*e.g.*, Adj-SID), and there are those that need to be computed (*i.e.*, global SIDs) using (2.1) based on the *index* and the *SRGB*.

Let us consider that the node receives a SID advertisement. At step 100, the node checks if the SID is within its SRGB. If No, then it's a local SID. At step 200, determine if the local SID belongs or not to the current node. If no, no corresponding forwarding entry for that SID will be installed. At step 201, the local SID belongs to the current node, now its type needs to be determined because the forwarding entry depends on the type of the local SID:

- At step 210, the SID is identified as an Adj-SID. Therefore, the node adds the following entry to its forwarding table: set the Adj-SID as incoming label and the operation is POP (Adj-SID) then forwards the packet out the interface associated to the Adj-SID.

- At step 220, the local-SID is identified as a Group-Adj-SID. Therefore, the node adds the following entry to its forwarding table: set the Group-Adj-SID as the incoming label and the operation is POP (Group-Adj-SID) then load-balances the packets out of the interfaces associated with the Group-Adj-SID.

- For other local-SID types, a corresponding behavior is to be defined.

At step 300, the SID value is within the current node SRGB thus it is a global SID. At step 310, the SID is determined to be the current node's Node-SID, therefore, the installed forwarding entry is: set the current Node-SID as the incoming label, the operation is POP (Node-SID) then instructed to look for a forwarding entry with as incoming label the new active SID.

At step 301, the owner of the global SID is identified then the next hop to reach that node is derived from the current node routing table. At step 302, check if the next hop's SRGB equals the current node one. If No, go to step 303, compute the equivalent global SID for the active SID using its index and the next hop SRGB. At step 304, check if Penultimate hop popping (PHP) is enabled for that SID. If no, install the following forwarding entry: set the active SID as the incoming label and the operation is SWAP (active SID, equivalent SID in the next hop) then forwards the packet out the interface to reach the next hop.

At step 305, check if the global SID belongs to the directly connected neighbor. If yes, install the following forwarding entry: set the active SID as the incoming label and the operation to POP (active SID) then forwards the packet out the interface to reach the next hop.

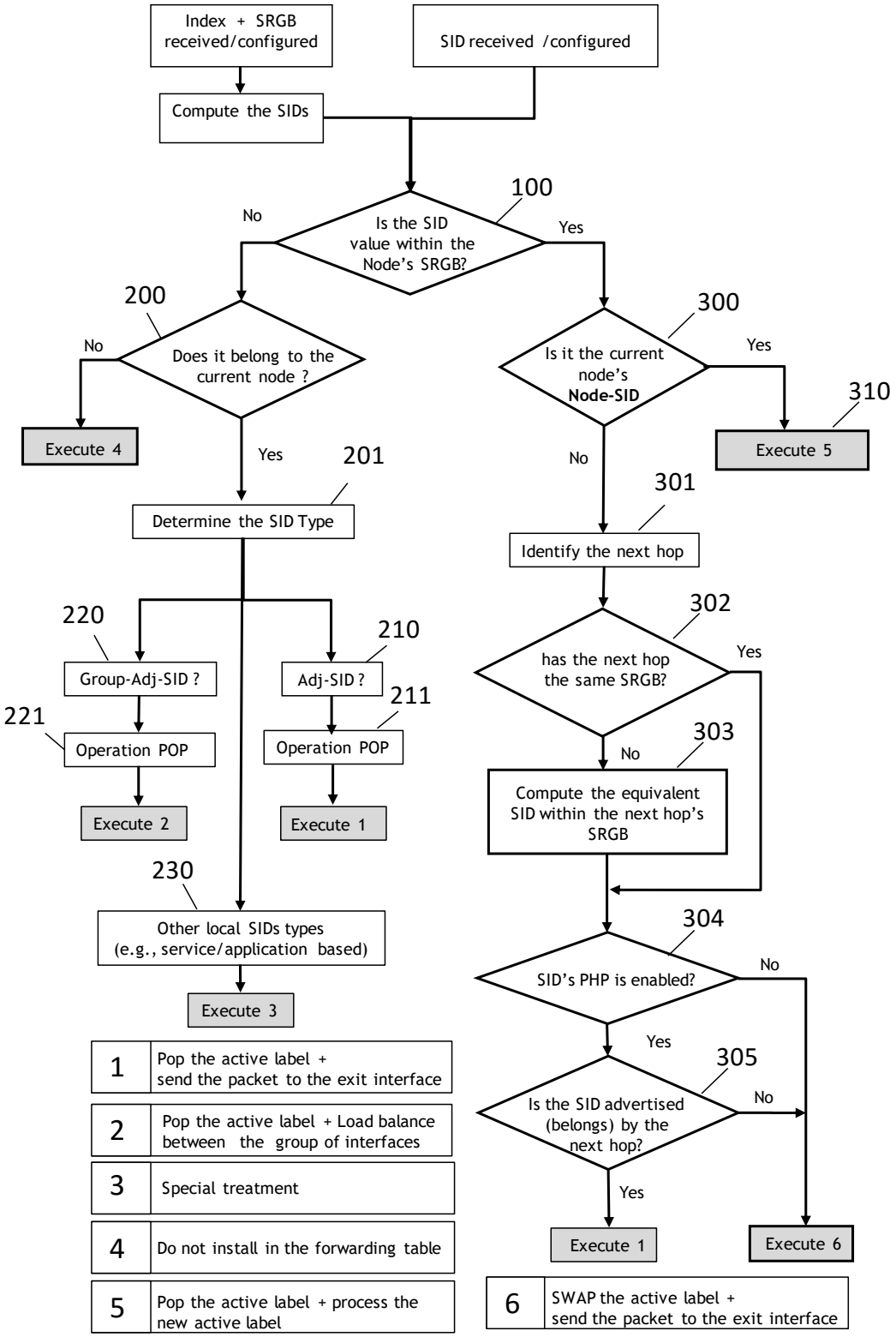


Figure 2.8: State Machine of SR-MPLS Forwarding Entries Installation

Example : Let us consider the case where all the SPRING nodes in Fig. 2.2 allocate different SRGBs (*i.e.*., per-node SRGB). $P2$ has the SRGB $[3000, 4000]$ and is configured with $index2$ attached to its loopback address 192.168.0.2/32. $P2$ computes the local label of its own Node-SID using (2.1) : $P2$'s $Node-SID = 3000+2 = 3002$. $P2$ installs in its forwarding table the following : $Incominglabel = 3002$ operation = $POP(3002)$

P2 learns via its IGP adjacency with $P3$, $P3$'s SRGB $[5000, 6000]$ and $Index3$ attached to $P3$'s loopback address 192.168.0.3/32. $P2$ constructs the forwarding instruction that will be used to forward traffic to $P3$. the traffic destined to $P3$ reaches $P2$ with the top label the $Node - SID$ of $P3$, the label value is inside $P2$'s SRGB. Therefore, the incoming label is equal to $P3$'s Node-SID inside P2's SRGB : $3000 + 3 = 3003$.

$P3$ allocate a different SRGB. Therefore, when P2 sends a packet to $P3$, the top label (active SID) has to be either inside $P3$'s SRGB or one of its local SIDs (*e.g.*, Adj-SID). Consequently, P2 has to swap P3's Node-SID (3003) with its equivalent computed inside $P3$'s SRGB *i.e.*, $5000 + 3 = 5003$

Finally P2 installs the following forwarding entry: *Incoming label = 3003, Operation = SWAP(3003,5003), Exit interface = P2-P3.*

In this section, we gave an overview on how the SR SIDs are translated to forwarding plane entries. In the next section, we detail different techniques to encode SR paths and the forwarding plane operation attached to each SID type.

2.4.6 MPLS Routing Source-routed Path

In SR-MPLS, the segment list that constructs the source-routed path is encoded as a stack of labels into the packet header. The SR Path can be expressed using one or multiple labels (SIDs). The computation of the label stack is addressed in [33, 34]. An LSR forwards packets based on the top label in the stack and the corresponding forwarding entry in the LFIB. The top label represents the active segment. It corresponds to the label number n in the label stack as shown in the Fig. 2.9. The last label in the stack has position 1 and its S bit is set to 1 to indicate that it is the last label in the stack.

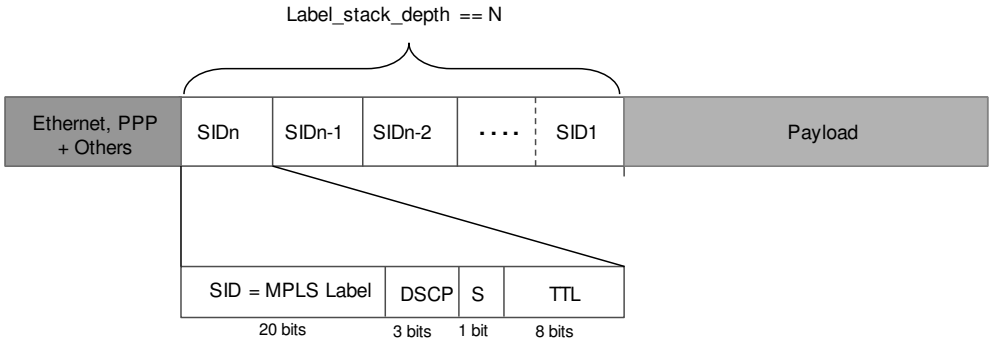


Figure 2.9: Packet header when using MPLS data plane for Segment Routing

When a packet is received, it gets matched to a specific FEC by the SR ingress node, which is typically a PE. Each FEC is associated with a source-routed path (list of SIDs). The source-routed path is expressed as a stack of labels that get pushed onto the packets header. Each label represents a forwarding instruction. The combination of all the instructions forwards the packet from the ingress node to the egress.

We use the same network topology as in Fig. 2.2. For simplification purposes, all the service provider nodes are SR-enabled and all the nodes allocate the same SRGB [1000, 2000]. Table 2.1 summarizes the result of Node-SIDs computation of all the network nodes. A SR path can be expressed using one SID type or a combination of SID. In following list of scenarios, we explain the different methods to express a SR path:

Scenario 1: Encoding a SR path with one Node-SID

Depending on the algorithm and the criteria used to compute the Source-Routed Path, the resulting path can be different or equal to the IGP's Shortest Path computed using the SPF algorithm. In the case where the resulting path is equal to the IGP's shortest path, then the SR path can be expressed using just one global SID. Consequently, the label stack contains only one label. In the example shown in Fig. 2.10, the chosen source-routed path to forward the client traffic between $CE1$ and $CE2$ follows the IGP shortest path: $PE1 \rightarrow P2 \rightarrow P3 \rightarrow PE4$. Therefore, PE4's Node-SID 1004 is sufficient to express the SR path. As the packets pass through $P2$ and $P3$, the label 1004 is maintained. In the case of a different per-node SRGB, the label 1004 would be swapped with a different label value *i.e.*, an equivalent label within the next hop's SRGB.

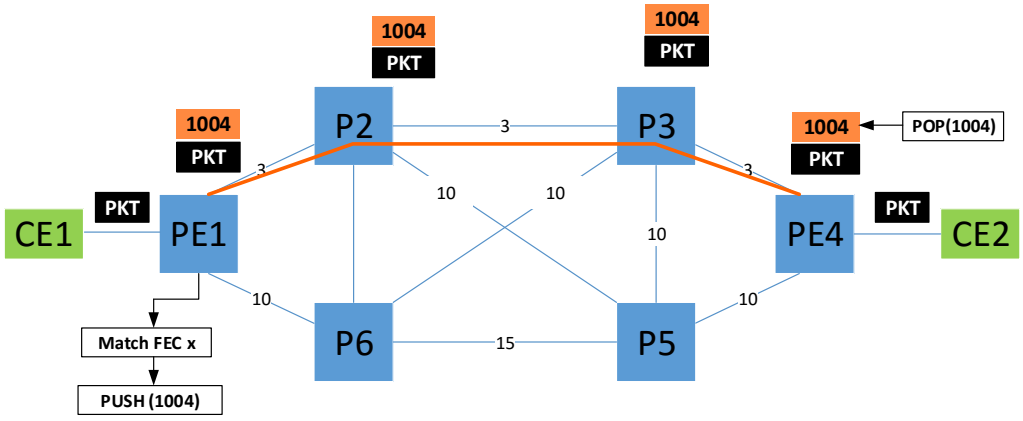


Figure 2.10: Example of Segment Routing path with one Node-SID only: Routers use the same Segment ID to forward the packet following the shortest path up to the Node SID.

Scenario 2: Encoding a SR path using only Node-SIDs

An SR path can be also expressed exclusively using Node-SIDs. This scenario is the generalization of scenario 1. It is not necessary to indicate all intermediate nodes Node-SIDs to construct the SR path, this results in a loose SR path. When two nodes are not adjacent, the forwarding decision between their corresponding Node-SIDs follows the IGP Shortest Path. This approach helps reduce the label stack depth. Consequently, the resulting SR path is very sensitive to the IGP changes (*e.g.*, changing one of the link's IGP Metrics). Note that some paths cannot be expressed using only Node-SIDs. In the scenario shown in Fig. 2.11, the chosen source-routed path to forward the client flow between $CE1$ and $CE2$ is $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$, which is not the IGP Shortest Path. Here, we supposed that: the IGP's shortest path between $PE1$ and $P5$ is via $P6$, the link between the neighbors ($P5$, $P3$) and ($P3$, $PE4$) is the shortest path between them. Thus, client traffic can be source routed using three Node-SIDs 1005, 1003, 1004.

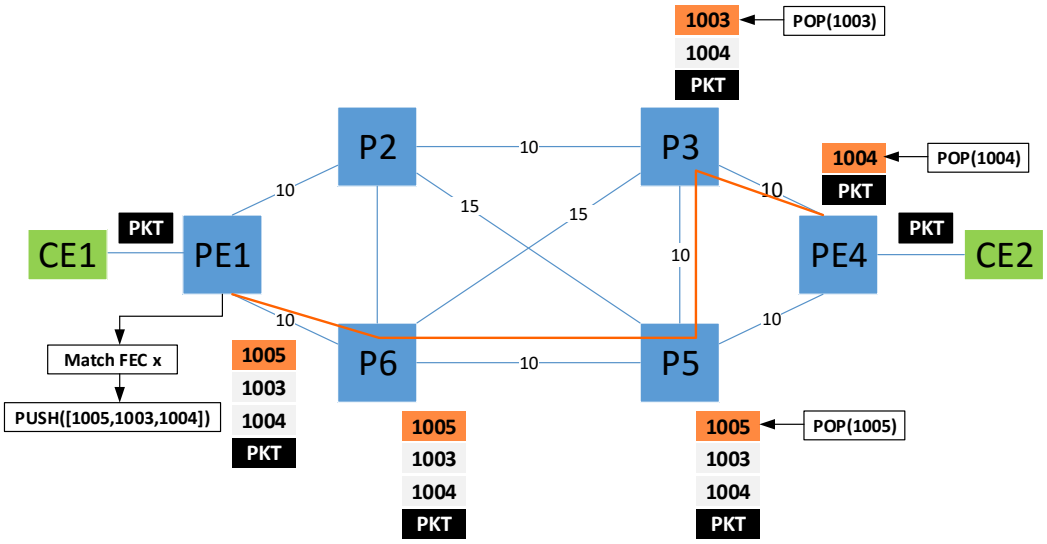


Figure 2.11: Example of Segment Routing path where the label stack is composed only of Node-SIDs

Scenario 3 Encoding a SR path using only Adj-SIDs

An SR path can be expressed also exclusively using Adj-SIDs. This is used to create strict paths that do not get affected by the IGP decision process. The Adj-SID labels are only installed in the forwarding tables of nodes advertising them. Expressing SR path in this way may result in a large label stack; this has numerous side effects such as some incompatibility with LSRs that cannot push a large label stack. It also increases client packet fragmentation if the MTU is not set properly, and may cause load-balancing problems. We will tackle this problem in chapter III.

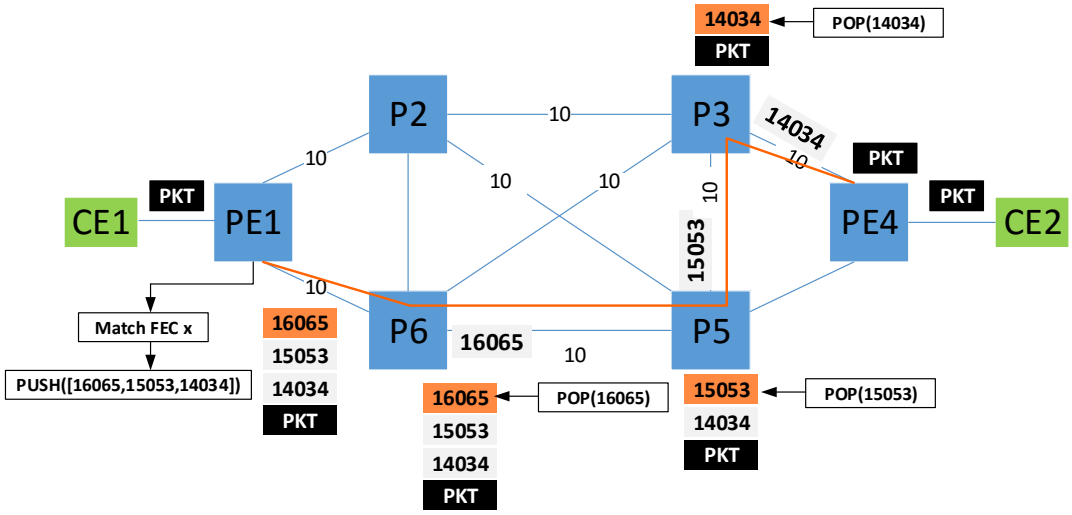


Figure 2.12: Example of Segment Routing path where the label stack is composed of only Adj-SIDs

In the scenario shown in Fig. 2.12, the chosen source-routed path to forward the client flow between $CE1$ and $CE2$ is $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$; the SR path can be expressed exclusively using Adj-SIDs: $P6 - P5$: 16065, $P5 - P3$: 15053 and $P3 - PE4$: 13034. When $P6$, $P5$ or $P3$ receives a packet and recognizes the top label as one of its Adj-SID, it performs the POP operation then forwards the packet out of the interface associated with Adj-SID.

Scenario 4 : Encoding a SR path using a mix of Node-SIDs and Adj-SIDs

In this scenario, a mix of Node-SIDs and Adj-SIDs are used to express the SR path. A Node-SID is used to forward the packet through the IGP paths, the use of Node-SIDs when possible reduce the size of the label stack. An Adj-SID is used to force the packets out of a specific interface, for example, if desired not to follow the IGP.

In order to forward the traffic through the $PE1 \rightarrow P2 \rightarrow P3 \rightarrow P5 \rightarrow PE4$ path as shown in Fig. 2.13, three labels 1003,9000,1004 are used to construct the SR path; label 1003 is the Node-SID of $P3$, the packets will follow the IGP shortest path to reach $P3$. At $P3$, the IGP shortest path between $P3$ and $P5$ is through PE4, *i.e.*, using $P5$'s Node-SID, $P3$ would forward packets down that path. To force the traffic to pass from $P3$ to P5, the Adj-SID (label 9000) allocated by $P3$ to the adjacency $P3 - P5$ is used. At $P5$, label 1004 used to forward the packet to PE4.

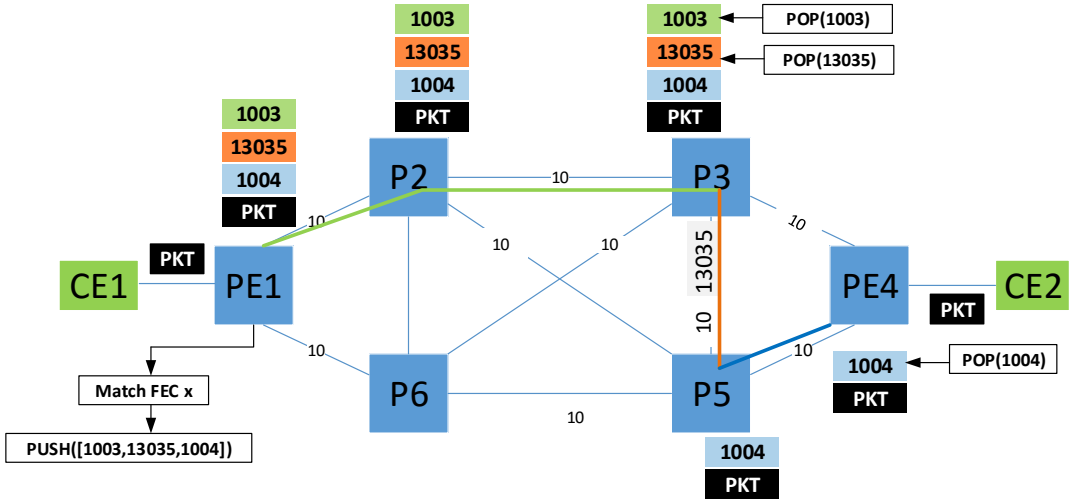


Figure 2.13: Example of Segment Routing path where the label stack is composed with node-SIDs (PE1 to P3 and P5 to PE4) and Adj-SIDs (P3 to P5)

Scenario 5: Load balancing over a SR path using the Group-Adj-SID

Group-Adj-SID is a special case of the Adj-SID; it is a locally significant label that represents a set of parallel adjacencies or interfaces; it is used for load balancing purposes. In the example shown in Fig. 2.14, $P2$ allocates the label (Group-Adjacency-Label) 9000 for its adjacencies with $P3$. $PE1$ sends the client traffic to $P2$ with the top label 1002. $P2$ pops its own Node-SID label 1002, then processes the next top label 9000 which matches a forwarding entry with the instruction to load balance that traffic between the two links connecting $P2$ to $P3$.

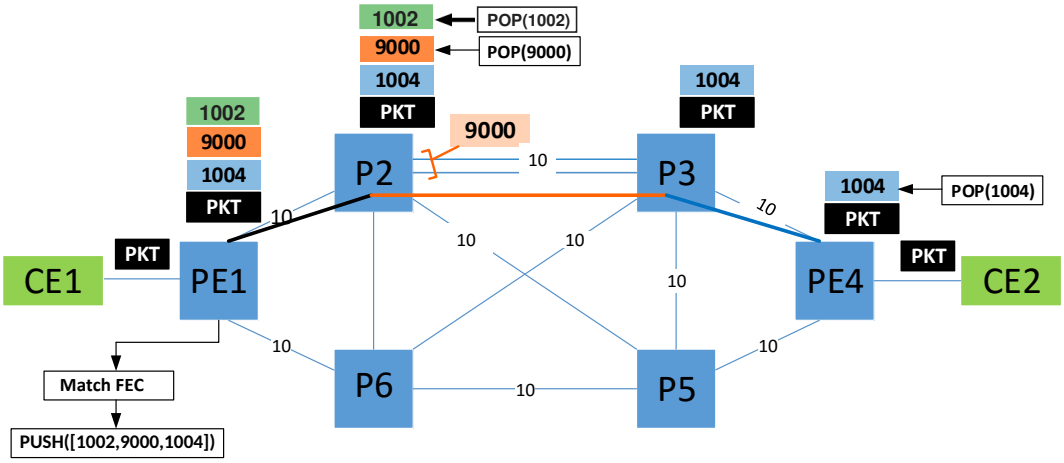


Figure 2.14: Load balancing of flows using Group-Adj-SID

2.4.7 Anycast with SR-MPLS

An anycast group is a set of SPRING nodes that offers the same service inside a network (*e.g.*, DNS servers). Nodes that belong to an anycast group get attributed the same IPv4 prefix, and then a unique index is attached to that prefix. The Anycast-SID is computed by every node using (2.1).

All the nodes of the anycast group advertise the same prefix and its index using the IGP update messages. Upon receiving multiple updates for the same prefix from different nodes, a node computes the shortest path to reach the closest node of that anycast group. Therefore, every packet with an Anycast-SID as the top label is forwarded to the nearest node member of that anycast group.

For anycast to work properly in an SR-MPLS network, all the network nodes have to use the same SRGB, so that all the global SIDs (*e.g.*, Anycast-SID) have the same label values across the network nodes. However, a unique SRGB is not mandatory in SR. Let us take the example depicted in Fig. 2.2, and Table 2.1. All the SPRING nodes reserve the same SRGB $[1000, 2000]$, $P3$ and $P5$ belong to the same anycast group, for that they both advertise the anycast IGP prefix 192.168.0.100/32 with the index 100, the label allocated for that Anycast-SID by all the nodes is 1100. If service provider chooses to load balance $CE1$ to $CE2$ traffic between the two SR paths: $PE1 \rightarrow P2 \rightarrow P3 \rightarrow PE4$ and $PE1 \rightarrow P2 \rightarrow P5 \rightarrow PE4$, $PE1$ has to push the label stack: 1100,1004, before forwarding the packet to $P2$. $P2$ load balance traffic between its two ECMP path to reach the Anycast-SID 1100: $P2 \rightarrow P3$ and $P2 \rightarrow P5$. Regardless, who receives the packet $P3$ or $P5$, they both recognize the active label 1100 as theirs and therefore pops it, the new active segment 1004, falls into the SRGB of both $P3$ and $P5$. Consequently, a corresponding forwarding entry with 1004 as the incoming label exist on both $P3$ and $P5$. PE4 receives the packet either from $P3$ or $P5$, it pops the label 1004 and forwards the packet to $CE2$.

However, routers from different vendors may allocate different SRGBs. For example, if $P3$ allocates the SRGB $[5000, 6000]$ and $P5$ $[9000, 10000]$. Hence, the labels allocated to global SIDs is not unique in the SR domain, and this breaks the SR anycast. The problem is that $P3$ and $P5$ compute different Node-SID labels for PE4, *i.e.*, 5004 at $P3$ and 9004 at $P5$, the load balancing decision happens at $P2$. Therefore, it is not possible at $PE1$ to determine which of PE4's Node-SID labels: 5004 or 9004 to push after the Anycast-SID label 3100, a solution to this problem is proposed in [35]. However, this proposition adds more complexity to SR architecture and requires that routers maintain additional states.

2.4.8 Interoperability and co-existence

Service providers can choose to deploy SR in a tactical manner where SPRING and non-SPRING nodes co-exist. For example, the service provider may choose to enable SR only on the PE nodes [36]. Two deployment scenarios have been identified:

- Scenario 1: in an MPLS network where only a subset of its nodes are SR-enabled. Interoperability between SR and LDP is needed. To do that, the service provider can use the SRMS to attribute and advertise Prefix-SIDs owned by non-SPRING nodes on their behalf. Then a SPRING node sending traffic to a non-SPRING swaps the Prefix-SID attributed by SRMS to the LDP label of the non-SPRING [36, 37].

- Scenario 2: if the non-SPRING nodes do not use the same data plane used by SPRING nodes, such as SPRING nodes with an MPLS data plane and non-SPRING nodes with IP forwarding. In this case, a tunneling mechanism is used: the traffic exchanged between two SPRING nodes through a non-SPRING node is tunneled (*e.g.*, encapsulation of MPLS in IPv4).

The first scenario allows a gradual and smooth deployment of the SR mechanism in an existing IP/MPLS network. This is an important feature of the technology, which will greatly help its fast and large adoption by operators.

Physical limitations on some equipment may get in the way or slow down SR deployment. For example, some LSRs are limited by the number of labels they can push onto a packet (*e.g.*, maximum 4 labels) [4], this is known as the Maximum SID Depth (MSD). Consequently, SR paths expressed with more labels than the ingress LSR's MSD (*i.e.*, $SR\ path\ size > MSD$) cannot be used, which may lead to the augmentations of traffic demands rejections and the overload of some network links. We address the MSD limitation problem in chapter III and IV.

MPLS LSRs use a key value to do load balancing; the key can be computed over a set of the packet field (IP source, IP destination, ports, etc.), which requires a deep inspection of the packet. However, some LSRs can inspect a limited number of labels called Readable Label Depth (RLD). Consequently, this may lead to an incorrect key value, which impacts the performance of the load balancing function. This results in sending packets that belongs to the same flow (*e.g.*, TCP session) over different paths, causing issues such as jitter, latency, and packet misordering. The other option is to use the Entropy Labels (EL) mechanism described in [38], where the ingress LSR computes the load balancing key and inserts it as a label into the label stack. A

transit LSR can use the EL for load balancing only if it is within its RLD. An ingress LSR may push one or multiple EL pairs in addition to the SR path onto the label stack; for the SR use case an algorithm on how to insert one or multiple ELs in order to respect the transit LSRs RLDs is defined in [39]. The overhead added by ELs increase the label stack size which worsens the MSD problem.

In this section, we gave the specifics of the SR-MPLS. We detailed the control plane and the data planes processes and how information collected by the control plane protocols are translated to data plane forwarding operations. In the next sections, we detail several use cases here SR has an important impact.

2.5 Use Cases

SR adoption is related to the maturity and interoperability of the implementations. Additionally, the SPRING working group focus on the uses cases put forward by service providers such Orange. In this section, we focus on the first uses cases that have been specified as the first scenarios to highlight the benefits of Segment Routing. The very first one that got a lot of attention and support, especially from service providers, is the link and node protection, which enables efficient fast recovery in case of failure. The second one is the VPN scenario, where SR simplifies the deployment of VPN. The third one is Traffic Engineering (TE), *i.e.*, how SR is used to constrain paths. The last one is the network monitoring and measurement use case. This list is not exhaustive and other use cases have been or will be considered later on.

2.5.1 Fast Reroute with Segment Routing

SR provides automatic traffic protection without any topological restrictions. The network can protect traffic against link and node failures without requiring additional signaling in the network. Existing IP Fast Reroute (FRR) technology, in combination with the explicit routing capabilities in SR, Furthermore, SR FRR mechanism known as Topology Independent Loop Free Alternate (TI-LFA) [40] enhances the classical IP-FRR solutions to provide 100 percent protection coverage as opposed to the LFA's varieties: LFA, remote LFA (RLFA) and directed LFA (DLFA).

In fact, a SPRING node automatically precomputes post convergence recovery paths for each segment (*e.g.*, Node-SID, Adj-SID) affected by the link or node failure, this is done with minimal operational impact. There is no need for control plane protocols to establish and maintain the protection paths (*e.g.*, directed LDP sessions or RSVP-TE tunnels). In SR-MPLS, the protection path takes the form of a stack of

labels. Depending on the type of failure link or node, a SPRING node computes the post convergence protection path to bypass the failure. Any type of protection can be computed: link protection, node protection, SRLG protection. The operations used to activate the protection depend on the active SID (top label) in the initial SR path: push the protection label stack, swap the active label with its equivalent in next hop of the protection path, or pop the active label and push the recovery path.

In the example shown in Fig. 2.15, relying on SR-MPLS, $PE1$ pushes one label 1004 on the packet of $CE1$ to reach $CE2$ using the IGP shortest path $PE1 \rightarrow P2 \rightarrow P3 \rightarrow PE4$. The node $P2$ installs in its forwarding table alternate entries to reroute the traffic via $P5$ in the case of the link $P2 - P3$ failure. If $P2$ detects that $P3$ is unreachable (*e.g.*, detected by the Bidirectional Forwarding Detection (BFD) protocol). Then, to reroute the traffic, $P2$ pushes the label 15035 (the Adj-SID attached to the adjacency $P5 - P3$) into the packets headers, and forwards it to $P5$. $P5$ pops the label 15035 and then forwards the packets to $P3$. From the $P3$ to $P4$ the packets use $P4$'s Node-SID: 1004 to reach P4 via the IGP shortest path.

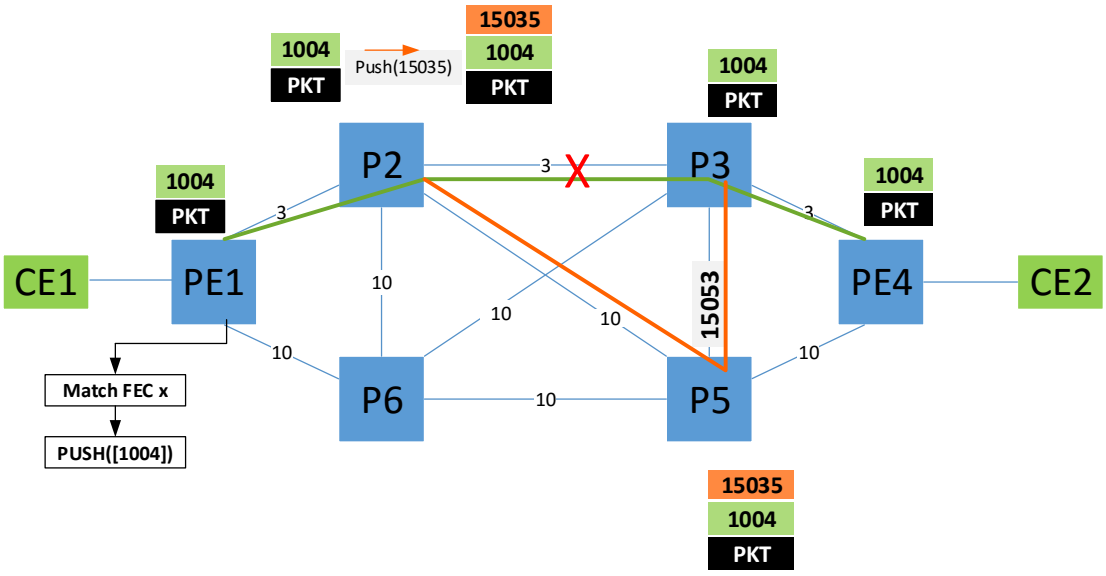


Figure 2.15: Local protection of the link between P2-P3 with Segment Routing

If $P2$ is configured for node protection, to reroute the traffic it swaps the NODE-SID of PE4 with its equivalent value at $P5$ and forwards it to $P5$ to force the SR path to bypass $P3$ by following the $P5 \rightarrow PE4$. In both cases, SR delivers under 50 milliseconds fast reroute without the need for pre-established (signaled) reroute paths.

Moreover, in our example, the failure is circumvented before $PE1$ gets the information; $P2$ detects and reacts quickly to the failure (reacts to the link $P2 - P3$ failure) by redirecting the traffic onto the $P2 \rightarrow P5 \rightarrow P3$ backup path. Next, $P2$'s IGP converges and propagates the failure information to its adjacent nodes, $PE1$ gets the information about the failure and decides to re-compute a new post convergence SR path with the same destination PE4. Once the new SR path (*i.e.*, $PE1 \rightarrow P2 \rightarrow P5 \rightarrow PE4$) is in place, $PE1$ sends the client packets with the new label stack.

2.5.2 IGP-Based MPLS Tunneling

MPLS is mainly used by service providers to provide VPN services for customers to connect their distant sites and enable QoS in their core network, possibly using MPLS-TE. SR simplifies the deployment of such services. In this section, we make a comparison between SR and LDP/RESVP-TE to establish an MPLS VPN connecting two client sites through the service provider's core network.

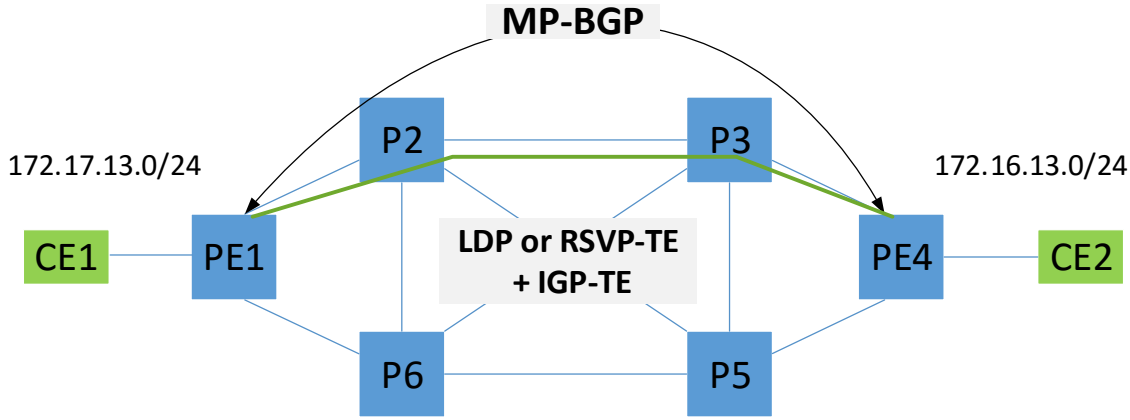


Figure 2.16: Standard MPLS VPNs with LDP or RSVP-TE for labels distribution

1. Scenario using LDP or RSVP-TE: A shown in Fig. 2.16, LDP may be used in the service provider's core network to exchange MPLS labels and establish a full mesh of multipoint to point Label-Switched Paths (LSPs) that connect PE routers; LDP LSPs follow the IGP (OSPF, ISIS) shortest path between the LSP's head-end (ingress) and the tail-end (egress) Label Edge Routers (LERs). RSVP-TE is used to establish tunnels are point to point; also with RSVP-TE, it is possible to establish tunnels that do not follow the IGP shortest path.

In addition, Multi-Protocol BGP (MP-BGP) is enabled on the edge routers establishes a BGP session over the core network between $PE1$ and $PE4$. Then, $CE1$ advertises the prefix 172.17.13.0/24 to $PE1$, which installs that prefix in the appropriate Virtual Routing Function (VRF), and binds it to the VPN label 50000 and announces it to PE4 using MP-BGP. Finally, $PE1$ installs the label 50000 in its forwarding table with an exit interface to $CE1$. PE4 considers $PE1$ as the next hop for the prefix 172.17.13.0/24; all the traffic from $CE2$ to destination $CE1$ will be tagged with the VPN label 50000. The same process happens for the $CE2$ prefix 172.16.13.0/24 for the reverse path.

LDP or RSVP-TE establishes a tunnel between $PE1$ and PE4 based on their loopback addresses. For example, PE4 learns using the LDP that to reach $PE1$ it must use $P3$ as the next hop with label 4000. Now, when $PE4$ receives traffic from $CE2$ and the destination address matches 172.17.13.0/24, $PE4$ pushes two labels on the packet $\{4000, 50000\}$. Label 4000 is used by $P3$ to identify the next hop of the LSP to reach $PE1$ following the IGP path. At $PE1$, the label 50000 is used to forward the traffic through the exit interface to $CE1$.

Alternatively, RSVP-TE could be use to build a tunnel between $PE1$ and $PE4$ that respects client QoS requirements. similarly to the LDP example, two labels are pushed by $PE4$: The VPN label (*e.g.*50000) advertised by $PE1$ using MP-BGP and the RSVP-TE tunnel label (*e.g.*45000).

2. SR scenario: SR is enabled in the core network as shown in Fig. 2.17, For simplification purposes all nodes are SR-enabled; also all the nodes allocate the same $SRGB[1000, 2000]$. Each PE router is allocated a Node-SID associated with its loopback as shown in Table 2.1. IGP is used to exchange SR information (SIDs, SRGB, etc.), which simplifies the control plane, management and the troubleshooting of the network by eliminating the need for signaling protocols such as LDP and RSVP-TE. It also resolves the synchronization problem between LDP and IGP, which is not an easy thing to do [41]. Finally, similar to LDP, ECMP load balancing between the available paths is a native function in SR and does not require any additional configuration.

   MP-BGP is enabled only on the LERs: $PE1$ and $PE4$. It is used to establish BGP sessions between the network edge routers. $CE1$ announce its prefix 172.17.13.0/24 to $PE1$, which installs the prefix in the appropriate VRF, binds it with the label 50000 then advertise it to $PE4$ using MP-BGP.

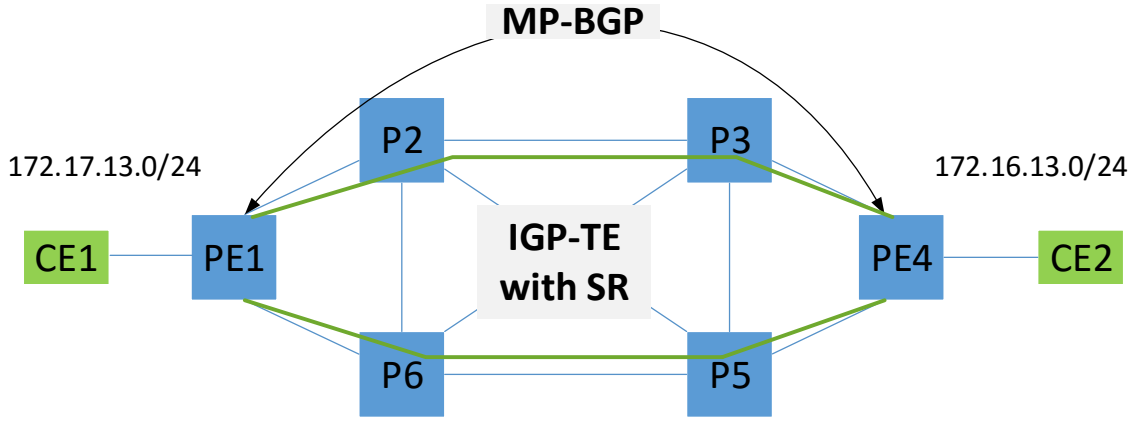


Figure 2.17: Segment Routing based MPLS VPN:MP-BGP for VPN labels exchange, no need for RSVP or LDP as the SIDs are advertised using the IGP

$PE4$ uses the index 1 attached to $PE1$ loopback address with its associated SRGB; it computes the label 1001 to reach $PE1$ via the ECMP shortest path. PE4 pushes the label stack {1001, 50000} into the packet header sent by $CE2$ to $CE1$. All the core network nodes are running SR, which means that they all have an entry in their forwarding table for the label 1001. consequently, multiple paths to reach the same destination may exist. Therefore, client traffic may be load balanced between the two available equal cost paths to reach $PE1$: $P3 \rightarrow P2 \rightarrow PE1$ and $P5 \rightarrow P6 \rightarrow PE1$. $PE1$ receives the packets pops its Node-SID label 1001 then processes VPN label 50000 to finally forward the packets via the appropriate exit interface to $CE1$.

SR-MPLS simplifies VPN services delivery, by reducing the control plane protocols overhead and eliminating the need for LDP/RSVP-TE signaling.

2.5.3 Segment Routing Traffic Engineering

Compared to the control plane of MPLS-TE, SR is not equipped with a resource (mainly bandwidth) reservation mechanism such as RSVP-TE, as no signaling is used and no per-flow states are maintained on the transit nodes. Consequently, resource availability is not updated and re-advertised by the IGP. This may cause incorrect SR path computation and therefore can lead to overuse and consequently cause the congestion of a link or a path. Without updating the available resources, the distributed control plane is unable to guarantee the client QoS constraints. In the current stage of SR standards, only tactical traffic engineering is possible, where traffic can be sent over non-IGP shortest paths and without any prior admission

control or resource reservation. The path computation done by network nodes is constrained only by taking into account fixed additive metrics, *e.g.*, delay, but not concave metrics such as bandwidth.

In order to support traffic engineering (*e.g.*, bandwidth reservation) in an SR network (SR-TE), a centralized entity to keep track of the of the network resource availability must be introduced into the network. The most suitable solution is based on the Path Computation Element (PCE) architecture, as demonstrated in [42–45]. Initially, the PCE architecture was designed for path computation in MPLS-TE/GMPLS networks. However, with the proposed PCE communication Protocol (PCEP) extensions [4] and IGP extensions, the PCE can accomplish SR-TE path computation and instantiation the same way it did for TE tunnels.

The PCE acquires a global view of the service provider's network (nodes and links) with the associated traffic engineering metrics. This information is stored in the Traffic Engineering Database (TED). This requires a tight synchronization with the network distributed control plane for the PCE to be able to perform path computations based on an up-to-date network state. For that purpose, the PCE setups a BGP Link State (BGP-LS) session with (at least) one node that acts as a BGP-LS speaker in order to retrieve the underlying topology, traffic engineering metrics and SR information (SIDs, SRGBs, etc.) [9], as shown in Fig. 2.18. Another option is to establish an IGP adjacency with one of the network nodes.

The stateful PCE [46] is best suited to enable traffic engineering in an SR enabled networks. In this mode, in addition to the TED, the PCE maintains a LSP-TE/SR-TE path database (LSP-DB). The stateful PCE keeps both databases up to date and in sync with what actually exists on the physical network (instantiated paths and their QoS requirement, resource reservation, etc.), in order to correctly perform path computation and correctly update the available resources over the links used by SR-TE paths and MPLS-TE LSPs.

In service provider scenario, the PE routers act as Path Computation Clients (PCC). Once a PCC establishes a PCEP session with the PCE, the PCC can then initiate path computation requests for specific FECs. A request contains the following parameters: source and destination IP addresses, TE requirements (bandwidth, delay, jitter) that will be used by the path computation objective function. Once computed, using PCEP, the PCE sends to the PCC the list of SIDs that composes the path. Instead of signaling the path using RSVP-TE, the PE binds the new path to a specific FEC and then pushes the list of SIDs into the packet header that belongs to that FEC. Once reported back by the PCC, the PCE stores in its SR path data base

(LSP-DB) the newly instantiated path with its associated TE characteristics such as the amount of used bandwidth. This is needed in order to maintain the state of bandwidth reservation in the network and allows new path computations that take into account previous reservations.

Compared to RSVP-TE, SR does not signal a path and do not update the resources availability in the network. However, with the use of a PCE, all the resource reservations are maintained in its databases and the SR-TE path computation takes into consideration previous reservations. This kind of architecture is in the spirit of the SDN approach, where the PCE functionalities and the PCEP stack can be integrated directly into an SDN controller as demostrated in [45, 47].

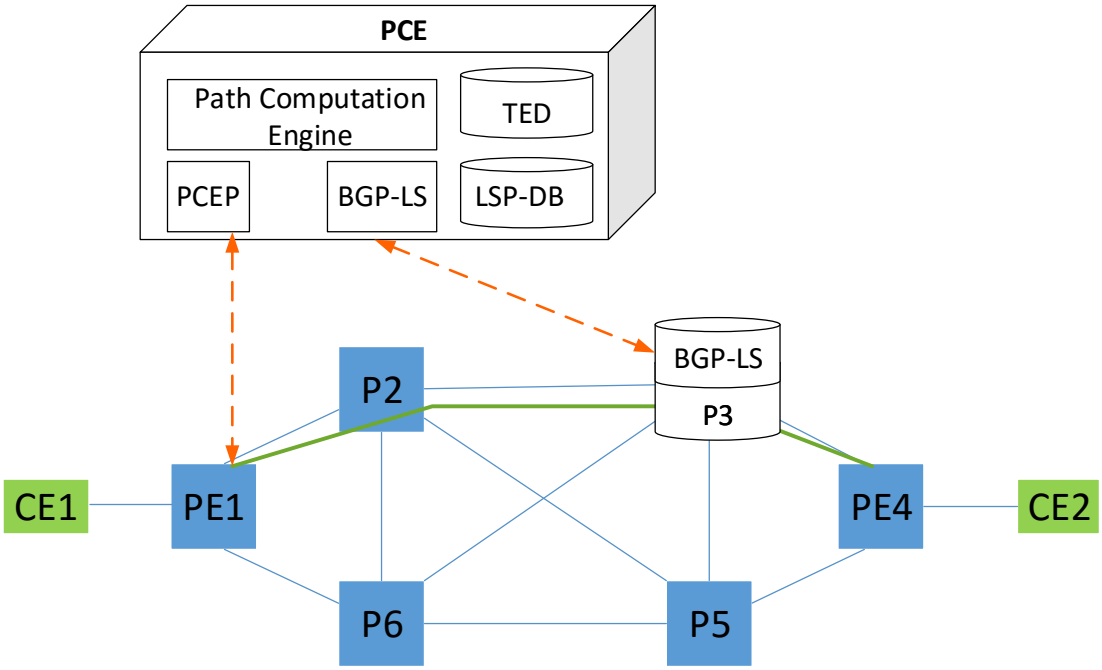


Figure 2.18: A Path Computation Element (PCE) is used to compute Segment Routing paths for Traffic Engineering. PCE protocol (PCEP) is used to send the label stack and BGP-LS protocol is used to collect topology information

2.5.4 Monitoring and Measurement

The major advantage of SR is the ability to express any topological path from any node in the network; also, it is possible to express paths that can pass through a node or a link multiple times or loops at a specific node back to the source. This mechanism can be used for data plane Operation and Maintenance (OAM), which is

a very important task for a network operator. However, SR OAM does not require control plane interaction as in MPLS OAM [48, 49] because the monitoring packets stay in the data plane. The requirements for SR OAM have been defined in [50] and the use case for a centralized monitoring station is detailed in [51].

Using SR, only one monitoring device (Path Monitoring System PMS) is needed to monitor the entire network. Using the reference network topology in Fig. 2.2, all the SPRING nodes reserve the same SRGB [1000, 2000], the results of the Node-SIDs computation as shown in Table I. A PMS monitoring device is connected to $PE1$, PMS is a SPRING node, and it has the Node-SID 1100.

For the example shown in Fig. 2.19, in order to monitor the IGP shortest path between $PE1$ and PE4, the PMS pushes onto the OAM packets two labels: [1004, 1100] then send them to $PE1$. When the packets arrive at $PE1$, it uses the top label 1004 to forward the packets down the IGP shortest path: $PE1 \rightarrow P2 \rightarrow P3 \rightarrow PE4$. Once at $PE4$, $PE4$'s Node-SID 1004 is popped and 1100 is used to determine the path back to the PMS. PE4 uses the IGP shortest path: $PE4 \rightarrow P3 \rightarrow P2 \rightarrow PE1$ to send back the packets to the PMS. Once back to the PMS, the Node-SID 1100 get popped. Finally, the PMS processes the OAM packets that have traveled the path to measure the Round-Trip Time (RTT) or to monitor the delay variation in order to detect routing changes.

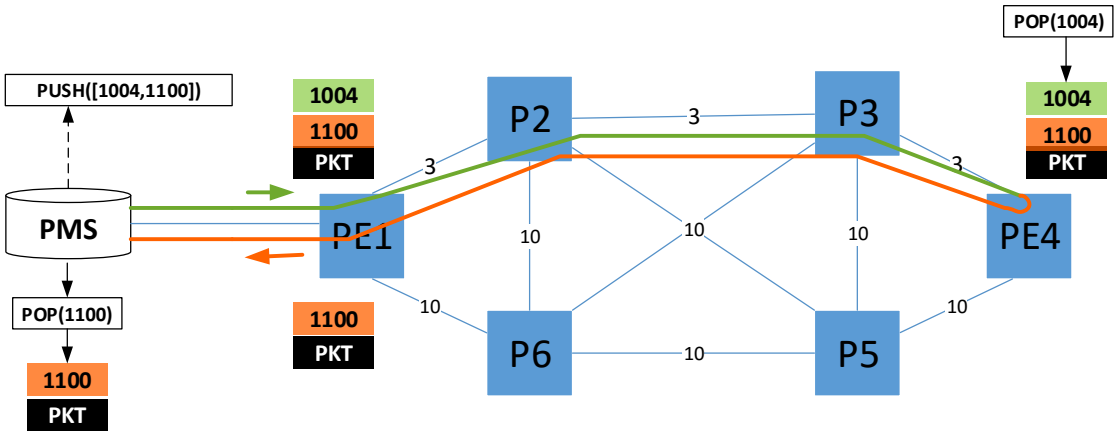


Figure 2.19: Network monitoring with Segment Routing. The SR path is composed of the round-trip stack

The main advantage compared to a classic probe deployment is that a service provider is only required to deploy a few number of probes; at least, only one would

be sufficient to monitor all links. By forging appropriate SID stacks, the probe could explore all potential paths in the network and perform monitoring and measurements.

2.6 Concluding remarks

In this chapter, we gave a state of the art of Segment Routing, which is a new architecture being standardized by the IETF. It can be instantiated currently over two widely deployed data planes: MPLS and IPv6. We have focused on its MPLS instantiation because of the significant interest expressed service providers such as Orange. Indeed, its lightweight control plane, small data plane forwarding states and easy integration with SDN controllers makes it a strong candidate to replace traditional MPLS networks. Additionally, SR leverages the source routing paradigm for packets forwarding, which brings much more flexibility to the network when combined with SDN centralized path computation and resource optimization.

SR standardization and deployment roadmap is advancing rapidly. In fact, Segment Routing is already deployed to enable some use cases such as explicit routing and failure restoration. However, there is still work to be done by standardization bodies and vendors before all the use cases are ready for deployment in production networks.

In this thesis, we focus on the traffic engineering use case (SR-TE). In fact, SR-TE needs to be consolidated in order to be presented as an alternative to RSVP-TE, especially when tight SLAs have to be respected. In the following chapters, we address the problems that face the deployment of SR-TE.

Chapter 3

Label Encoding Algorithm for MPLS Segment Routing

SR-MPLS is the central focus of the IETF working groups, mainly because of the important involvement of service providers (SPs) as they are an important factor for its wide adoption. Traffic engineering is one of the primary use cases being addressed. Several challenges were identified among it is the SR path encoding problem and how to bypass or limit the impact of the hardware limitation imposed by the Maximum SID Depth. In this chapter, we propose two encoding algorithms that reduce the impact of MSD and a reference implementation of an SDN based path encoding using our algorithm

In Segment Routing, packets are forwarded using the path that is encoded in their header. In the SR-MPLS the SID is represented as a 20-bit label. Consequently, It 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's header. In fact, pushing more than one label was supported since the early version of MPLS [1]. The label stack is used for multiple use cases: hierarchical tunnels [2], Layer 2 Virtual Private Network (L2VPN) [52], and Layer 3 VPN [3]. Those use cases require a relatively small label stack (two to three labels). For example, a scenario of L2VPN or L3VPN requires only simultaneously two labels: the tunnel's label and the VPN's label. However, a SR path requires a bigger label stack that can be composed from just one up to tens of labels depending on the network size. Consequently, as shown in Fig. 3.1, routers have to be able to push a larger number of labels in order to take full advantage of the SR potential.

Unfortunately, current hardware suffers from the physical limitation that constrains the number of labels that can be pushed simultaneously onto the packet's

header [4]. In the context of SR, this limitation is known as the Maximum SID Depth (MSD).

In order to achieve wire-speed packet processing, hardware vendors use Application-specific integrated circuits (ASIC), which are designed to perform specific operations very efficiently compared to general purpose processors. However, they are limited in the size and the type of the operations they can perform. The MSD limitation comes from the implementation of the PUSH operation in ASICs [5], which accept a maximum number of labels as input for the PUSH function. Therefore, efficient algorithms for SRP label encoding are essential to alleviate the MSD impact. A label encoding algorithm reduces the number of labels used to express a SRP.

In this chapter, we detail 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, their efficiency in alleviating the impact of the MSD limitation. Finally, we detail an SDN based implementation.

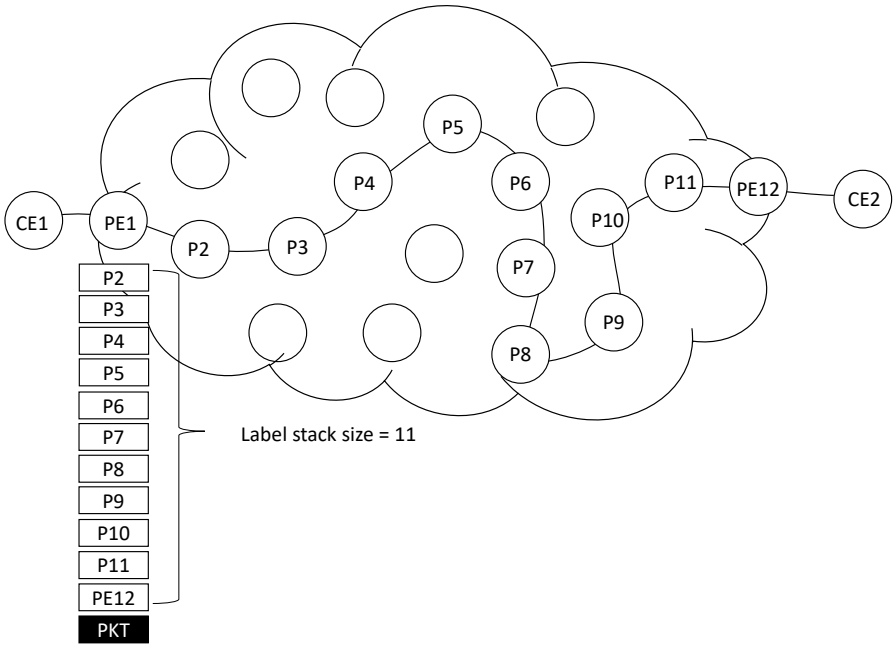


Figure 3.1: Reference network topology, all the links costs are 10 except the link P3-P7 its cost is 100.

3.1 Maximum SID Depth Signaling

The MSD corresponds to the maximum number of labels a router can push onto a packet header: it is a local characteristic of a router, it varies from one equipment vendor to another. This limitation is taken into consideration in the path computation process, because as it can render long paths that expressed with a label stack greater than the MSD unusable. Consequently, it forces the network traffic to follow only short paths which cause inefficient traffic distribution or worse network congestion. The MSD value is specific to the router's interfaces (line card). Consequently, a router may have different MSD values: one for each interface (line card). The MSD can be advertised in two ways:

- A single MSD per node which represent the lowest MSD of the node interfaces.
- Multiple MSD values are advertised for each node, one per inetrface.

The MSD can advertised into the distributed control plane using the IGP protocols extensions: OSPF [53] and IS-IS [54]. Additionally, In an architecture where the path computation is delegated by the SR nodes 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 [55] or via BGP-LS [56] SR extensions.

3.2 Related works

Overcoming the MSD limitation is essential to bootstrap the adoption of SR, this can be sensed as the first academic works on SR were the ones that address this problem. Because this limitation is linked to the hardware which is not easy to replace as it represents a considerable investment for service providers. The community focused on optimizing the label stack: the proposed encoding algorithms allowed to reduce the size of the label stack used to express SR path. In what follows we detail two of the first works on encoding SR paths:

In [33], Giorgetti *et al.* propose two SR path encoding algorithms: Segment Routing Direct (SR-D) and Segment Routing Reverse (SR-R). Both algorithms produce for the same SR path two label stacks of equal size. However, the $SR-D$ algorithm is better as it produces less packet end to end processing overhead compared to $SR-R$. We identified a problem with both algorithms that cause the resulting label stack to express a path different from the initial one. The proposed algorithms work well in a network where the shortest path between two directly connected nodes is a direct link.

However, because they use exclusively Node-SIDs we can have a case where the direct link to reach the next node that owns the Node-SID is not the shortest path leading to unwanted detour, such scenario happens when a network administrator chooses to attribute higher costs to particular links to achieve traffic engineering [57, 58].

In [34], Lazzer *et al.* propose a new approach to compute jointly the traffic engineering path and its label stack. First, a new graph is constructed based on the initial graph by adding a virtual link between every pair of physical nodes in addition to the existing links. The virtual links represent the Equal-Cost Multiple Paths (ECMP) between the two nodes. The proposed algorithm jointly compute the path and the minimum label stack. In the label encoding portion of the algorithm, the virtual link is replaced 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 than the initial network graph, with a number of links equal to the number of combinations of network nodes. For example, a network composed of a thousand 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.

Both works mentioned before suffer from limitations that make them unpractical to implement or lead to unexpected behavior. In the next section, we detail two algorithms SR path encoding algorithms and evaluate their performances.

3.3 Segment Routing Path Encoding

As mentioned before, a SR path can be encoded using a combination of SIDs (*i.e.*, local or global). The label stack provides a forwarding continuation along the SR path *i.e.*, each node the packet traverses has a forwarding instruction to reach the next node until the egress node. In this work, we focus on the expression of intradomain topological paths. Consequently, we only consider the use of two SIDs types: Node-SID and Adj-SID, other SID types such as service SIDs, BGP peering SIDs, etc. will be subject of future works.

The two proposed SR path 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 as detailled in 2.3.1.4.

The SR path length varies depending on the network diameter, QoS requirements, and network resources availability. Accordingly, the label stack to express a SR path

may exceed the ingress router's MSD rendering the SR path unuseable. A small MSD has other side effects such as preventing the use of additional label types like the entropy labels [39]. Therefore, an efficient encoding algorithm is required to minimize the size of the label stack.

3.3.1 Encoding types

A source routed path may be strict or loose as detailed in section 2.2. SR paths can be expressed exclusively using Node-SIDs, local Adj-SIDs, Global Adj-SIDs or a combination of those SID types. For the remainder of this manuscript, we consider a SR path as strict if it is encoded using only Adj-SIDs. Otherwise, it is considered loose as shown in Fig. 3.2:

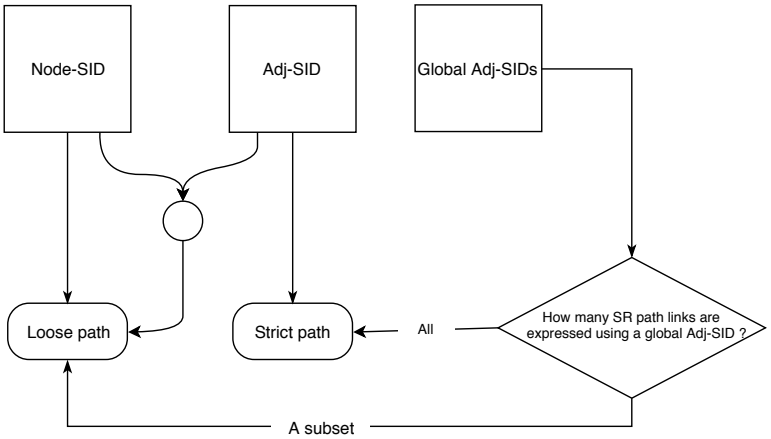


Figure 3.2: loose or strict path classification.

- A SR path encoded exclusively with Node-SID or a combination of Node-SIDs and Adj-SIDs is a loose path, because two successive Node-SIDs in the label stack can be separated by one or more network nodes. In this case 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 SR path encoded with local Adj-SID is a strict path, because the Adj-SIDs are local to the nodes advertising them and each one is associated to a to one link along the SR path. Therefore, the packet has to go through only the nodes that own the Adj-SIDs.

- A SR path 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.

3.3.2 Encoding algorithms

To reduce the impact of the MSD limitation, we propose two SR path path encoding algorithms that compute the minimum label stack to express a given topological path.

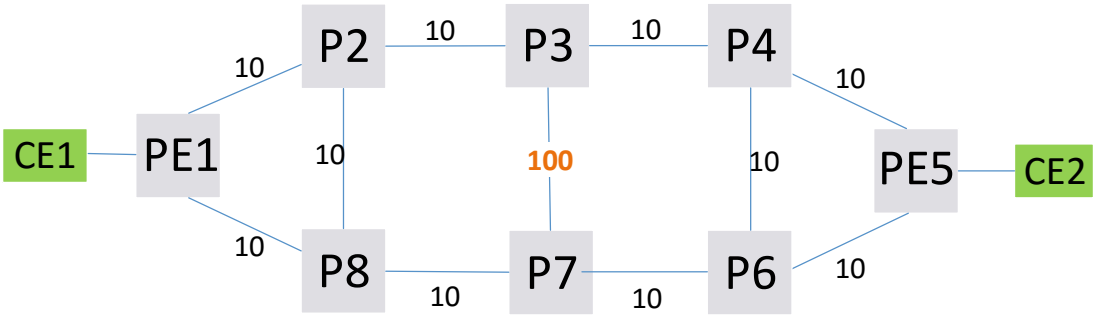


Figure 3.3: Reference network topology, all the links costs are 10 except the link P3-P7 its cost is 100.

Let us consider the topology detailed in Fig. 3.4. 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 CE1-CE2 flow packets. In what follows we detail SR path encoding algorithms.

3.3.2.1 Strict Encoding

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

- Using exclusively Node-SIDs to encode a SR path: each node in the SR path is replaced by its Node-SID. This approach suffers from the same problem as in [33]. In fact, the resulting label stack expresses the requested SR path 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 the path P: the packets at $P3$ will be sent to $P7$ over the first ECMP $P3 \rightarrow P4 \rightarrow P6 \rightarrow P7$ because link $P3 \rightarrow P7$ has a cost if 100. Therefore, it is not the shortest path.

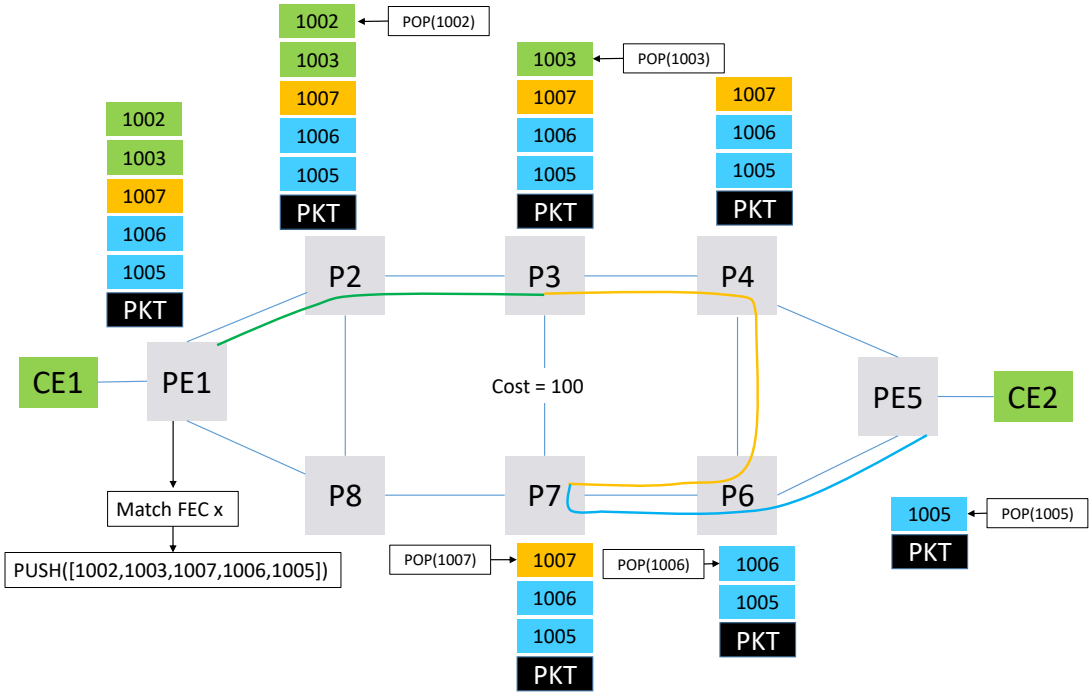


Figure 3.4: The problem that rises when expressing the SR path to connect CE1 and CE2 exclusively using Node-SIDs.

- Using exclusively Adj-SIDs to encode a SR path: 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. 3.5, a strict encoding of the 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 associated interface to that Adj-SID.

Strict encoding can be essential to accomplish certain tasks such as Operations, Administration, and Maintenance (OAM) [51]. For example, to monitor a specific path when ECMPs exist. The reference topology (shown in Fig. 3.4) is composed of 8 nodes. However, a service provider's 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 may violates the MSD constraint, also it adds a considerable overhead to packets.

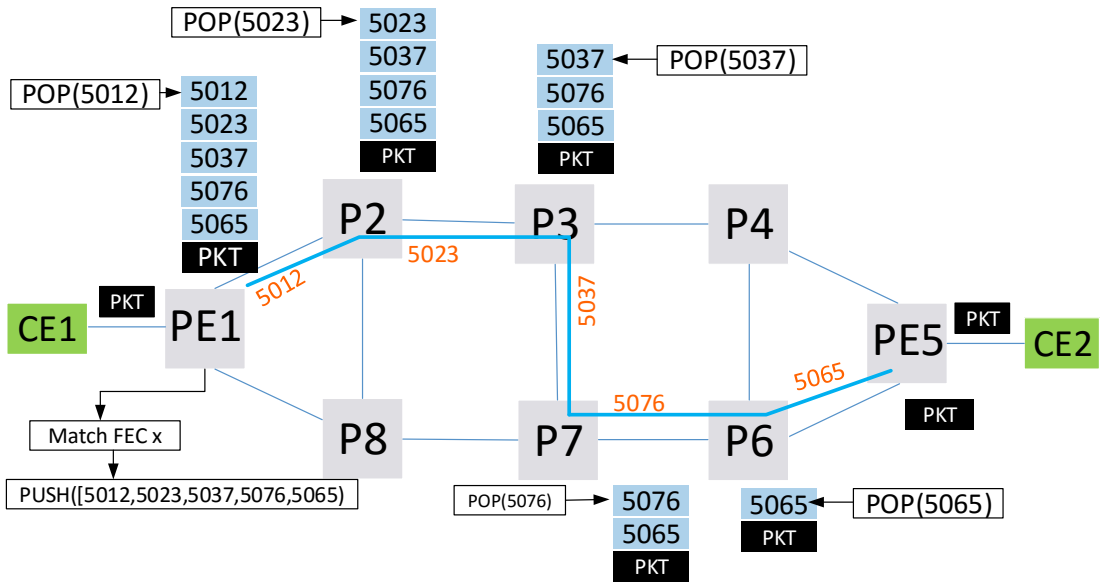


Figure 3.5: The SR path to connect CE1 and CE2 is expressed exclusively using Adj-SIDs by a strict encoding algorithm.

In the next two sections, we detail our two algorithms to reduce the impact of the MSD limitation: the SR-LEA that uses Node-SIDs and local Adj-SID to efficiently encode a SR path, and the SR-LEA-A algorithm which is an enhancement over the SR-LEA algorithm, it takes advantage of the possibility to advertise Adj-SIDs as global segments

3.3.2.2 SR-LEA Algorithm

We propose the SR-LEA algorithm to reduce the impact of the MSD limitation. The algorithm takes the initial path expressed as a list of IP addresses then computes the smallest sequence of SIDs able to represent exactly the same path. The initial path can be imposed manually or computed by a centralized entity such as a Software Defined Network (SDN) controller [44] [43] or by a Path Computation Element (PCE) [46] [45]. 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.

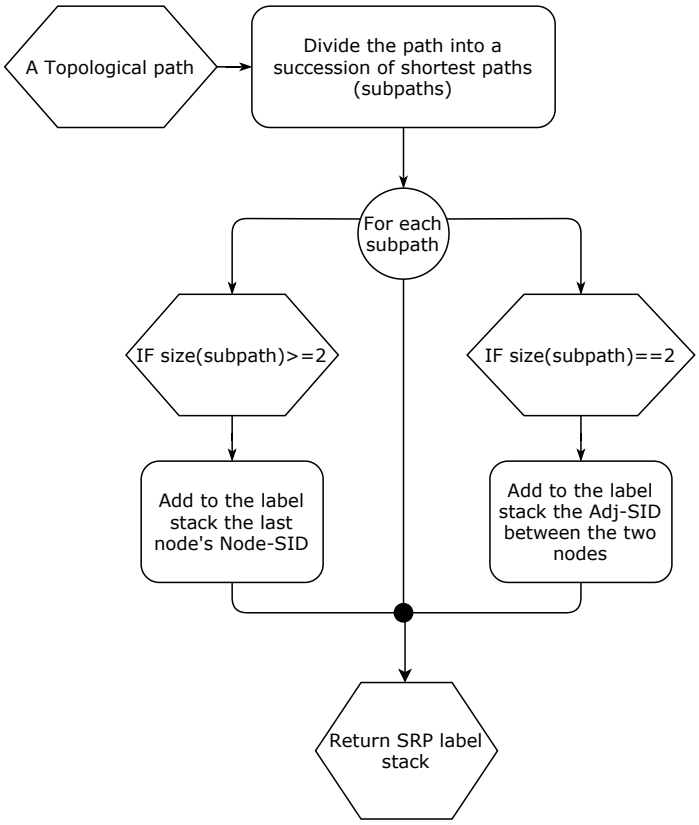


Figure 3.6: SR-LEA flowchart.

SR-LEA has two main steps as shown in Fig. 3.6 and detailed by the pseudocode in Algorithm 1 In the first step, the path is spliced to a succession of subpaths. The number of subpaths represents the size of the final label stack. In the second step, we replace each subpath by a single SID. The order of SIDs is important when added to final label stack in order to respect the initial path:

- In the first step, the SR path is spliced into a succession of shortest paths (subpaths) using the Dijkstra algorithm: container A holds the final SR path splices, whereas the container B, will hold the current subpath that is being computed when finished it is moved to the container A.
- In the second step, 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 SR path follows the shortest path. Consequently, container A will hold one splice equals to the initial path. Then step two of the algorithm will output a label stack composed of one label: the egress node's Node-SID.

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 subpaths.
$B = [\ ]$: A temporary variable used to construct a single 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.

$i = 1$: Points to the current node of the SRP.
$k = length(SRP)$: Points to the last node of the candidate subpath.
while $i <= length(SRP)$ **do**
  $push(B,\ SRP[i])$
  if $i == length(SRP)$ **then**
    $push(A,\ B)$
  else if $B \nsubseteq SPF$ **then**
    if $length(B) == 2$ **then**
      if $k > i$ **then**
        $k = k - 1$
        $B = B[1]$
        $SPF = Dijkstra(G,\ B[1],\ SRP[k])$
        $continue$ $\Longrightarrow$ jumps to the beginning of the loop for next iteration
      else
        $push(A,\ B)$
        $B = B[end]$
        $SPF = Djikstra(G,\ B[1],\ SRP[k])$
      end if
    else
      $push(A,\ B[1 : end - 1])$
      $SPF = Djikstra(G,\ B[end - 1],\ SRP[k])$
      $B = [\ ]$
      $i = i - 1$
      $continue$
    end if
  end if
  $i = i + 1$
  $k = length(SRP)$
end while

STEP 2: The construction of the label stack.**for** $i \leftarrow 1$ To $Size(A)$ **do**
if $length(A[i]) > 2$ **then**
$push(labelStack,\ NodeSID(A[i][end]))$
else
$push(labelStack,\ AdjSID(A[i]))$
end if
end for

To compute the minimum label sack to encode the SR path P, we follow the two steps of the SR-LEA algorithm. First, the splices that compose the path P are computed and saved in A: $\{(PE1, P2, P3), (P3, P7), (P7, P6, PE5))$. Then, each subpath in A is replaced with the appropriate SID which results in the following label stack: [1003, 5037, 1005]. The details on how we convert the subpaths in A to the final the label stack are as follow:

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

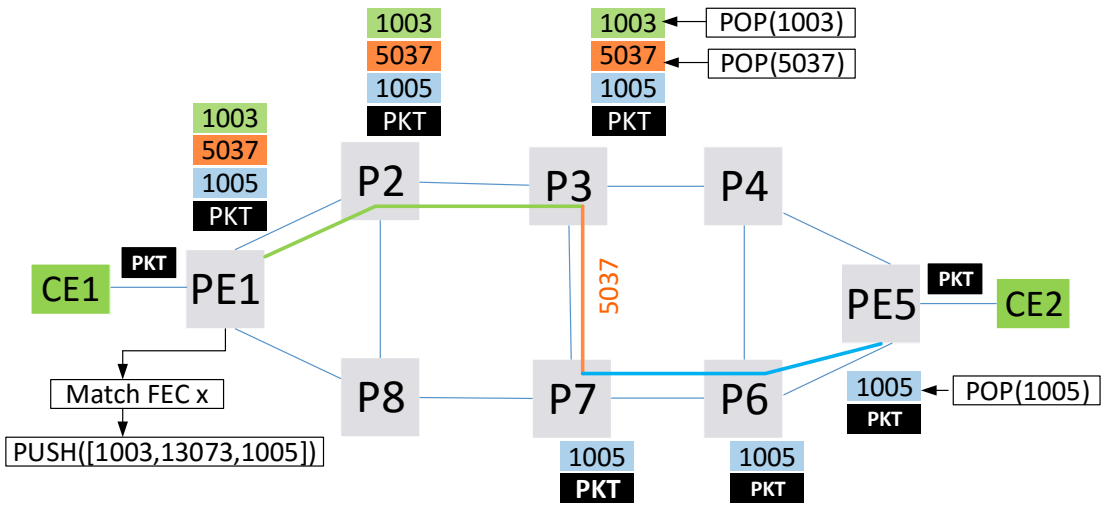


Figure 3.7: The SR path to connect CE1 and CE2 is expressed as a label stack computed using the SR-LEA algorithm.

As shown in Fig. 3.7, 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 $CE2$.

3.3.2.3 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 all or a subset of Adj-SIDs are advertised as global segments, the resulting label may be composed of Node-SIDs, local Adj-SIDs, and global Adj-SIDs. The size of the 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 larger than 3 followed by one of size equal to 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.

In the example described in Fig. 3.8, 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, 1005]. 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 $PE5$.

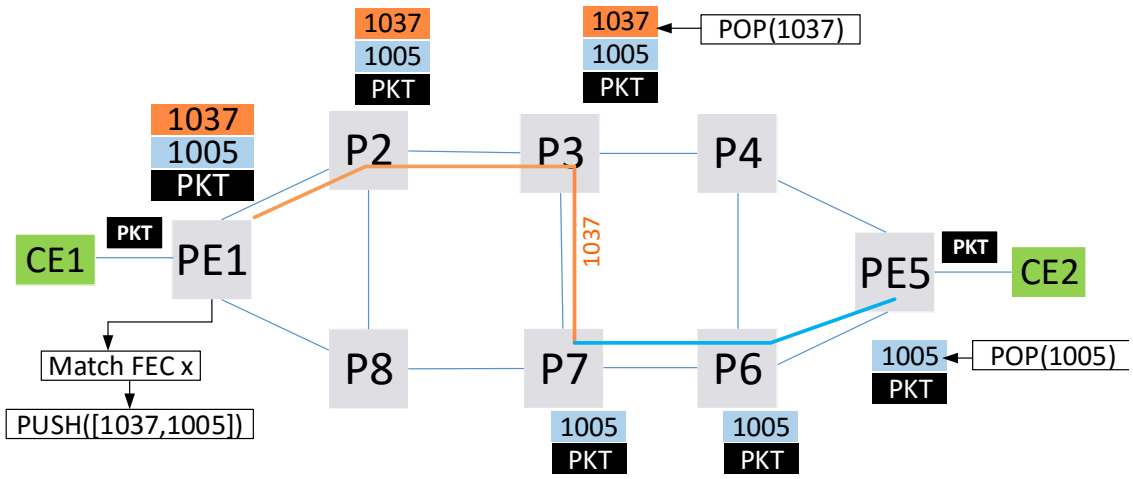


Figure 3.8: The SR path to connect CE1 and CE2 is expressed as a label stack computed using the SR-LEA-A algorithm.

Algorithm 2 Efficient Label Encoding algorithm with global Adj-SIDs

STEP 1 Same as for SR-LEA

STEP 2

1: **for** $i \leftarrow 1$ To $Size(A)$ **do**
2: **if** $length(A[i]) > 2$ **then**
3: **if** $length(A[i+1]) == 2$ & & $exist(GlobalAdjSID(A[i][end],\ A[i+1][1]))$ **then**
4: $push(labelStack, GlobalAdjSID(A[i][end],\ A[i+1][1]))$
5: $i = i + 2$
6: *continue* $\Longrightarrow$ jumps to the beginning of the loop for next iteration
7: **end if**
8: $push(labelStack, NodeSID(A[i][end]))$
9: **else**
10: $push(labelStack, AdjSID(A[i]))$
11: **end if**
12: **end for**

3.4 Simulation Results

In order to evaluate the performance of the proposed algorithms, we experimented on several network typologies available in SNDlib library [6] [59]. To get a representative set of paths. First, for each topology, we consider a sample bandwidth demand matrix D based on detailed measurements of traffic in real IP networks, the values represent the node-to-node demand trace. Second, We solve a multicommodity flow problem [60] to identify the optimal set of paths to satisfy a demand matrix on several

topologies characterized in Table 3.1. Third, The paths are then encoded using the strict Adj-SID, SR-LEA and SR-LEA-A algorithms. Equipment vendors such as Cisco [61], Juniper, Nokia and Huawei announce different MSD values for different router series. For this study, we fixed the MSD to 5 labels, which is the most common values that we probe inside Orange network.

Topologies	Number of vertices $\|V\|$	Number of edges $\|E\|$	Number of demads $\|D\|$
Geant	22	36	431
Albilene	12	18	131
Brain	161	166	9045
Germany50	50	80	1270
Nobel-germany	17	26	248

Table 3.1: characteristics of the topologies used in the simulations.

The two proposed algorithms, compute the minimum label stack to express the SR path. Recall that SR-LEA is used when the Adj-SIDs are local segments whilst SR-LEA-A is used where there are global Adj-SIDs. 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, we compute the average label stack size and the percentage of network paths from solving the multicommodity problem encoded with a label stack size lower than the MSD.

Fig. 3.9 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%. When compared to the SR-LEA we see a slight improvement in the label stack size which allows for additional useable paths. The amount of additional path that can be used when the SR-LEA-A is used

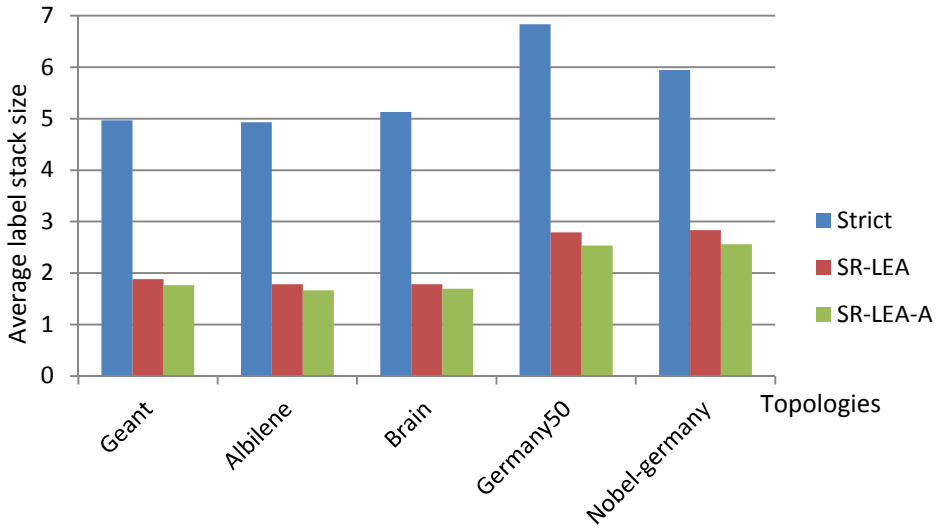


Figure 3.9: Comparison of the average label stack size generated using a strict encoding, SR-LEA and SR-LEA-A algorithms.

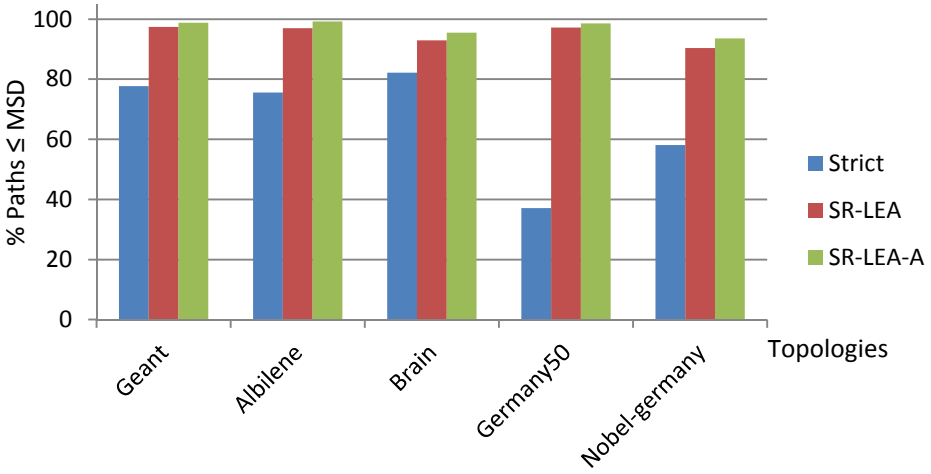


Figure 3.10: Paths expressed with a label stack size lower that the MSD ($MSD = 5$).

Fig. 3.10, 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 *Germany*50 topology. Using SR-LEA, increases considerably the amount of useable paths *e.g.*, from 37% to 97% for *Germany*50 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.

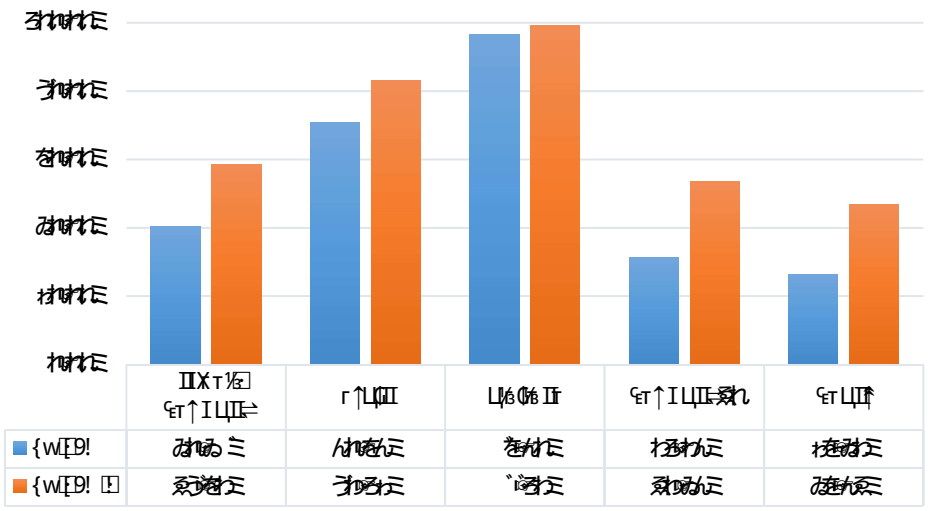


Figure 3.11: Comparison of SR-LEA and SR-LEA-A over a large set of paths.

We notice a slight improvement in the number of paths that can be encoded with a label stack size less than the MSD using SR-LEA-A compared to the SR-LEA, the set of paths that we use in this comparison are resulted from solving the multicommodity flow problem: one path per demand. However, the number of paths is limited by the demand matrix size. Therefore, the set of paths is not sufficient to compare the two algorithms. To showcase the advantage of SR-LEA-A we consider for each topology a large number of paths as detailed in Table 3.2: we compute up to five hundred paths between every two nodes in the network graph using the Yen's k-shortest path algorithm (*i.e.*, $K == 500$). Then encode each path using both algorithms. with this amount of paths, we can see clearly as shown in Fig. 3.11, that SR-LEA-A provides a considerable improvement compared to SR-LEA: from 20% to 28% on all topologies besides Albilene. which has small graph composed of only twelve vertices and eighteen edges. Therefore, there are not enough paths to explore.

Topology	Total number of paths
Geant	198146
Albilene	1031
Brain	161931
Germany50	635000
Nobel-germany	23674

Table 3.2: Number of paths computed using Yen's K-shortest path algorithm with $k == 500$

We conclude that the proposed algorithms are very efficient in reducing the label stack size, also to minimize considerably the impact of the MSD limitation. However, both algorithms do not completely eliminate the MSD problem, as we still have paths that are expressed with a label stack greater than the MSD. In chapter 4.6, we introduce a new segment type called Target SID and an encoding algorithm that uses this new segment to solve the MSD limitation.

3.5 SR-LEA SDN based Implementation

SR-MPLS couples MPLS's robust data plane with SR light distribution control. The SR control plane simplicity comes from extending already deployed protocols such as OSPF, ISIS, and BGP-LS. Additionally, a SR path is carried in the packet's header as a label stack; this minimizes considerably the number of states core routers have to maintain. Therefore, no signaling protocols such as RSVP-TE or LDP are required. Unfortunately, losing the signaling process means that the resources availability information is not updated hence not advertised in the network. Consequently, SR benefits from an SDN based architecture, where the controller maintains a global SR traffic engineering database to track all SR path computation requests and update the resources availability accordingly.

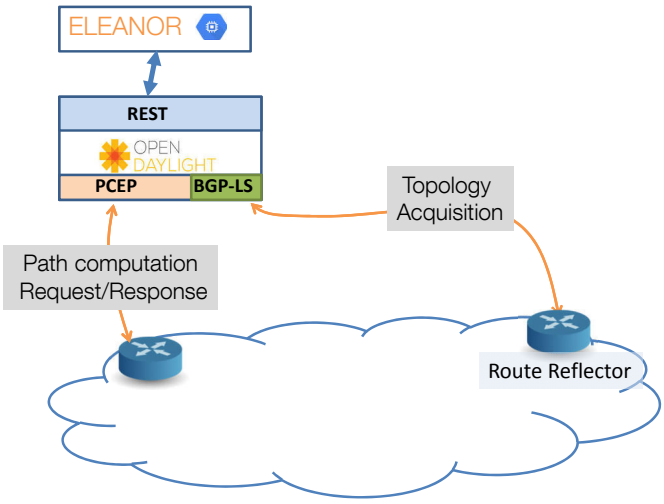


Figure 3.12: ELEANOR Reference Architecture

In this section, we detail ELEANOR the application that we developed as a northbound application for the OpenDayLight SDN controller. ELEANOR is the first SDN application that offers SDN based label stack encoding. ELEANOR is tested on real topologies. ELEANOR resides outside of the controller and communicates with OpenDayLight using its northbound REST API [62]. Specifically, it provides a SR path computation and management module to enable traffic engineering capabilities. ELEANOR mitigates also the impact of the MSD limitation through its label stack optimization module, that minimizes the size of label stack required to express SR paths.

In what follows, we detail ELEANOR's software architecture and its two main modules: Path computation and label stack optimization modules.

3.5.1 ELEANOR architecture

ELEANOR is an application developed for SR paths computation, management and label stack optimization. Computes, encodes, stores and track all the active SR paths with their QoS requirements. With the available information, new heuristics can be developed for global resource optimization. For example, an administrator can schedule periodic SR paths placement optimization to better distribute SR path in the network in order to increase the acceptance rate of future demands.

ELEANOR is based on the open source project Pathman-SR [63], and it communicates with OpenDayLight through its northbound REST API. As depicted in Fig. 3.13, ELEANOR's software architecture is composed of two main modules: the path

computation module and the label stack optimization module. The OpenDaylight SDN controller uses two of its southbound interfaces to communicate with the network: BGP-LS for topology acquisition and PCEP to push SR paths configuration onto the routers.

3.5.1.1 Path Computation Module

The path computation module host a suite of path computation algorithms and the Traffic Engineering Database (TED). The path computation requests are first handled by this module. The appropriate path computation algorithm is chosen based on the request parameters: path disjointness or QoS (*e.g.*, delay, bandwidth, path protection). The resulting path is then passed to the label stack optimization module.

3.5.1.2 Label Stack Optimization Module

In order to compute the minimum label stack to express SR paths and therefore reduce the impact of the MSD, we have implemented the Segment Routing Label Encoding Algorithm (SR-LEA) detailed in 3.3.2.2. As we can see in the Fig. 3.6, after the path computation module has computed the optimal path for given request. The path is then passed to the label stack optimization module. SR-LEA relies on the SID database to compute the minimum label stack. We did only implement SR-LEA and not SR-LEA-A due to lack of support by industrial routers for global Adj-SID.

For example, a client requests a path between *Amiens* and *Toulouse* with 100 MB of bandwidth. First, the appropriate CSPF is called to compute the path. Thus, the resulting best path is {*Amiens, Paris, Orleans, Lyon, Marseille, Toulouse*}. Second, the path is passed to SR-LEA algorithm, the SR path is spliced into three parts: {*Amiens, Paris, Orleans*}, {*Orleans, Lyon*} and {*Lyon, Marseille, Toulouse*}. Then as depicted in Fig. 3.6, the first subpath is encoded with the Node-SID of *Orleans*, the second subpath is encoded by the Adj-SID attributed by *Orleans* to its adjacency to *Lyon*, and the third subpath is encoded with the Node-SID of *Toulouse*. The resulting label stack is a combination of Node-SIDs and Adj-SIDs encoded in XML format, which get pushed into ODL using the POST method.

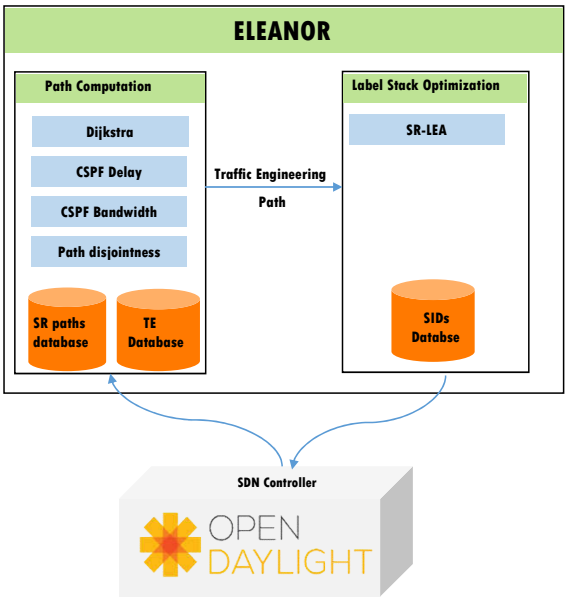


Figure 3.13: ELEANOR software architecture

3.5.2 Testbed Network Topology

To demonstrate ELEANOR, we built the testbed topology depicted in Fig.3.14, it is composed of several routers from Juniper, Cisco and FRRouting -SR our open source implementation of SR-MPLS based on the FRRouting routing software [64]. The routers are mapped over the France map, each router is named based on the city it is located in.

We use the OSPF protocol with SR extension enabled [65], network routers use OSPF-SR to exchange SR information such as Node-SID, Adj-SID, and SRGB, etc. The border of the network is composed of a mix of industry routers, for the transit nodes we use the FRRouting-SR routers. The border routers run the Path Computation Client (PCC) application, it is used to request a path computation to be performed by the SDN controller: each PCC establishes a PCEP [4] session with OpenDayLight's southbound interface PCEP. Additionally, OpenDayLight uses the PCEP session for the creation and deletion of the SR paths requested by ELEANOR. OpenDayLight learns the link state topology (*e.g.*, the network graph the traffic engineering information) by establishing a BGP-LS session with a single router in our case *Rennes* that plays the role of a Route Reflector (RR) [66]. The route reflector advertises using the BGP-LS session to OpenDayLight the network information that it sources from the interior gateway protocols. This information is copied by ELEANOR in order to perform its computations.

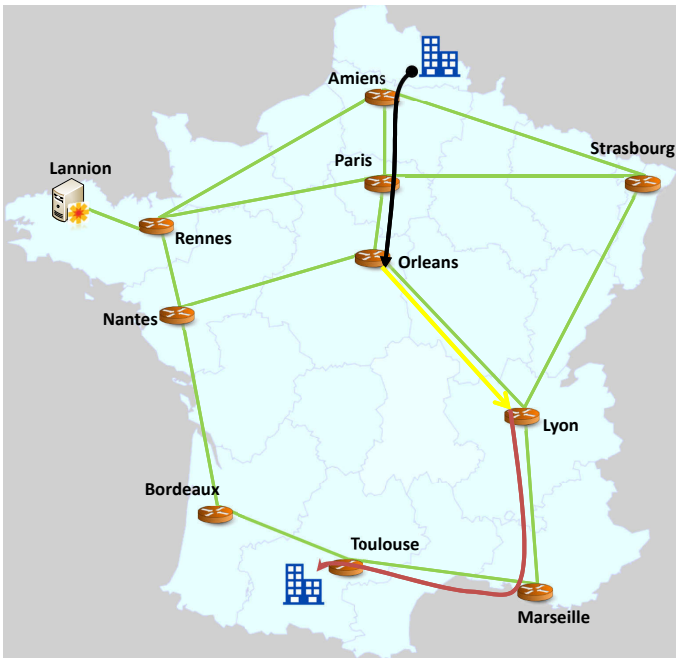


Figure 3.14: Testbed Topology

3.5.3 FRRouting-SR

FRRouting is a routing software suite [67] that runs on Unix platforms. It provides an open source implementation of routing protocols such as OSPFv2, OSPFv3, ISIS, and BGP. Consequently, it can run as a standalone router on a commodity hardware(white box). Its architecture is composed of mainly two modules: the core daemon where protocol instances run and the Zebra module that ensures the communication between the different routing daemons and the Linux kernel.

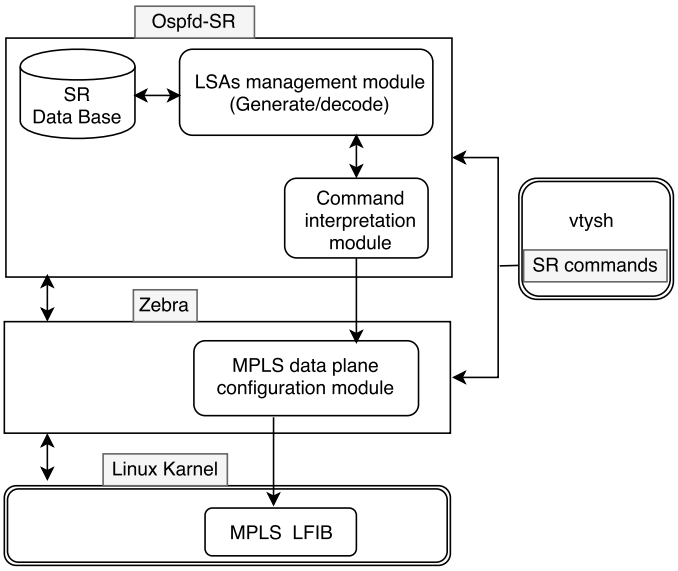


Figure 3.15: FRRouting-SR: Open source implementation of SR-MPLS

We have extended the FRRouting suite to deliver SR functionalities. This implementation requires a Linux Kernel 4.5. The detail of our implementation is depicted Fig. 3.15. Several modules and extensions had been developed in order to add the support of SR. Notably, the OSPF Daemon (OSPFD) is extended to support the encoding and decoding of the SR TLVs. SR database maintains the SR information locally configured (*e.g.*, SRGB, Adj-SID and Node-SID) or learned via the neighbors. Several SR specific command has been added to vtysh shell: to enable SR, SRGB configuration, Node-SID configuration, etc.

This implementation allows for the quick adoption of new SR standardization proposal. Additionally, it can be installed on white boxes to deliver SR functionalities. To ensure the proper functioning of FRRouting -SR router, interoperability tests with routers from different vendors has been successfully performed.

3.6 CONCLUSION

In this chapter, we detailed 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 performances have been evaluated over real topologies which demonstrated their efficiency in alleviating the impact of the MSD. The proposed algorithms are essential for the wide adoption of SR as they give an easy software solution for the MSD limitation. Additionally, we have detailed ELEANOR an OpenDayLight northbound application for segment routing path computation,

management, and label stack encoding optimization based on our SR-LEA algorithm. This application can be used by service providers to deliver traffic engineering over segment routing. We tested ELEANOR over a network topology composed of routers from different vendors in addition to FRRouting-SR router: our implementation of SR-MPLS.

In the next chapter, we introduce a new segment type that we couple with a new algorithm to encode SR paths. We evaluate its performance in solving the MSD problem.

Chapter 4

A New Method For Encoding MPLS Segment Routing TE Paths

4.1 Introduction

In the previous chapter, we proposed algorithms to reduce the label stack required to encode a SR path. Even though, the results were good we still didn't reach a 100% network coverage. For that purpose in this chapter, we explore path fragmentation to tackle the MSD limitation. We define the Targeted SID (TSID), a new segment type attached/assigned to a slice of the SR path. TSID's role is to reduce the size of the label stack to express a SR path. 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 SR path. In this chapter, we prove that SR paths 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.

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

4.2 Related Work

In the literature, several algorithms have been proposed to efficiently encode SRPs [68] [34] [33]. Their focus is to minimize the number of labels used to encode a SR path, 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 SR path 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 SR path as loose makes the SR path 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 SR paths, 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 SR path label stack while maintaining the expression of the path as strict. In this approach, we use the proposed TSID mechanism to substitute a subset of the path labels. A binding between the TSID and the labels it substitutes is installed into a network node. When the packet reaches that specific node the TSID get replaced with the labels bound to it. To the best of our knowledge, this work is the first to use the path segmentation approach as a solution to the MSD limitation. This is a standalone approach yet it can be combined with an efficient encoding algorithm such as those previously discussed.

4.3 Path Segmentation

Traffic Engineering, QoS requirements enforcement and path diversity are use cases that require the computation and the enforcement of paths that are usually not preferred by the IGP. However, the corresponding label stack to implement such paths in SR may be greater than what is allowed by the ingress node's MSD. Consequently,

in addition to the encoding algorithms proposed in 3.6, We propose the path segmentation approach, where the initial label stack to express a SR path 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.

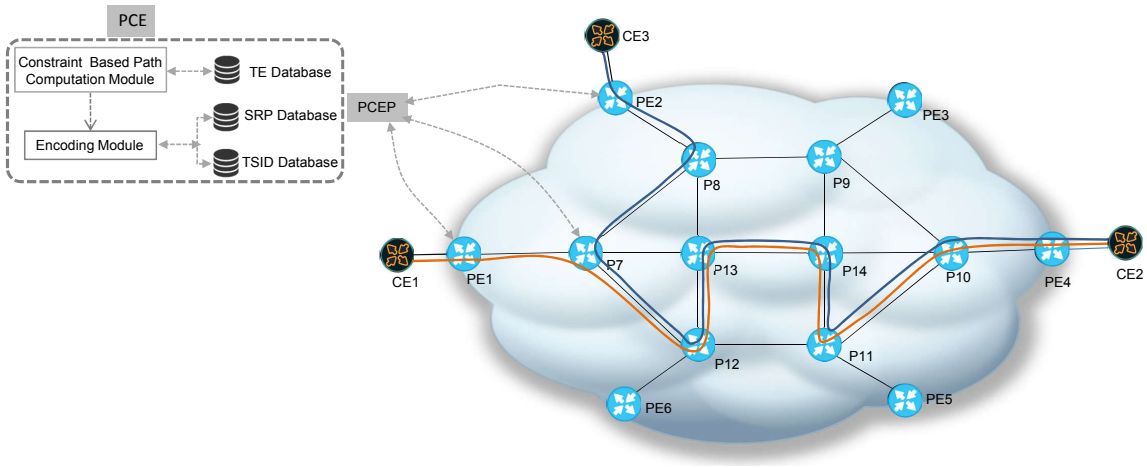


Figure 4.1: TSID Design Architecture.

For illustration purposes, let us consider the network depicted in Fig. 4.1. 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 SR path 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. 4.1, 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 4.1: P7's LFIB

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

4.3.1 Targeted SID Architecture

As stated previously SR control plan relies on a centralized server such as a PCE or an SDN controller to deliver traffic engineering and QoS. We extend such architecture in order to leverage TSIDs to reduce the impact of the MSD. Several architectural components need to be combined 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 SR path. Consequently, the PCE has to maintain a TSID database in order to be able to reuse previously installed TSID for future SR paths. In this proposed architecture, all network nodes implement the Path Computation Element Clients (PCC). When a request reaches 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 4.1, 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, [69] defines new Type Length Values (TLV) for SR. The SR Explicit Route Object (SR-ERO) carry the label stack to express a SR path. We propose to reuse the same mechanism and TLVs to install a TSID and the label stack it substitutes. Because the installation of TSIDs has to be initiated by the PCE, we propose to extend the mechanism described in [69] 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 makes sense 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. Accordingly, the reporting mechanism is currently defined in [70] to let the PCC reports to the PCE the label value it has associated/bind to a TSID.

In addition, it might be of interest to service providers to advertise the TSIDs in the network using the IGP. This can be done via simple IGP protocols extension. In this scenario, the PCE may not be the only entity responsible for SR paths 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 the case, of a 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.

In the next section, we formalize the problem statement for TSIDs placement. Additionally, we propose two linear programming models for offline TSID placement optimization, then we follow with two online optimization algorithms, we finish off by assessing the performance of the online algorithms against the offline ones.

4.4 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

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

4.4.1 Offline Optimization of TSIDs Placement

The first offline LP model (4.1) 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 because it is the most common value in production 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 TSIDs associated with the same label stack because they are installed on different nodes. This important for our model as we need to distinguish TSIDs not only by their label stack but also by the node they are installed on. For example, in Fig. 4.1, the label stack composed of three Node-SIDs *[P13,P14,P11]* can be installed on different nodes: P7,P8 or 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. α_{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.

$$\textbf{Minimize} \sum_{t \in \mathcal{T}} \hat{f}_t \tag{4.1a}$$

Subject to :

$$\hat{f}_t \geq f_{pt} \qquad \forall p \in \mathcal{P}, \forall t \in \mathcal{T} \tag{4.1b}$$

$$s_p - \sum_{t \in \mathcal{T}_p} f_{pt} * (s_t - 1) \leq MSD \quad \forall p \in \mathcal{P} \tag{4.1c}$$

$$\sum_{t \in \mathcal{T}_p} f_{pt} * \alpha_{lt} \leq 1 \qquad \forall p \in \mathcal{P}, \forall l \in p \tag{4.1d}$$

The objective function (4.1a) minimizes the sum of $\hat{f}_t$ (*i.e.*, the total number of used TSIDs). The Equation (4.1b) ensures that a TSIDs is computed once even it is used to reduce multiple paths. Equation (4.1c) 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 SR path 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 (4.1d) 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.

4.4.2 Offline Minimization of PCEP sessions

The TSID architecture as depicted in Fig, 4.1 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, service providers tend 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 service provider 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 paths will be affected especially that no protection mechanism is defined for the TSID approach.

$$\textbf{Minimize} \sum_{n \in \mathcal{V}} k_n \tag{4.2a}$$

Subject to:

$$\hat{f}_t \geq f_{pt} \qquad \forall p \in \mathcal{P}, \forall t \in \mathcal{T} \tag{4.2b}$$

$$s_p - \sum_{t \in \mathcal{T}_p} f_{pt} * (s_t - 1) \leq MSD \quad \forall p \in \mathcal{P} \tag{4.2c}$$

$$\sum_{t \in \mathcal{T}_p} f_{pt} * \alpha_{lt} \leq 1 \qquad \forall p \in \mathcal{P}, \forall l \in p \tag{4.2d}$$

$$k_n \geq \hat{f}_t * \zeta_{n,t} \qquad \forall p \in \mathcal{P}, \forall t \in \mathcal{T} \tag{4.2e}$$

The offline LP model (4.2) 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 (4.2a) minimizes the network nodes that have to be a PCC. In (4.2a) 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 (4.2e), the LP model (4.2) is subject to the same constraints as the LP model (**??**). $\zeta_{n,t}$ 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.

In the next section, we detail online optimization algorithms for PCEP and TSIDs. We compare their performances to the offline LPs detailed in this section.

4.4.3 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 service provider does not have the full demand matrix. Therefore, the connection demands are treated by the controller one by one, where each demand contains 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 variations of an online optimization algorithm, referred to 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 TSIDs available 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 3. 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 on the MSD value and the longer of the SRP. The TSIDs that constitute a solution must not intersect or otherwise we a forwarding is created. 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 the 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, the 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. 4.1. The encoding engine triggers the installation of SRP and TSIDs, also maintains the TSID Database.

Algorithm 3 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
```

4.5 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 Section 3.3.2.2. The experiments use network topologies provided by SNDlib [6] [59] and their demand matrices. We fixed the MSD to 5 labels, which is the value announced currently by the major equipment vendors. Table. 4.2, details for each topology, the number of paths to encode and the number of possible TSIDs.

4.5.1 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 a 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.

Table 4.2: 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

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 solution that reduces the SRP's label sack is chosen. As a result, new TSIDs are created more frequently. As seen in Fig. 4.2, 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 4.2. 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.

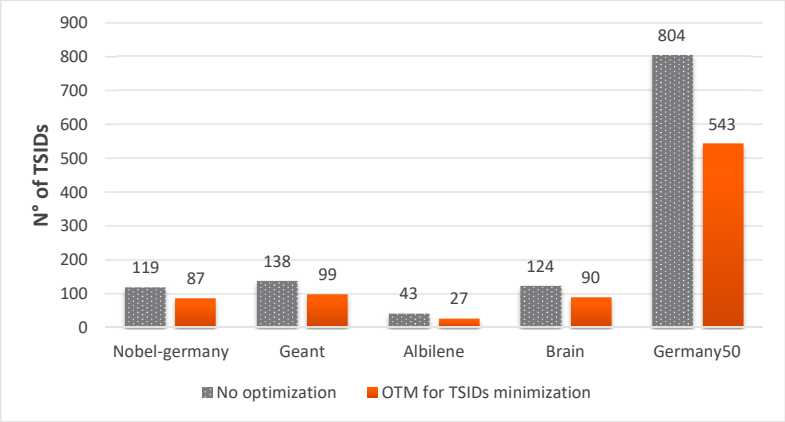


Figure 4.2: OTO for TSIDs minimization compared to the worst-case scenario *i.e.*, online TSID installation with no optimization.

The LP model (**??**), computes the minimum number of TSIDs required for a given path set. It is used as a benchmark for the OTO algorithm. The close OTO results are to LP the better. As it can be seen in Fig. 4.3, The OTO algorithm performs very well especially for small path sets. The number of TSIDs installed 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.

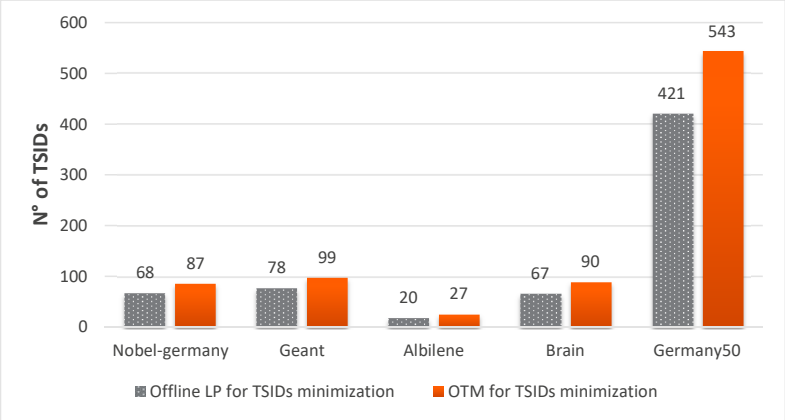


Figure 4.3: OTO for TSIDs minimization compared to the Offline LP for TSIDs minimization

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. 4.4, 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.

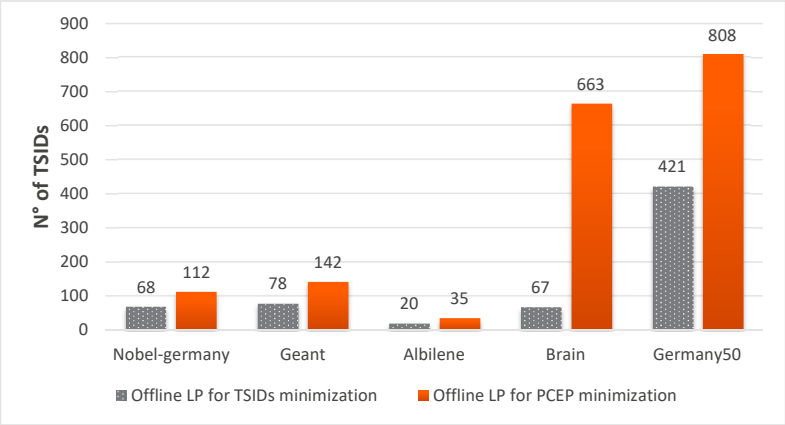
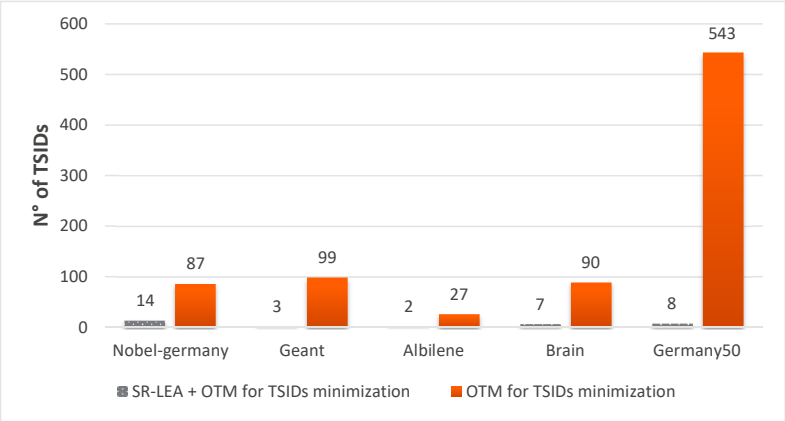


Figure 4.4: Comparison of the number of TSIDs created by the two offline LPs

The PCE's encoding engine can rely solely on the OTO algorithm to reduce the size of all the label stacks. Furthermore, SR-LEA encoding algorithm presented in Section 3.13 can be introduced as an intermediate step. The OTO algorithm is called only if the SR-LEA algorithm fails at computing a label stack with a size less than the MSD. In this scenario, the PCE first requests the SR-LEA algorithm to reduce the size of the label stack. If the resulted stack if is still bigger than the MSD, then it is passed to the OTO algorithm. Otherwise, it is the final stack. Consequently, we observe a drastic decrease in the number TSIDs as it can be seen in Fig. 4.5.

4.5.2 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 way service providers are using the PCE and RSVP-TE is to establish



3

Figure 4.5: Comparison of the number of TSIDs created solely by OTO and SR-LEA encoding algorithm combined with OTO

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 service providers especially for large networks.

In an online scenario, connection demands arrive sequentially to the PCE. Therefore, anticipation the establishment of PCEP session with a subset of the network nodes is not possible. In the path segmentation architecture, we trigger the establishment of new PCEP sessions only when required. For each SRP, we generate a set of solutions composed of TSIDs to reduce the stack size. We use a weight function to penalize solutions that require 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.

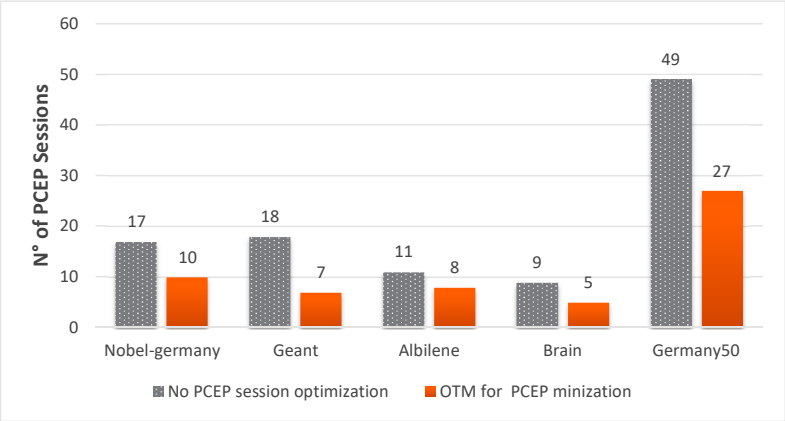


Figure 4.6: OTO for PCEP sessions minimization compared to the worst-case scenario of an online TSIDs installation with no PCEP optimization.

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. 4.6, the weight function of the OTO algorithm allows to reduce the number of PCEP sessions.

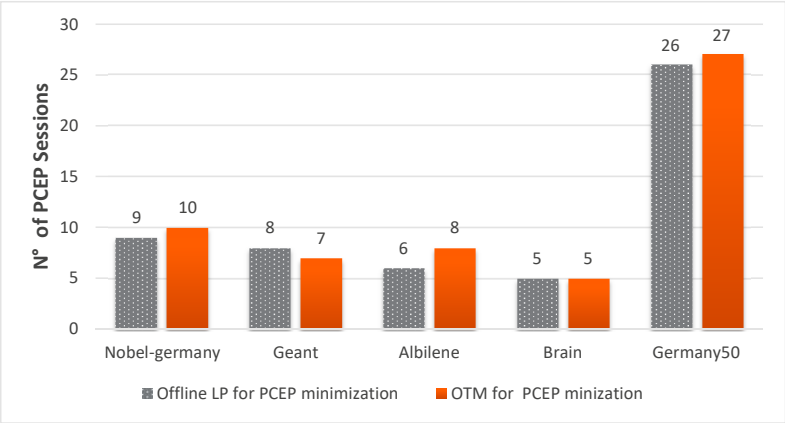


Figure 4.7: OTO for PCEP sessions minimization compared to the offline LP for PCEP sessions minimization

The offline LP model for PCEP session minimization (4.2), serve as a reference to evaluate the performance of the OTO algorithm. As it can be seen in Fig. 4.7, on all the tested topologies, the gap between the offline LP and OTO is very small. Hence, we conclude that OTO performs very well.

Minimizing the number of TSIDs comes at a price of augmenting the number of PCEP sessions. As it can be seen in Fig. 4.8. When comparing the results of the two LPs (**??**)(4.2) 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.

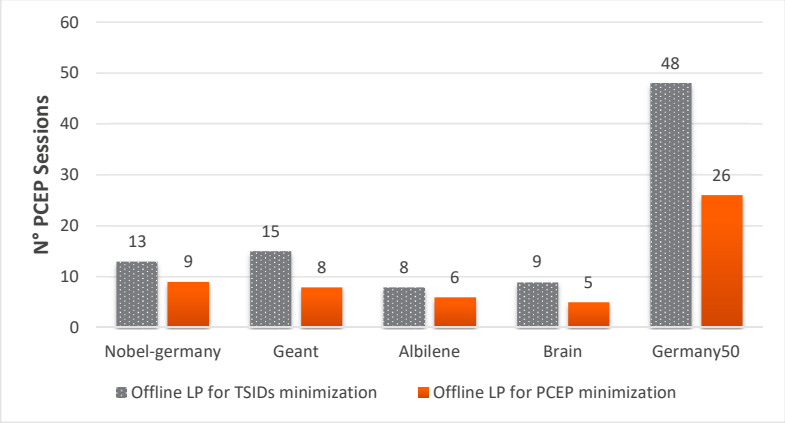


Figure 4.8: Comparison of the number of PCEP sessions required by the two offline LPs

4.6 CONCLUSION

In this chapter, we proposed the 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.*, it 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.

Chapter 5

General Conclusion and future work

Our work is placed in the context of the MPLS instantiation of the Segment Routing architecture. This thesis has investigated how service providers can leverage the Segment Routing architecture to deliver traffic engineering capabilities. Service providers are under important stress to scale their networks to meet their customers' requirements in terms of volume and QoS. Historically the volume of traffic handled by networks never saw a decline, and it continues to grow at an accelerated rate especially with the increase of geographically distributed datacenter, and the explosion of connected people and objects. Consequently, service providers are facing the challenge to increase their network throughput capacity while reducing CapEx. In this context, traffic engineering is of a strategic importance to service providers because of its capacity to reduce costs by optimizing the utilization of the available network resources.

Service providers rely on MPLS as the defacto layer 3 WAN connectivity technology. Customers leverage it to provide IP level connectivity between their remote offices. With the Additional benefits of QoS capabilities natively supported in MPLS. For example, at a customer edge router, packets can be tagged to specify to what QoS class they belong. Consequently, the service provider core routers prioritize those packets based on their loss, jitter and latency limits. However, over the years, WAN networks became complicated to manage and troubleshoot due to the complexity added by the various uses cases supported. The incremental support of new use cases led to over-engineered solutions resulting in a complex control plane composed of multiple protocols with intersecting capabilities. The currently distributed control plane relays on a soft state model where all the network nodes have to maintain a consistent database of all the network information (topology, traffic engineering, tunnel, etc).

Consequently, the maintenance and the troubleshooting of currently deployed services or the support of new ones is hard and causes an increase in OpEx. For example, a service provider receives a client demand to connect to of its remote site using a VPN with a bandwidth of one gigabit/second, this requires to trigger the tunnel signaling using the RSVP-TE then the IGP is synchronized with RSVP-TE process in order propagate the reservation information so that all the network nodes update their traffic engineering database. The RSVP-TE continues to maintain the tunnel using periodic update messages. Additionally, the increase in the number of tunnels and the slow signaling and tearing down processes prevent the service providers from achieving network-wide traffic engineering. Thus, leading to a sub-optimal network resource utilization. A variety of solutions were proposed to reduce the severance of these side effect however the problem persisted. Consequently, service providers prefer to over-dimensioning their networks to avoid fully relying on such complex control plane. However, this approach is not sustainable in current and future markets especial with the drastic decline in the dedicated WAN connection prices due to the emergence of new technologies such as Software Defined WAN or SD-WAN.

Therefore, service providers have to simplify the design of their networks and optimize as possible the available resources. For the reasons mentioned before, service provider supports the idea of a simple network design based on a light control plane coupled with a logically centralized software intelligence. A simple control plane allows for an easy service instantiation and troubleshooting while the centralized software layer handles the complex tasks such as the maintenance of the traffic engineering database, path computation, and resource optimization. To concretize such model the IETF SPRING working group proposed the segment routing architecture which can be instantiated over two WAN data planes MPLS (SR-MPLS) and IPv6 (SR-v6). It has a light control plane because it relays on source routing to deliver end to end connectivity. SR standardization is driven jointly by vendors and service providers. Our work is placed in the context of the SR instantiation over the MPLS data plane (*i.e.*, SR-MPLS). SR simple control plane does not require new protocols, rather it extends existing and well-established routing protocols such as OSPF, ISIS, and BGP-LS. Consequently, the deployment of SR-MPLS requires only a software upgrade, this will speed up SR deployment and encourages service providers to adopt it as no big investment in hardware is required. Ultimately, SR-MPLS is projected to become the defacto standard to deploy MPLS-based WAN services. Therefore, service providers look to replace the conventional MPLS control plan that relies on

RSVP-TE and LDP for signaling and distributing labels with SR-MPLS. Additionally, SR fits in the new wave of Software Defined Networks where the forwarding is assured by the SR control plane and the path computation, resource management and optimization algorithms are performed by an SDN controller.

Field deployment of segment routing has already started. However, it is limited to use cases such as fast reroute, and shortest path forwarding. In our work, we focused on solving the problems that service providers face in order to achieve SR traffic engineering. The main problem that we identified is the Maximum Stack Depth (MSD) limitation which is a hardware limitation. In SR-MPLS, the path that a packet has to go through is encoded as label stack. The MSD is the maximum number of MPLS label a router can push onto a packet's header. Thus, rendering a considerable amount of network paths unusable. This prevents achieving traffic engineering and leads to sub-optimal network resource utilization.

SR Path Efficient Label Stack Encoding

In this thesis, we studied SR path label stack encoding. Costumers request WAN connection with specific QoS requirements. The demand is formulated based on the traffic requirements in term of bandwidth, delay, jitter or loss. The service provider may impose manually a path but in most cases, it relates on a path Computation Engine (PCE) which implement intelligent computation algorithms and relies on its traffic engineering database that maintains an up to date knowledge of the topology and the available resources. The PCE receives the customer demand with its QoS criteria, then computes a path that is may not be the IGP shortest path. Furthermore, in traffic engineering, the PCE my trigger a network-wide resources optimization in order to optimize the traffic distribution in the network, costumers flow paths are rearranged to increase future demand acceptance rate. However, in both cases, we end up with long paths that when encoded as label stacks exceed the MSD. Therefore, optimization cannot be done. This seriously hampers the adoption of SR. In this work, we mainly focus on solving this problem.

First, we focused on developing efficient encoding algorithms that work natively and do not require any extensions, modify or alter the SR behavior. to solve this problem two label stack algorithms. Both of which performed very well against real-life network topologies. To evaluate their performance, we first solved the linear program to satisfy the demand matrix which resulted in a set of paths, then we used the two algorithms to reduce the size of the label stack. The resulted label stack is

the optimal smallest stack. The simulation results show an important decrease in the number of paths encoded with label stack greater than the MSD. Encouraged by the good results, we followed with an implementation of our algorithms in the OpenDayLight SDN controller, which allowed us to demonstrate, even more, the real-life application of such techniques.

A new label encoding approach

The encoding algorithms allowed to reduce the size of the SR path label stack. However, we still have some paths of the network that can not be encoded with a label stack less or equals to the MSD. This led to propose a new label encoding approach. This requires the introduction of a new segment type that is the Targeted Segment Identifier (TSID). This TSID replaced a sub-set of labels in the label stack. TSIDs are installed on specific network nodes. When the packet reaches that node the TSID get replaced by labels it replaced initially in the stack. The TSIDs architecture relays on the PCE to install new TSIDs additionally maintains a database of already installed TSIDs. We proposed online algorithms and linear programs to minimize the number of TSIDs installed into the network and the number of sessions the PCE have to maintain with the network nodes. We couple the TSIDs architecture with the encoding algorithms to achieve the best results.

5.1 Future Work

In this thesis, we tackled the problem of optimizing the encoding SR paths. In fact, SR relies mainly on source routing for packets forwarding. Therefore, the encoding of the SR paths is a key component of the technology, which demonstrates the importance of the contributions presented in this work. We provided solutions that reduce the impact of the MSD hardware limitation and therefore allows for a maximum utilization of the network resources. Consequently, This encourages service providers to deploy SR and extend its use case to include traffic engineering.

Enhanced Fast Rerout using Segment Routing Path encoding

The encoding of SR paths is going to play an important role in any current or future use cases. Consequently, The proposed algorithms can be extended to enable and

power these use cases. Actually, we are currently working on adapting the encoding algorithms presented here to the Fast Reroute protection paths. This work is inspired by the TI-LFA (Topology-Independant Loop Free Alternate) standard [40]. In order to anticipate a failure, protection paths are computed on the post-convergence topology graph in case of that failure occurring. That raises new challenges that the new encoding algorithms have to address such as forwarding loops that may occur due to label stack encoding errors.

In TI-LFA, a node pre-computes a single post-convergence shortest path that will be used upon a failure (*e.g.*, link failure) to forward the traffic. However, this approach does not consider the link utilization (*i.e.*, load) when computing the protection a path. This introduces a high risk of congestion due to switching the traffic impacted by the failure to links that do not have enough capacity. The solution we are studying is to consider the links load and to compute multiple protection paths. Upon a failure, each path will be used to forward a pre-determined ratio of the failure traffic.

Segment Routing for Service Chaining

A second perspective we are considering is the encoding of service chaining paths. Indeed, SR provides unprecedented flexibility to direct traffic flows through several network functions natively. This fits well with Network Functions Virtualization (NFV)s. As detailed in [71], in SR-MPLS, Each NFV is considered a segment and therefore has a SID assigned to it. However, this would increase the size of the stack to express the service path. Consequently, we are considering to extend our encoding algorithms for the expression of service paths. This extended version needs to take into account that the service labels cannot be reduced from the label stack. Additionally, an NFV may change the traffic flow profile and therefore its QoS requirements, this has to be reflected in the label stack in order to assure that the flow is handled with appropriate QoS.

TSID for Inter-AS path stitching

Finally, the TSID architecture can be generalized for inter-Autonomous System (AS) path encoding/stitching. Currently, a TSID is associated with a local network path. This can be extended to include paths provided by remote AS. For example, a service provider can send it traffic via a remote AS, without any knowledge of its network

topology graph, it has to assure that the top label is the TSID when the flow packets reach the remote AS ingress node.

Publications

1. Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Label encoding algorithm for mpls segment routing. In *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*, pages 113–117. IEEE, 2016

2. Olivier Dugeon, Rabah Guedrez, Samer Lahoud, and Géraldine Texier. Demonstration of segment routing with sdn based label stack optimization. In *Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on*, pages 143–145. IEEE, 2017

3. Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. A new method for encoding mpls segment routing te paths. In *Network of the Future (NOF), 2017 8th International Conference on th*, pages 58–65. IEEE, 2017

Posters and Demonstrations

1. Rabah Guedrez and Tantsura Jeff. The critical role of maximum sid depth (msd) hardware limitations in segment routing ecosystem and how to work around those. SDN+MPLS+NFV 2017 world congress, 2017

2. Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Demonstration of segment routing with sdn based label stack optimization. North American Network Operators' Group (NANOG), 2017

3. Rabah Guedrez and Tantsura Jeff. The critical role of maximum sid depth (msd) hardware limitations in segment routing ecosystem and how to work around those. NANOG 2017, 2017

4. Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Poster: Efficient encoding of segment routing paths. Orange Labs doctoral students day, 2017

Appendices

Appendix A

Segment Routing over IPv6 Data Plane

SR instantiation over the IPv6 data plane (SR-IPv6) uses IPv6 addresses as Segment IDs and the SRGB represents the pool of routable IPv6 addresses in the SR domain. Consequently, all the SIDs are globally significant IPv6 addresses.

SR-MPLS does not require any modification to the MPLS data plane. However, for SR-IPv6 to support source routing, the IPv6 header is extended and a special behavior is defined. This means that service providers may have to buy new hardware in order to deploy SR-IPv6 because equipment vendors use Application-Specific Integrated Circuits (ASICs) to do packet processing.

In SR-MPLS in order to process and forward the packet to the next segment (topological or service/application-based), the node has to POP (NEXT) the top label, leading to gradually lose the SR path as the packet goes through the network. In the SR-IPv6, the SR path encoded into the packet header is preserved in the IPv6 packet header when going through the network. Thus, the full path travels with the packet to its destination. Therefore, at any point of the path, the current node can determine the ingress and egress nodes of the packet as well as all intermediate nodes.

Several deployment scenarios can be identified for the SR-IPv6. For example, where the MPLS data plane is not available, or combined with the MPLS data plane: the core network implements the SR-MPLS while the home networks or the data center deploy the SR-IPv6.

A.0.1 SR-IPv6 Terminology

SR-IPv6 requires the adaptation of the generic building blocks of the SR architecture to the IPv6 data plane. The concepts of the SR architecture are implemented for the SR-IPv6 data plane as follows:

- The *SRGB* is the IPv6 addresses space reserved for the SR domain. Those addresses are routable by all the SPRING nodes.
- An *SID* is an IPv6 address.
- The *Prefix-SID* is an IPv6 address, routable by all the SPRING nodes.
- The *Global SID* is an IPv6 address within the SRGB. All the SPRING nodes must be capable of processing a global SID.
- The *Local SID* is the IPv6 addresses that are outside the SRGB, *i.e.*, not routable. Only the node advertising it is able to process it.
- The *Node-SID* is an IPv6 address of the node's loopback interface. It is a routable address in the SR domain, therefore, a Node-SID is considered as a global SID.
- The *Adj-SID* is an IPv6 address of a node interface. By default, it is advertised as a local SID, unless specified otherwise.

A.0.2 Segment Routing Header

With the deprecation of the type 0 Routing Header (RH0) [25], IPv6 nodes had no other method to source route their packets. For this reason, the IETF 6MAN WG defined the new Segment Routing Header (SRH) [27] (Type 4 is suggested to Internet Assigned Numbers Authority (IANA)). This enables IPv6 nodes to source route their packets by listing the IPv6 addresses of the intermediate nodes loopbacks and interfaces. SRH uses the HMAC security mechanism [78] in order to avoid the security flows similar to RH0.

The SRH depicted in Fig. A.1, maintains the list of segments that compose the SR path as a list of IPv6 addresses inside the *Segment List*$[n]$ field. The segments are encoded in the *Segment List*$[n]$ in a reverse order. Example: for an SR path composed of n segment, *Segment List*$[0]$ contains the last segment of the SR path and the *Segment List*$[n-1]$ contains the first one.

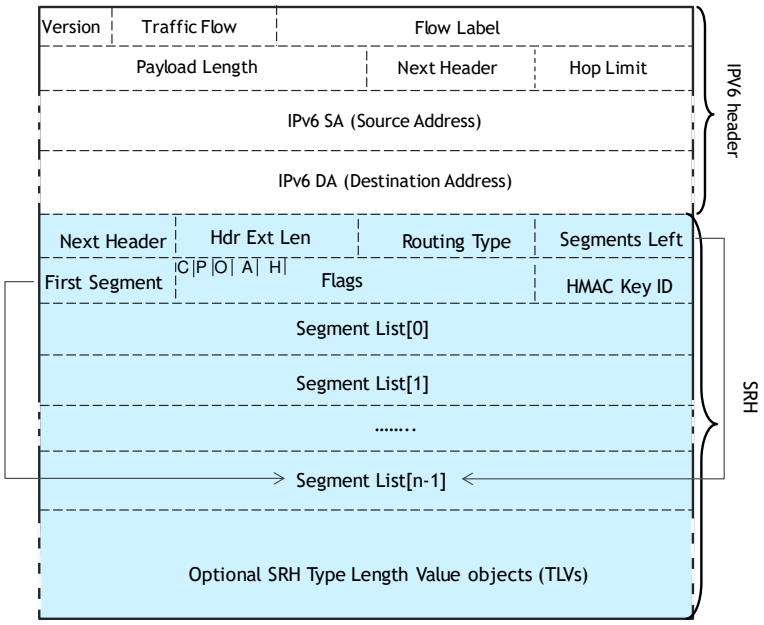


Figure A.1: Header of IPv6 packet with Segment Routing

The two pointers *Segments Left* and *First Segment* are used to go through the *Segment List*:

- *Segments Left* is an index used to point to the active SID in the SR path: $Segment\ List[Segments\ Left] == active\ SID$. Initially, it points to the first SID and its value is set to $N-1$, where N is the number of SIDs that compose the SR path. After a segment is processed, the pointer gets decremented to point to the next one to be inspected.

- *First Segment* is an index that always points to the first SID of the SR path. Initially, its value is set to $N-1$.

The flag field encapsulates several flags, each flag has an associated action that needs to be performed over the packet. For example, the *Clean-up flag* is the first bit in the Flags field. It indicates that the SRH is inserted into the client packet header according to Fig. A.2 (a). Consequently, the node prior to the last segment of the SR path knows that it has to remove the SRH. This behavior is similar to PHP in MPLS.

SRH Type Length Value (TLVs): SRH may include several additional information encoded as TLVs, each has its own role and meaning. The use of a TLV format makes

extending the SRH easier and more modular. One of the important defined TLVs is the HMAC TLV. The presence of the optional HMAC is indicated by the h-flag in the flags field. The service provider may choose to make it obligatory for all the traffic generated from outside its network to have a valid HMAC. In fact, HMAC is used to ensure that only trusted nodes from outside the service provider network participate in SR, nodes configured with a pre-shared key can send traffic into the service provider network, the use of HMAC eliminates the source routing security threats cause of the deprecation of the RH0 described in [25]. The HMAC is the output of a hashing algorithm [78] performed over the packet header, the ingress nodes of the service provider network drop traffic with invalid HMAC. Traffic generated from inside the operator's network (*e.g.*, by provider edge and transit nodes) is not required to add an HMAC TLV into the SRH, because service provider SPRING nodes are considered as trusted.

A.0.3 SR-IPv6 forwarding operations

SPRING nodes can perform the following set of operations on the client packets:

- *PUSH:* It is performed by the ingress nodes to encode the SRH into the packet. Two methods may be used to perform the push operation:

  1. The SRH is added at the end of the IPv6 extension header of the client packet as shown in Fig. A.2 (a).
  2. The client packets get encapsulated in a new IPv6 packet. The SRH is carried in the extended header of newly added IPv6 header as shown in Fig. A.2 (b), and this option allows service providers to tunnel the client traffic through the network without any changes.

  Intermediate nodes may perform the push operation in order to add a list of IPv6 addresses to existing SRH, for rerouting and protection purposes. After each PUSH operation, the pointer *Segments Left* is reset to point to the bottom of the stack (*Segment List[N-1]*).

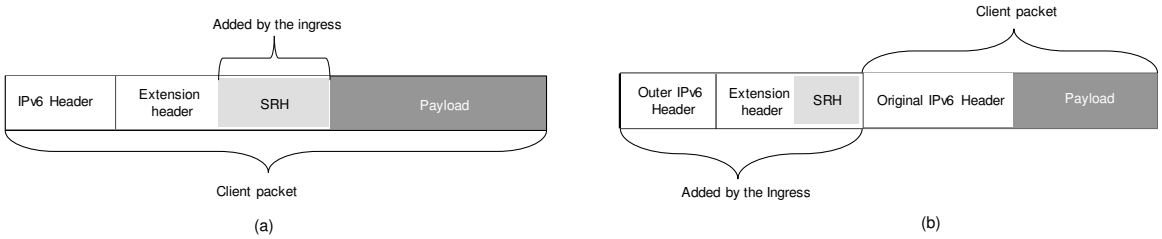


Figure A.2: illustrates how the SRH can be added to a client IPv6 packet :
a) SRH added at the end of the original IPv6 packet extension header.
b) client packet is encapsulated into a new IPv6 packet, the SRH is added into the extension header of the new IPv6 packet

- *CONTINUE:* it is the regular IPv6 forwarding operation, which is based on the DA located in the IPv6 header.

- *NEXT:* It is performed when the DA of the received packet belongs to the current node (*e.g.*, matches its loopback address). Fig. A.3 illustrates its execution process. Once a packet is received, the node starts at step 100 by checking if the packet's DA belongs to itself or not. If NO, the packet is forwarded based on that DA (*i.e.*, no need to process the SRH). If YES, that means that the packet's DA belongs to the current node. At step 200, the node starts processing the SRH. At step 201, the node checks if the *H-flag* if set, if Yes, then the *HMAC* TLV is present in the SRH. At step 211, the node then checks the validity of the *HMAC* field (*i.e.*, the packet is generated from a trusted source). If the *HMAC* is invalid, the packet is dropped. At step 202, the node checks if there are *Segments Left* in the SR path by checking if the *Segments Left* $>$ *0*. At step 203, the node decrements the *Segments Left* pointer by one. At step 204, the packet DA is replaced by the one pointed to by *Segments Left* in the *Segment List i.e.*, DA = *Segment List [Segments Left]*. At step 205, check if the *Segments Left* pointer is equal to zero. If YES, the active segment is the last one in the SR path. At step 206, the node checks if the c-flag is set. If Yes, the SRH has to be removed before sending the packet to the last segment. Finally, the packet is forwarded based on its *DA*.

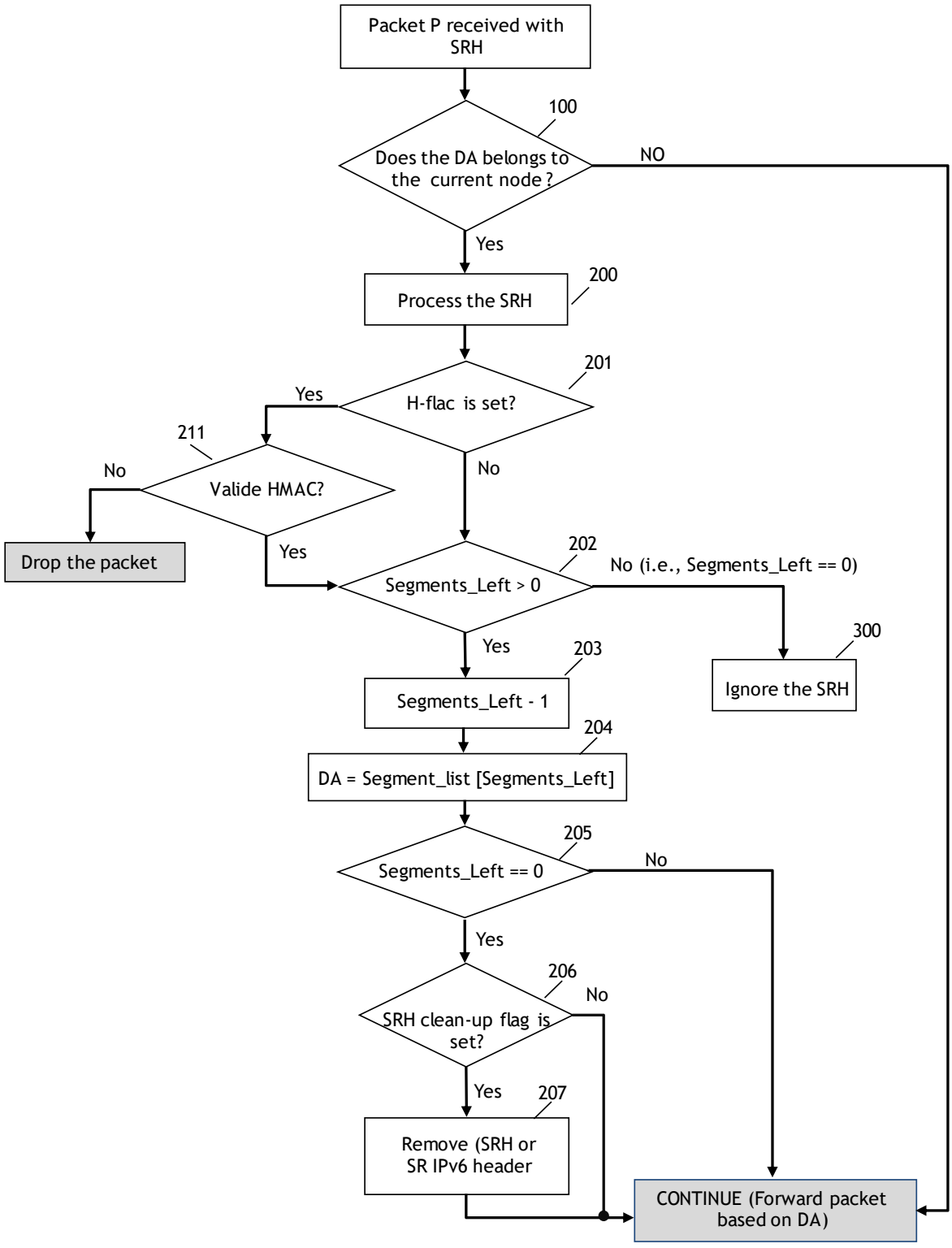


Figure A.3: State Machine of IPv6 node forwarding behavior when using Segment Routing

Let us see in Fig. A.4 how $PE1$ choses to forward the SR-IPv6 lient traffic over the path: $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$. Therefore, $PE1$ encapsulates client packets in IPv6 packets as shown in Fig. 2.3, the SRH is carried in the newly added header. Segments List field of the SRH carries the IPv6 addresses that compose the SR path, The SR path may be expressed as a loose or a strict path: the strict path for this example would be $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$ where all the intermediate nodes are listed in the Segments List field, a loose path only lists the intermediate nodes that are necessary to express the initial SR path. in order to reduce the size of the SR path the node's routing table (or the shortest path tree of a centralized controller) can be used. For example, $PE1$'s routing table has the shortest path between $PE1$ and $P5$ is through $P6$, this information can be used to express the initially computed SR path as $PE1 \rightarrow P5 \rightarrow P3 \rightarrow PE4$, the loose path represents the same topological path as the strict one *i.e.*, $PE1 \rightarrow P6 \rightarrow P5 \rightarrow P3 \rightarrow PE4$.

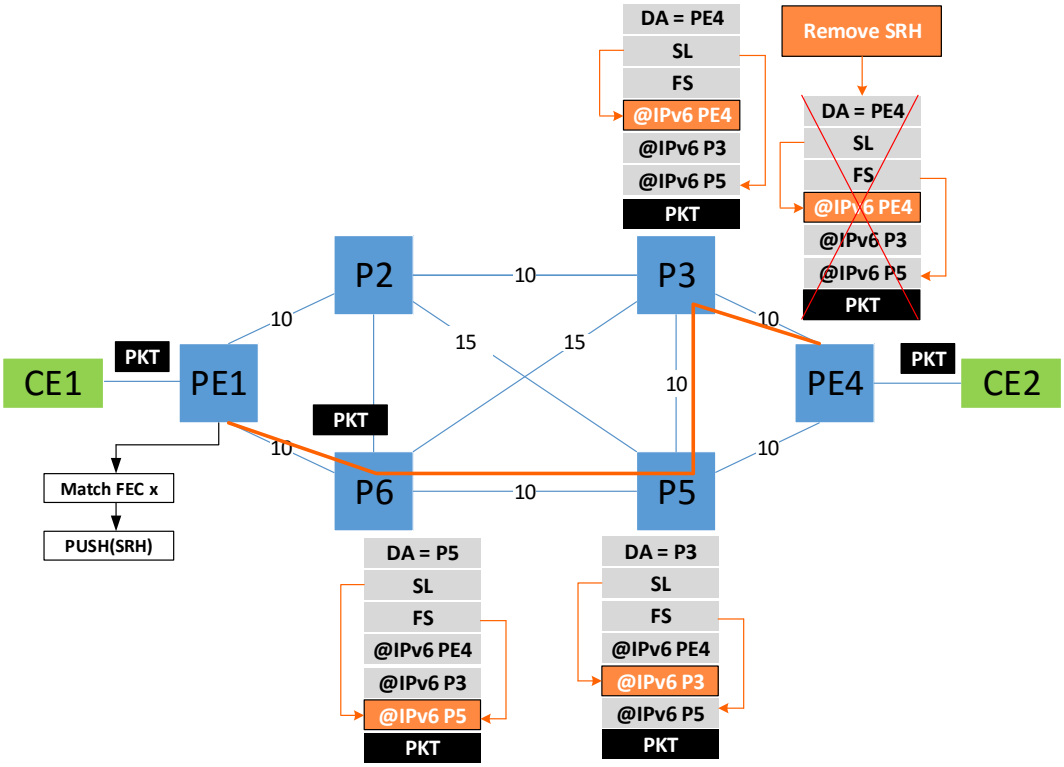


Figure A.4: Example of Segment Routing path when using IPv6 data plane

The fields in SRH have these initial values: DA = $P5$ (*i.e.*, take the shortest path

to reach $P5$), *Segment list* $[PE4, P3, P5]$, $Segments\ Left = 2$ (points to the active segment: $P5$), *First Segment = 2* (always points to the $First\ Segment$: $P5$).

$PE1$ sets the packets DA to $P5$'s IPv6 address. From $PE1$'s routing table, the next hop to reach $P5$ is $P6$, $PE1 - P6$ direct link is used to forward the packets. At $P6$, the node detects that the packets are not destined to itself (DA != $P6$). Therefore, $P6$ forward the packets to $P5$ via the direct link. At $P5$, the node detects that the packets are destined to itself (DA == $P5$), $P5$ execute the CONTINUE operation (depicted in Fig. A.3.): it decrements the pointer $Segments\ Left$ (*i.e.*, $Segments\ Left == 1$) which now points to $P3$ (*i.e.*, $Segments\ Left\ [1] == P3$), then copies the address of $P3$ in the DA field and forwards the packet based on that address. The same process is repeated at $P3$. At PE4, $Segments\ Left$ value is 0, based on that PE4 conclude that it is the last segment, then removes the SR IPv6 header including the SRH before forwarding the packet using its original DA to $CE2$.

Acronyms

6man IPv6 Maintenance

Adj-SID Adjacency Segment

ASIC Application-Specific Integrated Circuits

BGP Border Gateway Protocol

BGP-LS BGP Link State

CE Customer Edge

CLI Command Line Interface

DA Destination Address

DPI Deep Packet Inspection

DSR Dynamic Source Routing

ECMP Equal-cost multi-path

EL Entropy Labels

ENTC European Advanced Networking Test Center

FRR Fast Reroute

HMAC key-hashed message authentication code

IETF Internet Engineering Task Force

IGP Internal Gateway Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISIS IS-IS for IP Internets

LDP Label Distribution Protocol

LER Label Edge Router

LFIB Label Forwarding Information Base

LSA Link State Advertisement

LSP Label Switched Path

LSR Label Switching Routers

LSRR Loose Source and Record Route

MP-BGP Multi-Protocol BGP

MPLS Multi-Protocol Label Switching

MPLS-TE MPLS Traffic Engineering

MSD Maximum SID Depth

Node-SID Node Segment

OAM Operation and Maintenance

OSPF Open Shortest Path First

PCC Path Computation Clients

PCE Path Computation Element

PCEP PCE communication Protocol

PE Provider Edge Router

PHP Penultimate hop popping

PMS Path Monitoring System

Prefix-SID Prefix Segment

QoS Quality of Service

RSVP Resource Reservation Protocol

RSVP-TE Resource Reservation Protocol - Traffic Engineering

RTT Round-Trip Time

SID Segment Identifier

SPF Shortest Path First

SPRING Source Packet Routing in Networking

SR Segment Routing

SR-IPv6 Segment Routing over IPv6 data plane

SR-MPLS Segment Routing over MPLS data plane

SRGB Segment Routing Global Block

SRH Segment Routing Header

SRMS Segment Routing Mapping Server

SRP Source-Routed Path

SSRR Strict Source and Record Route

TE Traffic Engineering

TED Traffic Engineering Database

TLV Type Length Value

VPN Virtual private network

VRF Virtual Routing Function

WG Working Group

Bibliography

[1] Yakov Rekhter, Alex Conta, Guy Fedorkow, Eric Rosen, Dino Farinacci, and Tony Li. MPLS Label Stack Encoding. RFC 3032, March 2013.

[2] Kireeti Kompella and Yakov Rekhter. Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE). RFC 4206, October 2005.

[3] Harmen van der Linde and Thomas Nadeau. MPLS/BGP Layer 3 Virtual Private Network (VPN) Management Information Base. RFC 4382, February 2006.

[4] Siva Sivabalan, Jan Medved, Clarence Filsfils, Victor Lopez, Jeff Tantsura, Wim Henderickx, Edward Crabbe, and Jonathan Hardwick. PCEP Extensions for Segment Routing. Internet-Draft draft-ietf-pce-segment-routing-07, Internet Engineering Task Force, March 2016. Work in Progress.

[5] Evgeny Tantsura and Gregory Mirsky. Using border gateway protocol to expose maximum segment identifier depth to an external application, January 5 2017. US Patent App. 14/846,342.

[6] 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.

[7] Peter Psenak, Stefano Previdi, Clarence Filsfils, Hannes Gredler, Rob Shakir, Wim Henderickx, and Jeff Tantsura. OSPF Extensions for Segment Routing. Internet-Draft draft-ietf-ospf-segment-routing-extensions-25, Internet Engineering Task Force, April 2018. Work in Progress.

[8] Stefano Previdi, Les Ginsberg, Clarence Filsfils, Ahmed Bashandy, Hannes Gredler, Stephane Litkowski, Bruno Decraene, and Jeff Tantsura. IS-IS Ex-

tensions for Segment Routing. Internet-Draft draft-ietf-isis-segment-routing-extensions-19, Internet Engineering Task Force, July 2018. Work in Progress.

[9] Stefano Previdi, Peter Psenak, Clarence Filsfils, Hannes Gredler, Mach Chen, and Jeff Tantsura. BGP Link-State extensions for Segment Routing. Internet-Draft draft-gredler-idr-bgp-ls-segment-routing-ext-01, Internet Engineering Task Force, December 2015. Work in Progress.

[10] Adrian Farrel, Olufemi Komolafe, and Seisho Yasukawa. An Analysis of Scaling Issues in MPLS-TE Core Networks. RFC 5439, October 2015.

[11] *Internet Qos: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers In, 2001.

[12] George Swallow, Lou Berger, Der-Hwa Gan, Franco Tommasi, Simone Molendini, and Ping Pan. RSVP Refresh Overhead Reduction Extensions. RFC 2961, March 2013.

[13] Alia Atlas, George Swallow, and Ping Pan. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090, October 2015.

[14] Stefano Previdi, Clarence Filsfils, Bruno Decraene, Stephane Litkowski, Martin Horneffer, and Rob Shakir. Source Packet Routing in Networking (SPRING) Problem Statement and Requirements. RFC 7855, May 2016.

[15] Clarence Filsfils, Stefano Previdi, Bruno Decraene, Stephane Litkowski, and Rob Shakir. Segment Routing Architecture. Internet-Draft draft-ietf-spring-segment-routing-08, Internet Engineering Task Force, May 2016. Work in Progress.

[16] Clarence Filsfils, Nagendra Kumar Nainar, Carlos Pignataro, Juan Camilo Cardona, and Pierre Francois. The segment routing architecture. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[17] Interoperability feasibility showcase 2015 white paper. MPLS SDN World Congress, 2015.

[18] Internet Engineering Task Force. *RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification*, September 1981.

[19] Eric Rosen and Ross Callon. Multiprotocol Label Switching Architecture. RFC 3031, March 2013.

[20] Dr. Steve E. Deering. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, March 2013.

[21] Dave A. Maltz and David C. Johnson. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, March 2013.

[22] Dr. Deborah Estrin, Dr. Tony Li, Yakov Rekhter, Kannan Varadhan, and Daniel M.A. Zappala. Source Demand Routing: Packet Format and Forwarding Specification (Version 1). RFC 1940, March 2013.

[23] Fernando Gont. Security Assessment of the Internet Protocol Version 4. RFC 6274, October 2015.

[24] Fernando Gont, R. Atkinson, and Carlos Pignataro. Recommendations on Filtering of IPv4 Packets Containing IPv4 Options. RFC 7126, October 2015.

[25] George Neville-Neil, Pekka Savola, and Joe Abley. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095, October 2015.

[26] David Culler, Jonathan Hui, JP Vasseur, and Vishwas Manral. An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6554, October 2015.

[27] Stefano Previdi, Clarence Filsfils, Brian Field, Ida Leung, J. Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, and David Lebrun. IPv6 Segment Routing Header (SRH). Internet-Draft draft-ietf-6man-segment-routing-header-01, Internet Engineering Task Force, March 2016. Work in Progress.

[28] Philip A. Shafer. An Architecture for Network Management Using NETCONF and YANG. RFC 6244, October 2015.

[29] Stephane Litkowski, Yingzhen Qu, and Jeff Tantsura. YANG Data Model for Segment Routing. Internet-Draft draft-ietf-spring-sr-yang-02, Internet Engineering Task Force, March 2016. Work in Progress.

[30] Clarence Filsfils, Stefano Previdi, Ahmed Bashandy, Bruno Decraene, and Stephane Litkowski. Segment Routing interworking with LDP. Internet-Draft draft-ietf-spring-segment-routing-ldp-interop-09, Internet Engineering Task Force, September 2017. Work in Progress.

[31] Alex D. Zinin, Igor Bryskin, and Lou Berger. The OSPF Opaque LSA Option. RFC 5250, October 2015.

[32] Luc De Ghein. *MPLS Fundamentals*. Cisco Press, 2006.

[33] Alessio Giorgetti, Piero Castoldi, Filippo Cugini, Jeroen Nijhof, Francesco Lazzeri, and Gianmarco Bruno. Path encoding in segment routing. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[34] Francesco Lazzeri, Gianmarco Bruno, Jeroen Nijhof, Alessio Giorgetti, and Piero Castoldi. Efficient label encoding in segment-routing enabled optical networks. In *Optical Network Design and Modeling (ONDM), 2015 International Conference on*, pages 34–38. IEEE, 2015.

[35] Hannes Gredler, Clarence Filsfils, Stefano Previdi, Bruno Decraene, Martin Horneffer, and Pushpasis Sarkar. Anycast Segments in MPLS based Segment Routing. Internet-Draft draft-psarkar-spring-mpls-anycast-segments-02, Internet Engineering Task Force, April 2016. Work in Progress.

[36] Clarence Filsfils, Stefano Previdi, Ahmed Bashandy, Bruno Decraene, and Stephane Litkowski. Segment Routing interworking with LDP. Internet-Draft draft-ietf-spring-segment-routing-ldp-interop-01, Internet Engineering Task Force, April 2016. Work in Progress.

[37] Chris Bowers, Hannes Gredler, and Uma Chunduri. Advertising LSPs with Segment Routing. Internet-Draft draft-bowers-spring-advertising-lsps-with-sr-02, Internet Engineering Task Force, November 2015. Work in Progress.

[38] John Drake, Shane Amante, Wim Henderickx, Lucy Yong, and Kireeti Kompella. The Use of Entropy Labels in MPLS Forwarding. RFC 6790, October 2015.

[39] Kireeti Kompella, Siva Sivabalan, Stephane Litkowski, Rob Shakir, Sriganesh Kini, and jefftant@gmail.com. Entropy labels for source routed tunnels with label stacks. Internet-Draft draft-ietf-mpls-spring-entropy-label-03, Internet Engineering Task Force, April 2016. Work in Progress.

[40] Clarence Filsfils, Ahmed Bashandy, Bruno Decraene, and Pierre Francois. Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-francois-spring-segment-routing-ti-lfa-02, Internet Engineering Task Force, February 2016. Work in Progress.

[41] Luyuan Fang. LDP IGP Synchronization. RFC 5443, May 2016.

[42] A Sgambelluri, F Paolucci, A Giorgetti, F Cugini, and P Castoldi. Experimental demonstration of segment routing. *Journal of Lightwave Technology*, 34(1):205–212.

[43] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. Traffic engineering with segment routing: Sdn-based architectural design and open source implementation. In *2015 Fourth European Workshop on Software Defined Networks*, pages 111–112. IEEE, 2015.

[44] 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*, pages 1–3. IEEE, 2015.

[45] 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*, pages 1–4. IEEE, 2015.

[46] Jan Medved, Ina Minei, Edward Crabbe, and Robert Varga. PCEP Extensions for Stateful PCE. Internet-Draft draft-ietf-pce-stateful-pce-14, Internet Engineering Task Force, March 2016. Work in Progress.

[47] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. Traffic engineering with segment routing: Sdn-based architectural design and open source implementation.

[48] Kireeti Kompella and George Swallow. Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures. RFC 4379, October 2015.

[49] Thomas Nadeau, Rahul Aggarwal, Kireeti Kompella, and George Swallow. Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs). RFC 5884, October 2015.

[50] Nagendra Kumar, Carlos Pignataro, Nobo Akiya, Ruediger Geib, Greg Mirsky, and Stephane Litkowski. OAM Requirements for Segment Routing Network. Internet-Draft draft-ietf-spring-sr-oam-requirement-01, Internet Engineering Task Force, December 2015. Work in Progress.

[51] Ruediger Geib, Clarence Filsfils, Carlos Pignataro, and Nagendra Kumar. A Scalable and Topology-Aware MPLS Dataplane Monitoring System. Internet-Draft draft-ietf-spring-oam-usecase-03, Internet Engineering Task Force, April 2016. Work in Progress.

[52] Eric C. Rosen and Loa Andersson. Framework for Layer 2 Virtual Private Networks (L2VPNs). RFC 4664, September 2006.

[53] Jeff Tantsura, Uma Chunduri, Sam Aldrin, and Peter Psenak. Signaling MSD (Maximum SID Depth) using OSPF. Internet-Draft draft-ietf-ospf-segment-routing-msd-13, Internet Engineering Task Force, May 2018. Work in Progress.

[54] Jeff Tantsura, Uma Chunduri, Sam Aldrin, and Les Ginsberg. Signaling MSD (Maximum SID Depth) using IS-IS. Internet-Draft draft-ietf-isis-segment-routing-msd-12, Internet Engineering Task Force, May 2018. Work in Progress.

[55] Siva Sivabalan, Clarence Filsfils, Jeff Tantsura, Wim Henderickx, and Jonathan Hardwick. PCEP Extensions for Segment Routing. Internet-Draft draft-ietf-pce-segment-routing-11, Internet Engineering Task Force, November 2017. Work in Progress.

[56] Jeff Tantsura, Uma Chunduri, Gregory Mirsky, and Siva Sivabalan. Signaling Maximum SID Depth using Border Gateway Protocol Link-State. Internet-Draft draft-ietf-idr-bgp-ls-segment-routing-msd-01, Internet Engineering Task Force, October 2017. Work in Progress.

[57] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2, pages 519–528 vol.2, 2000.

[58] Josselin Vallet and Olivier Brun. Online ospf weights optimization in ip networks. *Comput. Netw.*, 60:1–12, February 2014.

[59] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. *Networks*, 55(3):276–286, 2010.

[60] Michal Pióro and Deepankar Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[61] Segment routing configuration guide, cisco ios xe fuji 16.7.x - advertise maximum sid depth by is-is and ospf to bgp-ls [cisco asr 1000 series aggregation services routers] - cisco. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-16-7/segrt-xe-16-7-book/sr-ad-max-SID-depth-is-is-ospf-bgp-ls.html. (Accessed on 05/28/2018).

[62] Opendaylight controller:rest reference and authentication - opendaylight project. https://wiki.opendaylight.org/view/OpenDaylight_Controller:REST_Reference_and_Authentication. (Accessed on 05/28/2018).

[63] Cisco. Pathman-sr. https://github.com/CiscoDevNet/pathman-sr, 2016.

[64] MS Windows NT kernel description. https://frrouting.org/. Accessed: 2018-11-10.

[65] Peter Psenak, Stefano Previdi, Clarence Filsfils, Hannes Gredler, Rob Shakir, Wim Henderickx, and Jeff Tantsura. OSPF Extensions for Segment Routing. Internet-Draft draft-ietf-ospf-segment-routing-extensions-25, Internet Engineering Task Force, April 2018. Work in Progress.

[66] Enke Chen, Tony J. Bates, and Ravi Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456, April 2006.

[67] FRRouting (FRR). Frrouting (frr) is an ip routing protocol suite for linux and unix platforms. https://github.com/FRRouting/frr. (Accessed: 2018-11-10).

[68] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Label encoding algorithm for mpls segment routing. In *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*, pages 113–117. IEEE, 2016.

[69] Siva Sivabalan, Clarence Filsfils, Jeff Tantsura, Wim Henderickx, and Jonathan Hardwick. PCEP Extensions for Segment Routing. Internet-Draft draft-ietf-pce-segment-routing-14, Internet Engineering Task Force, October 2018. Work in Progress.

[70] Siva Sivabalan, Jeff Tantsura, Clarence Filsfils, Stefano Previdi, Jonathan Hardwick, and Dhruv Dhody. Carrying Binding Label/Segment-ID in PCE-based Networks. Internet-Draft draft-sivabalan-pce-binding-label-sid-04, Internet Engineering Task Force, March 2018. Work in Progress.

[71] Francois Clad, Xiaohu Xu, Clarence Filsfils, daniel.bernier@bell.ca, Cheng Li, Bruno Decraene, Shaowen Ma, Chaitanya Yadlapalli, Wim Henderickx, and Stefano Salsano. Service Programming with Segment Routing. Internet-Draft draft-xuclad-spring-sr-service-programming-00, Internet Engineering Task Force, July 2018. Work in Progress.

[72] Olivier Dugeon, Rabah Guedrez, Samer Lahoud, and Géraldine Texier. Demonstration of segment routing with sdn based label stack optimization. In *Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on*, pages 143–145. IEEE, 2017.

[73] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. A new method for encoding mpls segment routing te paths. In *Network of the Future (NOF), 2017 8th International Conference on th*, pages 58–65. IEEE, 2017.

[74] Rabah Guedrez and Tantsura Jeff. The critical role of maximum sid depth (msd) hardware limitations in segment routing ecosystem and how to work around those. SDN+MPLS+NFV 2017 world congress, 2017.

[75] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Demonstration of segment routing with sdn based label stack optimization. North American Network Operators' Group (NANOG), 2017.

[76] Rabah Guedrez and Tantsura Jeff. The critical role of maximum sid depth (msd) hardware limitations in segment routing ecosystem and how to work around those. NANOG 2017, 2017.

[77] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Poster: Efficient encoding of segment routing paths. Orange Labs doctoral students day, 2017.

[78] Dr. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, March 2013.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Rendre possible l'ingénierie de trafic dans les réseaux avec routage par segment

Mots clés : Routage par segment, MPLS, ingénierie de trafic, SR-MPLS, PCE, RSVP-TE.

Résumé : La majorité des grands opérateurs utilise la technologie MPLS pour gérer leur réseau via des protocoles de signalisation et de distributions de labels. Or, ces protocoles sont complexes à déployer, à maintenir et la résolution des pannes est souvent très difficile. L'IETF a initié la standardisation d'une architecture de routage par segments (Segment Routing) s'appuyant sur un plan de contrôle simple, léger, facile à gérer et instanciée sur MPLS ou IPv6. Cette architecture repose sur le concept de routage à la source, dans lequel l'en-tête des paquets transporte les indications du chemin à suivre pour atteindre sa destination. Adapté aux cas d'usages simples et offrant nativement une résistance aux pannes, les cas d'usages plus complexes exigent de résoudre des verrous technologiques pour lesquels nous proposons plusieurs solutions.

Dans cette thèse effectuée au sein d'Orange Labs, nous nous sommes intéressés à l'instanciation de l'architecture Segment Routing sur le plan de transfert MPLS et plus particulièrement à l'ingénierie de trafic, notamment avec réservation de ressources. Nous avons proposé des solutions aux problèmes liés à la limitation matérielle des routeurs actuels ne permettant pas l'expression de tous les chemins contraints. Ce travail est divisé en deux parties : (i) la proposition d'algorithmes de calcul et d'encodage de chemins de routage par segment afin de contourner les limitations matérielles. (ii) la définition des exigences architecturales et la construction d'une preuve de concept fonctionnelle. Enfin, cette thèse propose de nouvelles pistes d'études afin de consolider les outils d'ingénierie de trafic pour le routage par segment.

Title : Enabling Traffic Engineering Over Segment Routing.

Keywords : Segment Routing, Traffic Engineering, MPLS, SR-MPLS, PCE, RSVP-TE.

Abstract : Most major operators use MPLS technology to manage their network via signalling and label distribution protocols. However, these protocols are complex to deploy, maintain and troubleshooting is often very difficult. The IETF has initiated the standardization of a segment routing architecture based on a simple control plane, lightweight, easy-to-manage and instantiated on MPLS or IPv6. This architecture is based on the concept of source routing, in which the packet header carries the indications of the path to follow to reach its destination. Suitable for simple use cases and natively resistant to failure, more complex use cases require the resolution of technological issues for which we offer several solutions.

In this thesis carried out within Orange Labs, we were interested in the instantiation of the Segment Routing architecture on the MPLS transfer plan and more particularly in traffic engineering, particularly with resource reservation. We have proposed solutions to the problems related to the hardware limitation of current routers that do not allow the expression of all constrained paths. This work is divided into two parts: (i) the proposal of algorithms for computing and encoding segment routing paths in order to bypass hardware limitations. (ii) the definition of architectural requirements and the construction of a functional proof of concept. Finally, this thesis proposes new research issues to consolidate traffic engineering tools for segment routing.